

Network Working Group
Internet-Draft
Expires: December 26, 2006

B. Campbell, Ed.
Estacado Systems
R. Mahy, Ed.
Plantronics
C. Jennings, Ed.
Cisco Systems, Inc.
June 24, 2006

The Message Session Relay Protocol
draft-ietf-simple-message-sessions-15

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

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/lid-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 December 26, 2006.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This document describes the Message Session Relay Protocol, a protocol for transmitting a series of related instant messages in the context of a session. Message sessions are treated like any other media stream when set up via a rendezvous or session creation protocol such as the Session Initiation Protocol.

Table of Contents

- 1. Introduction 4
- 2. Conventions 5
- 3. Applicability of MSRP 5
- 4. Protocol Overview 6
- 5. Key Concepts 9
 - 5.1. MSRP Framing and Message Chunking 9
 - 5.2. MSRP Addressing 10
 - 5.3. MSRP Transaction and Report Model 10
 - 5.4. MSRP Connection Model 12
- 6. MSRP URLs 14
 - 6.1. MSRP URL Comparison 15
 - 6.2. Resolving MSRP Host Device 16
- 7. Method-Specific Behavior 16
 - 7.1. Constructing Requests 16
 - 7.1.1. Sending SEND Requests 18
 - 7.1.2. Sending REPORT Requests 21
 - 7.1.3. Generating Success Reports 21
 - 7.1.4. Generating Failure Reports 22
 - 7.2. Constructing Responses 23
 - 7.3. Receiving Requests 24
 - 7.3.1. Receiving SEND Requests 24
 - 7.3.2. Receiving REPORT Requests 26
- 8. Using MSRP with SIP and SDP 27
 - 8.1. SDP Connection and Media Lines 27
 - 8.2. URL Negotiations 28
 - 8.3. Path Attributes with Multiple URLs 29
 - 8.4. Updated SDP Offers 30
 - 8.5. Connection Negotiation 30
 - 8.6. Content Type Negotiation 31
 - 8.7. Example SDP Exchange 32
 - 8.8. MSRP User Experience with SIP 33
 - 8.9. SDP direction attribute and MSRP 34
- 9. Formal Syntax 34
- 10. Response Code Descriptions 36
 - 10.1. 200 36
 - 10.2. 400 37
 - 10.3. 403 37
 - 10.4. 408 37
 - 10.5. 413 37
 - 10.6. 415 37
 - 10.7. 423 37
 - 10.8. 481 37
 - 10.9. 501 38
 - 10.10. 506 38
- 11. Examples 38
 - 11.1. Basic IM Session 38

- 11.2. Message with XHTML Content 41
- 11.3. Chunked Message 41
- 11.4. System Message 41
- 11.5. Positive Report 42
- 11.6. Forked IM 42
- 12. Extensibility 45
- 13. CPIM Compatibility 45
- 14. Security Considerations 46
 - 14.1. Transport Level Protection 46
 - 14.2. S/MIME 48
 - 14.3. Using TLS in Peer to Peer Mode 48
 - 14.4. Other Security Concerns 50
- 15. IANA Considerations 51
 - 15.1. MSRP Method Names 52
 - 15.2. MSRP Header Fields 52
 - 15.3. MSRP Status Codes 52
 - 15.4. MSRP Port 53
 - 15.5. MSRP URL Schemes 53
 - 15.6. SDP Transport Protocol 53
 - 15.7. SDP Attribute Names 53
 - 15.7.1. Accept Types 53
 - 15.7.2. Wrapped Types 54
 - 15.7.3. Max Size 54
 - 15.7.4. Path 54
- 16. Contributors and Acknowledgments 55
- 17. References 55
 - 17.1. Normative References 55
 - 17.2. Informational References 56
- Authors' Addresses 58
- Intellectual Property and Copyright Statements 59

1. Introduction

A series of related instant messages between two or more parties can be viewed as part of a "message session", that is, a conversational exchange of messages with a definite beginning and end. This is in contrast to individual messages each sent independently. Messaging schemes that track only individual messages can be described as "page-mode" messaging, whereas messaging that is part of a "session" with a definite start and end is called "session-mode" messaging.

Page-mode messaging is enabled in SIP via the SIP [4] MESSAGE method [21]. Session-mode messaging has a number of benefits over page-mode messaging, however, such as explicit rendezvous, tighter integration with other media types, direct client-to-client operation, and brokered privacy and security.

This document defines a session-oriented instant message transport protocol called the Message Session Relay Protocol (MSRP), whose sessions can be negotiated with an offer or answer [3] using the Session Description Protocol (SDP [2]). The exchange is carried by some signaling protocol, such as the Session Initiation Protocol (SIP [4]). This allows a communication user agent to offer a messaging session as one of the possible media types in a session. For instance, Alice may want to communicate with Bob. Alice doesn't know at the moment whether Bob has his phone or his IM client handy, but she's willing to use either. She sends an invitation to a session to the address of record she has for Bob, sip:bob@example.com. Her invitation offers both voice and an IM session. The SIP services at example.com forward the invitation to Bob at his currently registered clients. Bob accepts the invitation at his IM client and they begin a threaded chat conversation.

When a user uses an IM URL, RFC 3861 [31] defines how DNS can be used to map this to a particular protocol to establish the session such as SIP. SIP can use an offer answer model to transport the MSRP URLs for the media in SDP. This document defines how the offer/answer exchange works to establish MSRP connections and how messages are sent across the MSRP protocol, but it does not deal with the issues of mapping an IM URL to a session establishment protocol.

This session model allows message sessions to be integrated into advanced communications applications with little to no additional protocol development. For example, during the above chat session, Bob decides Alice really needs to be talking to Carol. Bob can transfer [20] Alice to Carol, introducing them into their own messaging session. Messaging sessions can then be easily integrated into call-center and dispatch environments using third-party call control [19] and conferencing [18] applications.

This document specifies MSRP behavior only for peer-to-peer sessions, that is, sessions crossing only a single hop. MSRP relay devices [22] (referred to herein as "relays") are specified in a separate document.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [5].

This document consistently refers to a "message" as a complete unit of MIME or text content. In some cases, a message is split and delivered in more than one MSRP request. Each of these portions of the complete message is called a "chunk".

3. Applicability of MSRP

MSRP is not designed for use as a standalone protocol. MSRP MUST be used only in the context of a rendezvous mechanism meeting the following requirements:

The rendezvous mechanism MUST provide both MSRP URLs associated with an MSRP session to each of the participating endpoints. The rendezvous mechanism MUST implement mechanisms to protect the confidentiality of these URLs - they MUST NOT be made available to an untrusted third party or be easily discoverable.

The rendezvous mechanism MUST provide mechanisms for the negotiation of any supported MSRP extensions that are not backwards compatible.

The rendezvous mechanism MUST be able to natively transport im: URIs or automatically translate im: URIs [26] into the addressing identifiers of the rendezvous protocol.

To use a rendezvous mechanism with MSRP, an RFC MUST be prepared describing how it exchanges MSRP URLs and meets these requirements listed here. This document provides such a description for the use of MSRP in the context of SIP and SDP.

SIP meets these requirements for a rendezvous mechanism. The MSRP URLs are exchanged using SDP in an offer/answer exchange via SIP. The exchanged SDP can also be used to negotiate MSRP extensions. This SDP can be secured using any of the mechanisms available in SIP, including using the sips mechanism to ensure transport security

across intermediaries and S/MIME for end-to-end protection of the SDP body. SIP can carry arbitrary URIs (including im: URIs) in the Request-URI, and procedures are available to map im: URIs to sip: or sips: URIs. It is expected that initial deployments of MSRP will use SIP as its rendezvous mechanism.

4. Protocol Overview

MSRP is a text-based, connection-oriented protocol for exchanging arbitrary (binary) MIME[8] content, especially instant messages. This section is a non-normative overview of how MSRP works and how it is used with SIP.

MSRP sessions are typically arranged using SIP the same way a session of audio or video media is set up. One SIP user agent (Alice) sends the other (Bob) a SIP invitation containing an offered session-description which includes a session of MSRP. The receiving SIP user agent can accept the invitation and include an answer session-description which acknowledges the choice of media. Alice's session description contains an MSRP URL that describes where she is willing to receive MSRP requests from Bob, and vice-versa. (Note: Some lines in the examples are removed for clarity and brevity.)

Alice sends to Bob:

```
INVITE sip:bob@biloxi.example.com SIP/2.0
To: <sip:bob@biloxi.example.com>
From: <sip:alice@atlanta.example.com>;tag=786
Call-ID: 3413an89KU
Content-Type: application/sdp
```

```
c=IN IP4 atlanta.example.com
m=message 7654 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://atlanta.example.com:7654/jshA7we;tcp
```

Bob sends to Alice:

```
SIP/2.0 200 OK
To: <sip:bob@biloxi.example.com>;tag=087js
From: <sip:alice@atlanta.example.com>;tag=786
Call-ID: 3413an89KU
Content-Type: application/sdp
```

```
c=IN IP4 biloxi.example.com
m=message 12763 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://biloxi.example.com:12763/kjhd37s2s2;tcp
```

Alice sends to Bob:

```
ACK sip:bob@biloxi SIP/2.0
To: <sip:bob@biloxi.example.com>;tag=087js
From: <sip:alice@atlanta.example.com>;tag=786
Call-ID: 3413an89KU
```

MSRP defines two request types, or methods. SEND requests are used to deliver a complete message or a chunk (a portion of a complete message), while REPORT requests report on the status of a previously sent message, or a range of bytes inside a message. When Alice receives Bob's answer, she checks to see if she has an existing connection to Bob. If not, she opens a new connection to Bob using the URL he provided in the SDP. Alice then delivers a SEND request to Bob with her initial message, and Bob replies indicating that Alice's request was received successfully.

```
MSRP a786hjs2 SEND
To-Path: msrp://biloxi.example.com:12763/kjhd37s2s2;tcp
From-Path: msrp://atlanta.example.com:7654/jshA7we;tcp
Message-ID: 87652
Byte-Range: 1-25/25
Content-Type: text/plain
```

```
Hey Bob, are you there?
-----a786hjs2$
```

```
MSRP a786hjs2 200 OK
To-Path: msrp://atlanta.example.com:7654/jshA7we;tcp
From-Path: msrp://biloxi.example.com:12763/kjhd37s2s2;tcp
Byte-Range: 1-25/25
-----a786hjs2$
```

Alice's request begins with the MSRP start line, which contains a transaction identifier that is also used for request framing. Next she includes the path of URLs to the destination in the To-Path header field, and her own URL in the From-Path header field. In this typical case there is just one "hop", so there is only one URL in each path header field. She also includes a message ID which she can use to correlate status reports with the original message. Next she puts the actual content. Finally she closes the request with an end-line of seven hyphens, the transaction identifier and a "\$" to indicate this request contains the end of a complete message.

If Alice wants to deliver a very large message, she can split the message into chunks and deliver each chunk in a separate SEND request. The message ID corresponds to the whole message, so the receiver can also use it to reassemble the message and tell which chunks belong with which message. Chunking is described in more detail in Section 5.1. The Byte-Range header field identifies the portion of the message carried in this chunk and the total size of the message.

Alice can also specify what type of reporting she would like in response to her request. If Alice requests positive acknowledgments, Bob sends a REPORT request to Alice confirming the delivery of her complete message. This is especially useful if Alice sent a series of SEND request containing chunks of a single message. More on requesting types of reports and errors is described in Section 5.3.

Alice and Bob generally choose their MSRP URLs in such a way that is difficult to guess the exact URL. Alice and Bob can reject requests to URLs they are not expecting to service, and can correlate the specific URL with the probable sender. Alice and Bob can also use

TLS [1] to provide channel security over this hop. To receive MSRP requests over a TLS protected connection, Alice or Bob could advertise URLs with the "msrps" scheme instead of "msrp."

MSRP is designed with the expectation that MSRP can carry URLs for nodes on the far side of relays. For this reason, a URL with the "msrps" scheme makes no assertion about the security properties of other hops, just the next hop. The user agent knows the URL for each hop, so it can verify that each URL has the desired security properties.

MSRP URLs are discussed in more detail in Section 6.

An adjacent pair of busy MSRP nodes (for example two relays) can easily have several sessions, and exchange traffic for several simultaneous users. The nodes can use existing connections to carry new traffic with the same destination host, port, transport protocol, and scheme. MSRP nodes can keep track of how many sessions are using a particular connection and close these connections when no sessions have used them for some period of time. Connection management is discussed in more detail in Section 5.4.

5. Key Concepts

5.1. MSRP Framing and Message Chunking

Messages sent using MSRP can be very large and can be delivered in several SEND requests, where each SEND request contains one chunk of the overall message. Long chunks may be interrupted in mid-transmission to ensure fairness across shared transport connections. To support this, MSRP uses a boundary-based framing mechanism. The start line of an MSRP request contains a unique identifier that is also used to indicate the end of the request. Included at the end of the end-line, there is a flag that indicates whether this is the last chunk of data for this message or whether the message will be continued in a subsequent chunk. There is also a Byte-Range header field in the request that indicates that the overall position of this chunk inside the complete message.

For example, the following snippet of two SEND requests demonstrates a message that contains the text "abcdefgh" being sent as two chunks.

```
MSRP dkei38sd SEND
Message-ID: 456
Byte-Range: 1-4/8
Content-Type: text/plain
```

```
abcd
-----dkei38sd+
```

```
MSRP dkei38ia SEND
Message-ID: 456
Byte-Range: 5-8/8
Content-Type: text/plain
```

```
EFGH
-----dkei38ia$
```

This chunking mechanism allows a sender to interrupt a chunk part of the way through sending it. The ability to interrupt messages allows multiple sessions to share a TCP connection, and for large messages to be sent efficiently while not blocking other messages that share the same connection, or even the same MSRP session. Any chunk that is larger than 2048 octets MUST be interruptible. While MSRP would be simpler to implement if each MSRP session used its own TCP connection, there are compelling reasons to conserve connection. For example, the TCP peer may be a relay device that connects to many other peers. Such a device will scale better if each peer does not create a large number of connections.

The chunking mechanism only applies to the SEND method, as it is the only method used to transfer message content.

5.2. MSRP Addressing

MSRP entities are addressed using URLs. The MSRP URL schemes are defined in Section 6. The syntax of the To-Path and From-Path header fields each allow for a list of URLs. This was done to allow the protocol to work with relays, which are defined in a separate document, to provide a complete path to the end recipient. When two MSRP nodes communicate directly they need only one URL in the To-Path list and one URL in the From-Path list.

5.3. MSRP Transaction and Report Model

A sender sends MSRP requests to a receiver. The receiver MUST quickly accept or reject the request. If the receiver initially accepted the request, it still may then do things that take significant time to succeed or fail. For example, if the receiver is an MSRP to XMPP [29] gateway, it may forward the message over XMPP.

The XMPP side may later indicate that the request did not work. At this point, the MSRP receiver may need to indicate that the request did not succeed. There are two important concepts here: first, the hop by hop delivery of the request may succeed or fail; second, the end result of the request may be successfully processed or not. The first type of status is referred to as "transaction status" and may be returned in response to a request. The second type of status is referred to as "delivery status" and may be returned in a REPORT transaction.

The original sender of a request can indicate if they wish to receive reports for requests that fail, and can independently indicate if they wish to receive reports for requests that succeed. A receiver only sends a success REPORT if it knows that the request was successfully delivered, and the sender requested a success report. A receiver only sends a failure REPORT if the request failed to be delivered and the sender requested failure reports.

This document describes the behavior of MSRP endpoints. MSRP relays will introduce additional conditions that indicate a failure REPORT should be sent, such as the failure to receive a positive response from the next hop.

Two header fields control the sender's desire to receive reports. The header field "Success-Report" can have a value of "yes" or "no" and the "Failure-Report" header field can have a value of "yes", "no", or "partial".

The combinations of reporting are needed to meet the various scenarios of currently deployed IM systems. Success-Report might be "no" in many public systems to reduce load but might be "yes" in certain enterprise systems, such as systems used for securities trading. A Failure-Report value of "no" is useful for sending system messages such as "the system is going down in 5 minutes" without causing a response explosion to the sender. A Failure-Report of "yes" is used by many systems that wish to notify the user if the message failed. A Failure-Report of "partial" is a way to report errors other than timeouts. The timeout error reporting requires the sending hop to run a timer and the receiving hop to send an acknowledgment to stop the timer. Some systems don't want the overhead of doing this. "Partial" allows them to choose not to do so, but still allows error responses to be sent in many cases.

The term "partial" denotes the fact that the hop-by-hop acknowledgment mechanism that would be required if with a Failure-Report value of "yes" is not invoked. Thus, each device uses only "part" of the set of error detection tools available to them. This allows a compromise between no reporting of failures at all,

and reporting every possible failure. For example, with "partial", an sending device does not have to keep transaction state around waiting for a positive acknowledgment. But it still allows devices to report other types of errors. The receiving device could still report a policy violation such as an unacceptable content-type, or an ICMP error trying to connect to a downstream device.

5.4. MSRP Connection Model

When an MSRP endpoint wishes to send a request to a peer identified by an MSRP URL, it first needs a transport connection, with the appropriate security properties, to the host specified in the URL. If the sender already has such a connection, that is, one associated with the same host, port, and URL scheme, then it SHOULD reuse that connection.

When a new MSRP session is created, the initiating endpoint MUST act as the "active" endpoint, meaning that it is responsible for opening the transport connection to the answerer, if a new connection is required. However, this requirement MAY be weakened if standardized mechanisms for negotiating the connection direction become available, and is implemented by both parties to the connection.

Likewise, the active endpoint MUST immediately issue a SEND request. This initial SEND request MAY have a body if the sender has content to send, or it MAY have no body at all.

The first SEND request serves to bind a connection to an MSRP session from the perspective of the passive endpoint. If the connection is not authenticated with TLS, and the active endpoint did not send an immediate request, the passive endpoint would have no way to determine who had connected, and would not be able to safely send any requests towards the active party until after the active party sends its first request.

When an element needs to form a new connection, it looks at the URL to decide on the type of connection (TLS, TCP, etc.) then connects to the host indicated by the URL, following the URL resolution rules in Section 6.2. Connections using the "msrps" scheme MUST use TLS. The SubjectAltName in the received certificate MUST match the hostname part of the URL and the certificate MUST be valid, including having a date that is valid and being signed by an acceptable certificate authority. At this point the device that initiated the connection can assume that this connection is with the correct host.

The rules on certificate name matching and CA signing MAY be relaxed when using TLS peer-to-peer. In this case, a mechanism to ensure

that the peer used a correct certificate MUST be used. See Section 14.3 for details.

If the connection used mutual TLS authentication, and the TLS client presented a valid certificate, then the element accepting the connection can immediately know the identity of the connecting host. When mutual TLS authentication is not used, the listening device MUST wait until it receives a request on the connection, at which time it infers the identity of the connecting device from the associated session description.

When the first request arrives, its To-Path header field should contain a URL that the listening element provided in the SDP for a session. The element that accepted the connection looks up the URL in the received request, and determines which session it matches. If a match exists, the node MUST assume that the host that formed the connection is the host to which this URL was given. If no match exists, the node MUST reject the request with a 481 response. The node MUST also check to make sure the session is not already in use on another connection. If the session is already in use, it MUST reject the request with a 506 response.

If it were legal to have multiple connections associated with the same session, a security problem would exist. If the initial SEND request is not protected, an eavesdropper might learn the URL, and use it to insert messages into the session via a different connection.

If a connection fails for any reason, then an MSRP endpoint MUST consider any sessions associated with the connection as also having failed. When either endpoint notices such a failure, it MAY attempt to re-create any such sessions. If it chooses to do so, it MUST use a new SDP exchange, for example, in a SIP re-INVITE. If a replacement session is successfully created, endpoints MAY attempt to resend any content for which delivery on the original session could not be confirmed. If it does this, the Message-ID values for the resent messages MUST match those used in the initial attempts. If the receiving endpoint receives more than one message with the same Message-ID, it SHOULD assume that the messages are duplicates. The specific action that an endpoint takes when it receives a duplicate message is a matter of local policy, except that it SHOULD NOT present the duplicate messages to the user without warning of the duplication. Note that acknowledgments as needed based on the Failure-Report and Success-Report settings are still necessary even for requests containing duplicate content.

When endpoints create a new session in this fashion, the chunks for a given logical message MAY be split across the sessions. However,

endpoints SHOULD NOT split chunks between sessions under non-failure circumstances.

If an endpoint attempts to re-create a failed session in this manner, it MUST NOT assume that the MSRP URLs in the SDP will be the same as the old ones.

A connection SHOULD NOT be closed while there are sessions associated with it.

6. MSRP URLs

URLs using the "msrp" and "msrps" schema are used to identify a session of instant messages at a particular MSRP device. MSRP URLs are ephemeral; an MSRP device will generally use a different MSRP URL for each distinct session. An MSRP URL generally has no meaning outside of the associated session.

An MSRP URL follows a subset of the URL syntax in Appendix A of RFC3986 [10], with a scheme of "msrp" or "msrps". The syntax is described in Section 9.

The constructions for "userinfo", and "unreserved" are detailed in RFC3986 [10]. In order to allow IPV6 addressing, the construction for hostport is that used for SIP in RFC3261. URLs designating MSRP over TCP MUST include the "tcp" transport parameter.

Since this document only specifies MSRP over TCP, all MSRP URLs herein use the "tcp" transport parameter. Documents that provide bindings on other transports should define respective parameters for those transports.

An MSRP URL hostport field identifies a participant in a particular MSRP session. If the hostport contains a numeric IP address, it MUST also contain a port. The session-id part identifies a particular session of the participant. The absence of the session-id part indicates a reference to an MSRP host device, but does not specifically refer to a particular session.

A scheme of "msrps" indicates that the underlying connection MUST be protected with TLS.

MSRP has an IANA-registered recommended port defined in Section 15.4. This value is not a default, as the URL negotiation process described herein will always include explicit port numbers. However, the URLs SHOULD be configured so that the recommended port is used whenever appropriate. This makes life easier for network administrators who

need to manage firewall policy for MSRP.

The hostport will typically not contain a userinfo component, but MAY do so to indicate a user account for which the session is valid. Note that this is not the same thing as identifying the session itself. If a userinfo component exists, it MUST be constructed only from "unreserved" characters, to avoid a need for escape processing. Escaping MUST NOT be used in an MSRP URL. Furthermore, a userinfo part MUST NOT contain password information.

The limitation of userinfo to unreserved characters is an additional restriction to the userinfo definition in RFC3986. That version allows reserved characters. The additional restriction is to avoid the need for escaping.

The following is an example of a typical MSRP URL:

```
msrp://host.example.com:8493/asfd34;tcp
```

6.1. MSRP URL Comparison

MSRP URL comparisons MUST be performed according to the following rules:

1. The scheme MUST match. Scheme comparison is case insensitive.
2. If the hostpart contains an explicit IP address, and/or port, these are compared for address and port equivalence. Otherwise, hostpart is compared as a case insensitive character string.
3. If the port exists explicitly in either URL, then it MUST match exactly. A URL with an explicit port is never equivalent to another with no port specified.
4. The session-id part is compared as case sensitive. A URL without a session-id part is never equivalent to one that includes one.
5. URLs with different "transport" parameters never match. Two URLs that are identical except for transport are not equivalent. The transport parameter is case-insensitive.
6. Userinfo parts are not considered for URL comparison.

Path normalization is not relevant for MSRP URLs. Escape normalization is not required due to character restrictions in the formal syntax.

6.2. Resolving MSRP Host Device

An MSRP host device is identified by the hostport of an MSRP URL.

If the hostport contains a numeric IP address and port, they MUST be used as listed.

If the hostport contains a host name and a port, the connecting device MUST determine a host address by doing an A or AAAA DNS query, and use the port as listed.

If a connection attempt fails, the device SHOULD attempt to connect to the addresses returned in any additional A or AAAA records, in the order the records were presented.

This process assumes that the connection port is always known prior to resolution. This is always true for the MSRP URL uses described in this document, that is, URLs exchanged in the SDP offer and answer. The introduction of relays may create situations where this is not the case. For example, the MSRP URL that a user enters into a client to configure it to use a relay may be intended to be easily remembered and communicated by humans, and therefore is likely to omit the port. Therefore, the relay specification [22] may describe additional steps to resolve the port number.

MSRP devices MAY use other methods for discovering other such devices, when appropriate. For example, MSRP endpoints may use other mechanisms to discover relays, which are beyond the scope of this document.

7. Method-Specific Behavior

7.1. Constructing Requests

To form a new request, the sender creates a unique transaction identifier and uses this and the method name to create an MSRP request start line. Next, the sender places the target URL in a To-Path header field, and the sender's URL in a From-Path header field. If multiple URLs are present in the To-Path, the leftmost is the first URL visited; the rightmost URL is the last URL visited. The processing then becomes method specific. Additional method-specific header fields are added as described in the following sections.

After any method-specific header fields are added, processing continues to handle a body, if present. If the request has a body, it MUST contain a Content-Type header field. It may contain other

MIME-specific header fields. The Content-Type header field MUST be the last field in the message header section. The body MUST be separated from the header fields with an extra CRLF.

Non-SEND requests are not intended to carry message content, and are therefore not interruptible. Non-SEND request bodies MUST NOT be larger than 10240 octets.

Although this document does not discuss any particular usage of bodies in non-SEND requests, they may be useful in the future for carrying security or identity information, information about a message in progress, etc. The 10K size limit was chosen to be large enough for most of such applications, but small enough to avoid the fairness issues caused by sending arbitrarily large content in non-interruptible method bodies.

A request with no body MUST NOT include a Content-Type header field. Note that, if no body is present, no extra CRLF will be present between the header section and the end-line.

Requests with no bodies are useful when a client wishes to send "traffic", but does not wish to send content to be rendered to the peer user. For example, the active endpoint sends a SEND request immediately upon establishing a connection. If it has nothing to say at the moment, it can send a request with no body. Bodiless requests may also be used in certain applications to keep NAT bindings alive, etc.

Bodiless requests are distinct from requests with empty bodies. A request with an empty body will have a Content-Type header field value, and will generally be rendered to the recipient according to the rules for that type.

The end-line that terminates the request MUST be composed of seven "-" (minus sign) characters, the transaction ID as used in the start line, and a flag character. If a body is present, the end-line MUST be preceded by a CRLF that is not part of the body. If the chunk represents the data that forms the end of the complete message, the flag value MUST be a "\$". If the sender is aborting an incomplete message, and intends to send no further chunks in that message, it MUST be a "#". Otherwise it MUST be a "+".

If the request contains a body, the sender MUST ensure that the end-line (seven hyphens, the transaction identifier, and a continuation flag) is not present in the body. If the end-line is present in the body, the sender MUST choose a new transaction identifier that is not present in the body, and add a CRLF if needed, and the end-line, including the "\$", "#", or "+" character.

Some implementations may choose to scan for the closing sequence as they send the body, and if it is encountered, simply interrupt the chunk at that point and start a new transaction with a different transaction identifier to carry the rest of the body. Other implementation may choose to scan the data and ensure that the body does not contain the transaction identifier before they start sending the transaction.

Finally, requests which have no body MUST NOT contain a Content-Type header field or any other MIME-specific header field. Requests without bodies MUST contain a end-line after the final header field.

Once a request is ready for delivery, the sender follows the connection management (Section 5.4) rules to forward the request over an existing open connection or create a new connection.

7.1.1. Sending SEND Requests

When an endpoint has a message to deliver, it first generates a new Message-ID. This ID MUST be globally unique. If necessary, it breaks the message into chunks. It then generates a SEND request for each chunk, following the procedures for constructing requests (Section 7.1).

The Message-ID header field provides a globally unique message identifier that refers to a particular version of a particular message. The term "Message" in this context refers to a unit of content that the sender wishes to convey to the recipient. While such a message may be broken into chunks, the Message-ID refers to the entire message, not a chunk of the message.

The uniqueness of the message identifier is guaranteed by the host that generates it. This message identifier is intended to be machine readable and not necessarily meaningful to humans. A message identifier pertains to exactly one version of a particular message; subsequent revisions to the message each receive new message identifiers.

Each chunk of a message MUST contain a Message-ID header field containing the Message-ID. If the sender wishes non-default status reporting, it MUST insert a Failure-Report and/or Success-Report header field with an appropriate value. All chunks of the same message MUST use the same Failure-Report and Success-Report values in their SEND requests.

If success reports are requested, i.e. the value of the Success-Report header field is "yes", the sending device MAY wish to run a timer of some value that makes sense for its application and take action if a success report is not received in this time. There is no

universal value for this timer. For many IM applications, it may be 2 minutes while for some trading systems it may be under a second. Regardless of whether such a timer is used, if the success report has not been received by the time the session is ended, the device SHOULD inform the user.

If the value of "Failure-Report" is set to "yes", then the sender of the request runs a timer. If a 200 response to the transaction is not received within 30 seconds from the time the last byte of the transaction is sent, or submitted to the operating system for sending, the element MUST inform the user that the request probably failed. If the value is set to "partial", then the element sending the transaction does not have to run a timer, but MUST inform the user if it receives a non-recoverable error response to the transaction.

The treatment of timers for success reports and failure reports is intentionally inconsistent. An explicit timeout value makes sense for failure reports since such reports will usually refer to a message "chunk" which is acknowledged on a hop-by-hop basis. This is not the case for success reports, which are end-to-end and may refer to the entire message content, which can be arbitrarily large.

If no Success-Report header field is present in a SEND request, it MUST be treated the same as a Success-Report header field with value of "no". If no Failure-Report header field is present, it MUST be treated the same as a Failure-Report header field with value of "yes". If an MSRP endpoint receives a REPORT for a Message-ID it does not recognize, it SHOULD silently ignore the REPORT.

The Byte-Range header field value contains a starting value (range-start) followed by a "-", an ending value (range-end) followed by a "/", and finally the total length. The first octet in the message has a position of one, rather than a zero.

The first chunk of the message SHOULD, and all subsequent chunks MUST include a Byte-Range header field. The range-start field MUST indicate the position of the first byte in the body in the overall message (for the first chunk this field will have a value of one). The range-end field SHOULD indicate the position of the last byte in the body, if known. It MUST take the value of "*" if the position is unknown, or if the request needs to be interruptible. The total field SHOULD contain the total size of the message, if known. The total field MAY contain a "*" if the total size of the message is not known in advance. The sender MUST send all chunks in Byte-Range order. (However, the receiver cannot assume that the requests will be delivered in order, as intervening relays may have changed the

order.)

There are some circumstances where an endpoint may choose to send an empty SEND request. For the sake of consistency, a Byte-Range header field referring to nonexistent or zero-length content MUST still have a range-start value of 1. For example, "1-0/0"

To ensure fairness over a connection, senders MUST NOT send chunks with a body larger than 2048 octets unless they are prepared to interrupt them (meaning that any chunk with a body of greater than 2048 octets will have a "*" character in the range-end field). A sender can use one of the following two strategies to satisfy this requirement. The sender is STRONGLY RECOMMENDED to send messages larger than 2048 octets using as few chunks as possible, interrupting chunks (at least 2048 octets long) only when other traffic is waiting to use the same connection. Alternatively, the sender MAY simply send chunks in 2048 octet increments until the final chunk. Note that the former strategy results in markedly more efficient use of the connection. All MSRP nodes MUST be able to receive chunks of any size from zero octets to the maximum number of octets they can receive for a complete message. Senders SHOULD NOT break messages into chunks smaller than 2048 octets, except for the final chunk of a complete message.

A SEND request is interrupted while a body is in the process of being written to the connection by simply noting how much of the message has already been written to the connection, then writing out the end-line to end the chunk. It can then be resumed in a another chunk with the same Message-ID and a Byte-Range header field range start field containing the position of the first byte after the interruption occurred.

SEND requests larger than 2048 octets MUST be interrupted if the sender needs to send pending responses or REPORT requests. If multiple SEND requests from different sessions are concurrently being sent over the same connection, the device SHOULD implement some scheme to alternate between them such that each concurrent request gets a chance to send some fair portion of data at regular intervals suitable to the application.

The sender MUST NOT assume that a message is received by the peer with the same chunk allocation with which it was sent. An intervening relay could possibly break SEND requests into smaller chunks, or aggregate multiple chunks into larger ones.

The default disposition of bodies is "render". If the sender wants a different disposition, it MAY insert a Content-Disposition[9] header field. Since MSRP can carry unencoded binary payloads, transfer

encoding is always "binary", and transfer-encoding parameters MUST NOT be present.

7.1.2. Sending REPORT Requests

REPORT requests are similar to SEND requests, except that report requests MUST NOT include Success-Report or Failure-Report header fields, and MUST contain a Status header field. REPORT requests MUST contain the Message-ID header field from the original SEND request.

If an MSRP element receives a REPORT for a Message-ID it does not recognize, it SHOULD silently ignore the REPORT.

An MSRP endpoint MUST be able to generate success REPORT requests.

REPORT requests will normally not include a body, as the REPORT request header fields can carry sufficient information in most cases. However, REPORT requests MAY include a body containing additional information about the status of the associated SEND request. Such a body is informational only, and the sender of the REPORT request SHOULD NOT assume that the recipient pays any attention to the body. REPORT requests are not interruptible.

Success-Report and Failure-Report header fields MUST NOT be present in REPORT requests. MSRP nodes MUST NOT send REPORT requests in response to REPORT requests. MSRP Nodes MUST NOT send MSRP responses to REPORT requests.

7.1.3. Generating Success Reports

When an endpoint receives a message in one or more chunks that contain a Success-Reports value of "true", it MUST send a success report or reports covering all bytes that are received successfully. The success reports are sent in the form of REPORT requests, following the normal procedures (Section 7.1), with a few additional requirements.

The receiver MAY wait until it receives the last chunk of a message, and send a success report that covers the complete message. Alternately, it MAY generate incremental success REPORTs as the chunks are received. These can be sent periodically and cover all the bytes that have been received so far, or they can be sent after a chunk arrives and cover just the part from that chunk.

It is helpful to think of a success REPORT as reporting on a particular range of bytes, rather than on a particular chunk sent by a client. The sending client cannot depend on the Byte-Range header field in a given success report matching that of a

particular SEND request. For example, an intervening MSRP relay may break chunks into smaller chunks, or aggregate multiple chunks into larger ones.

A side effect of this is, even if no relay is used, the receiving client may report on byte ranges that do not exactly match those in the original chunks sent by the sender. It can wait until all bytes in a message are received and report on the whole, it can report as it receives each chunk, or it can report on any other received range.

Reporting on ranges smaller than the entire message contents allows certain improved user experiences for the sender. For example, a sending client could display incremental status information showing which ranges of bytes have been acknowledged by the receiver.

However, the choice on whether to report incrementally is entirely up to the receiving client. There is no mechanism for the sender to assert its desire to receive incremental reports or not. Since the presence of a relay can cause the receiver to see a very different chunk allocation than the sender, such a mechanism would be of questionable value.

When generating a REPORT request, the endpoint inserts a To-Path header field containing the From-Path value from the original request, and a From-Path header field containing the URL identifying itself in the session. The endpoint then inserts a Status header field with a namespace of "000", a status-code of "200" and an implementation-defined comment phrase. It also inserts a Message-ID header field containing the value from the original request.

The namespace field denotes the context of the status-code field. The namespace value of "000" means the status-code should be interpreted in the same way as the matching MSRP transaction response code. If a future specification uses the status-code field for some other purpose, it MUST define a new namespace field value.

The endpoint MUST NOT send a success report for a SEND request that either contained no Success-Report header field, or contained such a field with a value of "no". That is, if no Success-Report header field is present, it is treated identically to one with a value of "no."

7.1.4. Generating Failure Reports

If an MSRP endpoint receives a SEND request that it cannot process for some reason, and the Failure-Report header field either was not present in the original request, or had a value of "yes", it SHOULD simply include the appropriate error code in the transaction

response. However, there may be situations where the error cannot be determined quickly, such as when the endpoint is a gateway that waits for a downstream network to indicate an error. In this situation, it MAY send a 200 OK response to the request, and then send a failure REPORT request when the error is detected.

If the endpoint receives a SEND request with a Failure-Report header field value of "no", then it MUST NOT send a failure REPORT request, and MUST NOT send a transaction response. If the value is "partial", it MUST NOT send a 200 transaction response to the request, but SHOULD send an appropriate non-200 class response if a failure occurs.

As stated above, if no Failure-Report header field is present, it MUST be treated the same as a Failure-Report header field with value of "yes".

Construction of failure REPORT requests is identical to that for success REPORT requests, except the Status header field code and reason fields MUST contain appropriate error codes. Any error response code defined in this specification MAY also be used in failure reports.

If a failure REPORT request is sent in response to a SEND request that contained a chunk, it MUST include a Byte-Range header field indicating the actual range being reported on. It can take the range-start and total values from the original SEND request, but MUST calculate the range-end field from the actual body data.

Endpoints SHOULD NOT send REPORT requests if they have reason to believe the request will not be delivered. For example, they SHOULD NOT send a REPORT request on a session that is no longer valid.

This section only describes failure report generation behavior for MSRP endpoints. Relay behavior is beyond the scope of this document, and will be considered in a separate document [22]. We expect failure reports to be more commonly generated by relays than by endpoints.

7.2. Constructing Responses

If an MSRP endpoint receives a request that either contains a Failure-Report header field value of "yes", or does not contain a Failure-Report header field at all, it MUST immediately generate a response. Likewise, if an MSRP endpoint receives a request that contains a Failure-Report header field value of "partial", and the receiver is unable to process the request, it SHOULD immediately generate a response.

To construct the response, the endpoint first creates the response start-line, inserting appropriate response code and reason fields. The transaction identifier in the response start line MUST match the transaction identifier from the original request.

The endpoint then inserts an appropriate To-Path header field. If the request triggering the response was a SEND request, the To-Path header field is formed by copying the last (right-most) URL in the From-Path header field of the request. (Responses to SEND requests are returned only to the previous hop.) For responses to all other request methods, the To-Path header field contains the full path back to the original sender. This full path is generated by taking the list of URLs from the From-Path of the original request, reversing the list, and writing the reversed list into the To-Path of the response. (Legal REPORT requests do not request responses, so this specification doesn't exercise the behavior described above, however we expect that extensions for gateways and relays will need such behavior.)

Finally, the endpoint inserts a From-Path header field containing the URL that identifies it in the context of the session, followed by the end-line after the last header field. The response MUST be transmitted back on the same connection on which the original request arrived.

7.3. Receiving Requests

The receiving endpoint MUST first check the URL in the To-Path to make sure the request belongs to an existing session. When the request is received, the To-Path will have exactly one URL, which MUST map to an existing session that is associated with the connection on which the request arrived. If this is not true, then the receiver MUST generate a 481 error and ignore the request. Note that if the Failure-Report header field had a value of "no", then no error report would be sent.

Further request processing by the receiver is method specific.

7.3.1. Receiving SEND Requests

When the receiving endpoint receives a SEND request, it first determines if it contains a complete message, or a chunk from a larger message. If the request contains no Byte-Range header field, or contains one with a range-start value of "1", and the closing line continuation flag has a value of "\$", then the request contained the entire message. Otherwise, the receiver looks at the Message-ID value to associate chunks together into the original message. It forms a virtual buffer to receive the message, keeping track of which

bytes have been received and which are missing. The receiver takes the data from the request and places it in the appropriate place in the buffer. The receiver SHOULD determine the actual length of each chunk by inspecting the payload itself; it is possible the body is shorter than the range-end field indicates. This can occur if the sender interrupted a SEND request unexpectedly. It is worth noting that the chunk that has a termination character of "\$" defines the total length of the message.

It is technically illegal for the sender to prematurely interrupt a request that had anything other than "*" in the last-byte position of the Byte-Range header field. But having the receiver calculate a chunk length based on actual content adds resilience in the face of sender errors. Since this should never happen with compliant senders, this only has a SHOULD strength.

Receivers MUST not assume that the chunks will be delivered in order or that they will receive all the chunks with "+" flags before they receive the chunk with the "\$" flag. In certain cases of connection failure, it is possible for information to be duplicated. If chunk data is received that overlaps already received data for the same message, the last chunk received SHOULD take precedence (even though this may not have been the last chunk transmitted). For example, if bytes 1 to 100 were received and a chunk arrives that contains bytes 50 to 150, this second chunk will overwrite bytes 50 to 100 of the data that had already been received. Although other schemes work, this is the easiest for the receiver and results in consistent behavior between clients.

There are situations in which the receiver may not be able to give precedence to the last chunk received when chunks overlap. For example, the recipient might incrementally render chunks as they arrive. If a new chunk arrives that overlaps with a previously rendered chunk, it would be too late to "take back" any conflicting data from the first chunk. Therefore, the requirement to give precedence to the most recent chunk is specified at a "SHOULD" strength. This requirement is not intended to disallow applications where this behavior does not make sense.

The seven "-" in the end-line are used so that the receiver can search for the value "----", 32 bits at a time to find the probable location of the end-line. This allows most processors to locate the boundaries and copy the memory at the same rate that a normal memory copy could be done. This approach results in a system that is as fast as framing based on specifying the body length in the header fields of the request, but also allows for the interruption of messages.

What is done with the body is outside the scope of MSRP and largely determined by the MIME Content-Type and Content-Disposition. The body MAY be rendered after the whole message is received or partially rendered as it is being received.

If the SEND request contained a Content-Type header field indicating an unsupported MIME type, and the Failure-Report value is not "no", the receiver MUST generate a response with a status code of 415. All MSRP endpoints MUST be able to receive the multipart/mixed [15] and multipart/alternative [15] MIME types.

If the Success-Report header field was set to "yes", the receiver must construct and send one or more success reports, as described in Section 7.1.3.

7.3.2. Receiving REPORT Requests

When an endpoint receives a REPORT request, it correlates it to the original SEND request using the Message-ID and the Byte-Range, if present. If it requested success reports, then it SHOULD keep enough state about each outstanding sent message so that it can correlate REPORT requests to the original messages.

An endpoint that receives a REPORT request containing a Status header field with a namespace field of "000" MUST interpret the report in exactly the same way it would interpret an MSRP transaction response with a response code matching the status-code field.

It is possible to receive a failure report or a failure transaction response for a chunk that is currently being delivered. In this case, the entire message corresponding to that chunk SHOULD be aborted, by including the "#" character in the continuation field of the end-line.

It is possible that an endpoint will receive a REPORT request on a session that is no longer valid. The endpoint's behavior if this happens is a matter of local policy. The endpoint is not required to take any steps to facilitate such late delivery, i.e. it is not expected to keep a connection active in case late REPORTs might arrive.

When an endpoint that sent a SEND request receives a failure REPORT indicating that a particular byte range was not received, it MUST treat the session as failed. If it wishes to recover, it MUST first re-negotiate the URLs at the signaling level then resend that range of bytes of the message on the resulting new session.

MSRP nodes MUST NOT send MSRP REPORT requests in response to other

REPORT requests.

8. Using MSRP with SIP and SDP

MSRP sessions will typically be initiated using the Session Description Protocol (SDP) [2] via the SIP offer/answer mechanism [3].

This document defines a handful of new SDP parameters to set up MSRP sessions. These are detailed below and in the IANA Considerations section.

An MSRP media-line (that is, a media-line proposing MSRP) in the session description is accompanied by a mandatory "path" attribute. This attribute contains a space-separated list of URLs to be visited to contact the user agent advertising this session-description. If more than one URL is present, the leftmost URL is the first URL to be visited to reach the target resource. (The path list can contain multiple URLs to allow for the deployment of gateways or relays in the future.) MSRP implementations that can accept incoming connections without the need for relays will typically only provide a single URL here.

An MSRP media line is also accompanied by an "accept-types" attribute, and optionally an "accept-wrapped-types" attribute. These attributes are used to specify the MIME types that are acceptable to the endpoint.

8.1. SDP Connection and Media Lines

An SDP connection-line takes the following format:

```
c=<network type> <address type> <connection address>
```

The network type and address type fields are used as normal for SDP. The connection address field MUST be set to the IP address or fully qualified domain name from the MSRP URL identifying the endpoint in its path attribute.

The general format of an SDP media-line is:

```
m=<media> <port> <protocol> <format list>
```

An offered or accepted media-line for MSRP over TCP MUST include a protocol field value of "TCP/MSRP", or "TCP/TLS/MSRP" for TLS. The media field value MUST be "message". The format list field MUST be set to "*".

The port field value MUST match the port value used in the endpoint's MSRP URL in the path attribute, except that, as described in [3], a user agent that wishes to accept an offer, but not a specific media-line, MUST set the port number of that media-line to zero (0) in the response. Since MSRP allows multiple sessions to share the same TCP connection, multiple m-lines in a single SDP document may share the same port field value; MSRP devices MUST NOT assume any particular relationship between m-lines on the sole basis that they have matching port field values.

MSRP devices do not use the c-line address field, or the m-line port and format list fields to determine where to connect. Rather, they use the attributes defined in this specification. The connection information is copied to the c-line and m-line for purposes of backwards compatibility with conventional SDP usages. While MSRP could theoretically carry any media type, "message" is appropriate.

8.2. URL Negotiations

Each endpoint in an MSRP session is identified by a URL. These URLs are negotiated in the SDP exchange. Each SDP offer or answer that proposes MSRP MUST contain a path attribute containing one or more MSRP URLs. The path attribute is used in an SDP a-line, and has the following syntax:

```
path = path-label ":" path-list
path-label = "path"
path-list= MSRP-URL *(SP MSRP-URL)
```

where MSRP-URL is an "msrp" or "msrps" URL as defined in Section 6. MSRP URLs included in an SDP offer or answer MUST include explicit port numbers.

An MSRP device uses the URL to determine a host address, port, transport, and protection level when connecting, and to identify the target when sending requests and responses.

The offerer and answerer each selects a URL to represent itself and sends it to the peer device in the SDP document. Each device stores the path value received from the peer and uses that value as the target for requests inside the resulting session. If the path attribute received from the peer contains more than one URL, then the target URL is the rightmost, while the leftmost entry represents the adjacent hop. If only one entry is present, then it is both the peer and adjacent hop URL. The target path is the entire path attribute value received from the peer.

The following example shows an SDP offer with a session URL of "msrp://alice.example.com:7394/2s93i;tcp"

```
v=0
o=alice 2890844526 2890844527 IN IP4 alice.example.com
s= -
c=IN IP4 alice.example.com
t=0 0
m=message 7394 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://alice.example.com:7394/2s93i;tcp
```

The rightmost URL in the path attribute MUST identify the endpoint that generated the SDP document, or some other location where that endpoint wishes to receive requests associated with the session. It MUST be assigned for this particular session, and MUST NOT duplicate any URL in use for any other session in which the endpoint is currently participating. It SHOULD be hard to guess, and protected from eavesdroppers. This is discussed in more detail in Section 14.

8.3. Path Attributes with Multiple URLs

As mentioned previously, this document describes MSRP for peer-to-peer scenarios, that is, when no relays are used. The use of relays are described in a separate document [22]. In order to allow an MSRP device that only implements the core specification to interoperate with devices that use relays, this document must include a few assumptions about how relays work.

An endpoint that uses one or more relays will indicate that by putting a URL for each device in the relay chain into the SDP path attribute. The final entry will point to the endpoint itself. The other entries will indicate each proposed relay, in order. The first entry will point to the first relay in the chain from the perspective of the peer; that is, the relay to which the peer device, or a relay operating on its behalf, should connect.

Endpoints that do not wish to insert a relay, including those that do not support relays at all, will put exactly one URL into the path attribute. This URL represents both the endpoint for the session, and the connection point.

Even though endpoints that implement only this specification will never introduce a relay, they need to be able to interoperate with other endpoints that do use relays. Therefore, they MUST be prepared to receive more than one URL in the SDP path attribute. When an endpoint receives more than one URL in a path attribute, only the first entry is relevant for purposes of resolving the address and

port, and establishing the network connection, as it describes the first adjacent hop.

If an endpoint puts more than one URL in a path attribute, the final URL in the path attribute (the peer URL) identifies the session, and MUST not duplicate the URL of any other session in which the endpoint is currently participating. Uniqueness requirements for other entries in the path attribute are out of scope for this document.

8.4. Updated SDP Offers

MSRP endpoints may sometimes need to send additional SDP exchanges for an existing session. They may need to send periodic exchanges with no change to refresh state in the network, for example, SIP session timers or the SIP UPDATE[23] request. They may need to change some other stream in a session without affecting the MSRP stream, or they may need to change an MSRP stream without affecting some other stream.

Either peer may initiate an updated exchange at any time. The endpoint that sends the new offer assumes the role of offerer for all purposes. The answerer MUST respond with a path attribute that represents a valid path to itself at the time of the updated exchange. This new path may be the same as its previous path, but may be different. The new offerer MUST NOT assume that the peer will answer with the same path it used previously.

If either party wishes to send an SDP document that changes nothing at all, then it MUST have the same o-line as in the previous exchange.

8.5. Connection Negotiation

Previous versions of this document included a mechanism to negotiate the direction for any required TCP connection. The mechanism was loosely based on the COMEDIA [25] work being done in the MMUSIC working group. The primary motivation was to allow MSRP sessions to succeed in situations where the offerer could not accept connections but the answerer could. For example, the offerer might be behind a NAT, while the answerer might have a globally routable address.

The SIMPLE working group chose to remove that mechanism from MSRP, as it added a great deal of complexity to connection management. Instead, MSRP now specifies a default connection direction. The party that sent the original offer is responsible for connecting to its peer.

8.6. Content Type Negotiation

An SDP media-line proposing MSRP MUST be accompanied by an accept-types attribute.

An entry of "*" in the accept-types attribute indicates that the sender may attempt to send content with media types that have not been explicitly listed. Likewise, an entry with an explicit type and a "*" character as the subtype indicates that the sender may attempt to send content with any subtype of that type. If the receiver receives an MSRP request and is able to process the media type, it does so. If not, it will respond with a 415 response. Note that all explicit entries SHOULD be considered preferred over any non-listed types. This feature is needed as, otherwise, the list of formats for rich IM devices may be prohibitively large.

This specification requires the support of certain data formats. Mandatory formats MUST be signaled like any other, either explicitly or by the use of a "*".

The accept-types attribute may include container types, that is, MIME formats that contain other types internally. If compound types are used, the types listed in the accept-types attribute may be used both as the root payload, or may be wrapped in a listed container type. Any container types MUST also be listed in the accept-types attribute.

Occasionally an endpoint will need to specify a MIME body type that can only be used if wrapped inside a listed container type.

Endpoints MAY specify MIME types that are only allowed when wrapped inside compound types using the "accept-wrapped-types" attribute in an SDP a-line.

The semantics for accept-wrapped-types are identical to those of the accept-types attribute, with the exception that the specified types may only be used when wrapped inside container types listed in accept-types attribute. Only types listed in the accept-types attribute may be used as the "root" type for the entire body. Since any type listed in accept-types may be used both as a root body, and wrapped in other bodies, format entries from accept-types SHOULD NOT be repeated in this attribute.

This approach does not allow for specifying distinct lists of acceptable wrapped types for different types of containers. If an endpoint understands a MIME type in the context of one wrapper, it is assumed to understand it in the context of any other acceptable wrappers, subject to any constraints defined by the wrapper types

themselves.

The approach of specifying types that are only allowed inside of containers separately from the primary payload types allows an endpoint to force the use of certain wrappers. For example, a CPIM [12] gateway device may require all messages to be wrapped inside message/cpim bodies, but may allow several content types inside the wrapper. If the gateway were to specify the wrapped types in the accept-types attribute, its peer might attempt to use those types without the wrapper.

If the recipient of an offer does not understand any of the payload types indicated in the offered SDP, it SHOULD indicate that using the appropriate mechanism of the rendezvous protocol. For example, in SIP, it SHOULD return a SIP 488 response.

An endpoint MAY indicate the maximum size message they wish to receive using the max-size a-line attribute. Max-size refers to the complete message in octets, not the size of any one chunk. Senders SHOULD NOT exceed the max-size limit for any message sent in the resulting session. However, the receiver should consider max-size value as a hint.

The formal syntax for these attributes are as follows:

```
accept-types = accept-types-label ":" format-list
accept-types-label = "accept-types"
accept-wrapped-types = wrapped-types-label ":" format-list
wrapped-types-label = "accept-wrapped-types"
format-list = format-entry *( SP format-entry)
format-entry = (type "/" subtype) / (type "/" "*" ) / ("*")
type = token
subtype = token

max-size = max-size-label ":" max-size-value
max-size-label = "max-size"
max-size-value = 1*(DIGIT) ;max size in octets
```

8.7. Example SDP Exchange

Endpoint A wishes to invite Endpoint B to an MSRP session. A offers the following session description:

```
v=0
o=usera 2890844526 2890844527 IN IP4 alice.example.com
s= -
c=IN IP4 alice.example.com
t=0 0
m=message 7394 TCP/MSRP *
a=accept-types: message/cpim text/plain text/html
a=path:msrp://alice.example.com:7394/2s93i9;tcp
```

B responds with its own URL:

```
v=0
o=userb 2890844530 2890844532 IN IP4 bob.example.com
s= -
c=IN IP4 bob.example.com
t=0 0
m=message 8493 TCP/MSRP *
a=accept-types:message/cpim text/plain
a=path:msrp://bob.example.com:8493/si438ds;tcp
```

8.8. MSRP User Experience with SIP

In typical SIP applications, when an endpoint receives an INVITE request, it alerts the user, and waits for user input before responding. This is analogous to the typical telephone user experience, where the callee "answers" the call.

In contrast, the typical user experience for instant messaging applications is that the initial received message is immediately displayed to the user, without waiting for the user to "join" the conversation. Therefore, the principle of least surprise would suggest that MSRP endpoints using SIP signaling SHOULD allow a mode where the endpoint quietly accepts the session, and begins displaying messages.

This guideline may not make sense for all situations, such as for mixed media applications, where both MSRP and audio sessions are offered in the same INVITE. In general, good application design should take precedence.

SIP INVITE requests may be forked by a SIP proxy, resulting in more than one endpoint receiving the same INVITE. SIP early media [28] techniques can be used to establish a preliminary session with each endpoint so the initial message(s) are displayed on each endpoint, and canceling the INVITE transaction for any endpoints that do not send MSRP traffic after some period of time, so that they cease receiving MSRP traffic from the inviter.

8.9. SDP direction attribute and MSRP

SDP defines a number of attributes that modify the direction of media flows. These are the "sendonly", "recvonly", "inactive", and "sendrecv" attributes.

If a "sendonly" or "recvonly" attribute modifies an MSRP media description line, the attribute indicates the direction of MSRP SEND requests that contain regular message payloads. Unless otherwise specified, these attributes do not affect the direction of other types of requests, such as REPORT. SEND requests that contain some kind of control or reporting protocol rather than regular message payload (e.g., IMDN reports) should be generated according to the protocol rules as if no direction attribute were present.

9. Formal Syntax

MSRP is a text protocol that uses the UTF-8 [14] transformation format.

The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in RFC-2234 [6].

```
msrp-req-or-resp = msrp-request / msrp-response
msrp-request   = req-start headers [content-stuff] end-line
msrp-response  = resp-start headers end-line

req-start     = pMSRP SP transact-id SP method CRLF
resp-start    = pMSRP SP transact-id SP status-code [SP comment] CRLF
comment       = utf8text

pMSRP         = %x4D.53.52.50 ; MSRP in caps
transact-id   = ident
method        = mSEND / mREPORT / other-method
mSEND         = %x53.45.4e.44 ; SEND in caps
mREPORT       = %x52.45.50.4f.52.54; REPORT in caps

other-method  = 1*UPALPHA
status-code   = 3DIGIT ; any code defined in this document
                ; or an extension document

MSRP-URL      = msrp-scheme "://" [userinfo "@"] hostport
                [ "/" session-id ] ";" transport *( ";" url-parameter )
                ; userinfo as defined in RFC3986, except
                ; limited to unreserved.
                ; hostport as defined in RFC3261
```

```
msrp-scheme = "msrp" / "msrps"
session-id = 1*( unreserved / "+" / "=" / "/" )
              ; unreserved as defined in RFC3986
transport = "tcp" / ALPHANUM
url-parameter = token ["=" token]

headers = To-Path CRLF From-Path CRLF 1*( header CRLF )

header = Message-ID
        / Success-Report
        / Failure-Report
        / Byte-Range
        / Status
        / ext-header

To-Path = "To-Path:" SP MSRP-URL *( SP MSRP-URL )
From-Path = "From-Path:" SP MSRP-URL *( SP MSRP-URL )
Message-ID = "Message-ID:" SP ident
Success-Report = "Success-Report:" SP ("yes" / "no" )
Failure-Report = "Failure-Report:" SP ("yes" / "no" / "partial" )
Byte-Range = "Byte-Range:" SP range-start "-" range-end "/" total
range-start = 1*DIGIT
range-end   = 1*DIGIT / "*"
total       = 1*DIGIT / "*"

Status = "Status:" SP namespace SP status-code [SP text-reason]
namespace = 3(DIGIT); "000" for all codes defined in this document.
text-reason = utf8text

ident = alphanum 3*31ident-char
ident-char = alphanum / "." / "-" / "+" / "%" / "="

content-stuff = *(Other-Mime-header CRLF)
              Content-Type 2CRLF data CRLF

Content-Type = "Content-Type:" SP media-type
media-type = type "/" subtype *( ";" gen-param )
type = token
subtype = token

gen-param = pname [ "=" pval ]
pname = token
pval = token / quoted-string

token = 1*(%x21 / %x23-27 / %x2A-2B / %x2D-2E
          / %x30-39 / %x41-5A / %x5E-7E)
        ; token is compared case-insensitive
```

```

quoted-string = DQUOTE *(qdttext / qd-esc) DQUOTE
qdttext = SP / HTAB / %x21 / %x23-5B / %x5D-7E
          / UTF8-NONASCII
qd-esc = (BACKSLASH BACKSLASH) / (BACKSLASH DQUOTE)
BACKSLASH = "\"
UPALPHA = %x41-5A
ALPHANUM = ALPHA / DIGIT

Other-Mime-header = (Content-ID
  / Content-Description
  / Content-Disposition
  / mime-extension-field);

      ; Content-ID, and Content-Description are defined in RFC2045.
      ; Content-Disposition is defined in RFC2183
      ; MIME-extension-field indicates additional MIME extension
      ; header fields as described in RFC2045

data = *OCTET
end-line = "-----" transact-id continuation-flag CRLF
continuation-flag = "+" / "$" / "#"

ext-header = hname ":" SP hval CRLF
hname = ALPHA *token
hval = utf8text

utf8text = *(HTAB / %x20-7E / UTF8-NONASCII)

UTF8-NONASCII = %xC0-DF 1UTF8-CONT
               / %xE0-EF 2UTF8-CONT
               / %xF0-F7 3UTF8-CONT
               / %xF8-Fb 4UTF8-CONT
               / %xFC-FD 5UTF8-CONT
UTF8-CONT     = %x80-BF

```

10. Response Code Descriptions

This section summarizes the semantics of various response codes that may be used in MSRP transaction responses. These codes may also be used in the Status header field in REPORT requests.

10.1. 200

The 200 response code indicates a successful transaction.

10.2. 400

A 400 response indicates a request was unintelligible. The sender may retry the request after correcting the error.

10.3. 403

A 403 response indicates the attempted action is not allowed. The sender should not try the request again.

10.4. 408

A 408 response indicates that a downstream transaction did not complete in the allotted time. It is never sent by any elements described in this specification. However, 408 is used in the MSRP Relay extension; therefore MSRP endpoints may receive it. An endpoint **MUST** treat a 408 response in the same manner as it would treat a local timeout.

10.5. 413

A 413 response indicates that the receiver wishes the sender to stop sending the particular message. Typically, a 413 is sent in response to a chunk of an undesired message.

If a message sender receives a 413 in a response, or in a REPORT request, it **MUST NOT** send any further chunks in the message, that is, any further chunks with the same Message-ID value. If the sender receives the 413 while in the process of sending a chunk, and the chunk is interruptible, the sender **MUST** interrupt it.

10.6. 415

A 415 response indicates the SEND request contained a MIME content-type that is not understood by the receiver. The sender should not send any further messages with the same content-type for the duration of the session.

10.7. 423

A 423 response indicates that one of the requested parameters is out of bounds. It is used by the relay extensions to this document.

10.8. 481

A 481 response indicates that the indicated session does not exist. The sender should terminate the session.

10.9. 501

A 501 response indicates that the recipient does not understand the request method.

The 501 response code exists to allow some degree of method extensibility. It is not intended as a license to ignore methods defined in this document; rather it is a mechanism to report lack of support of extension methods.

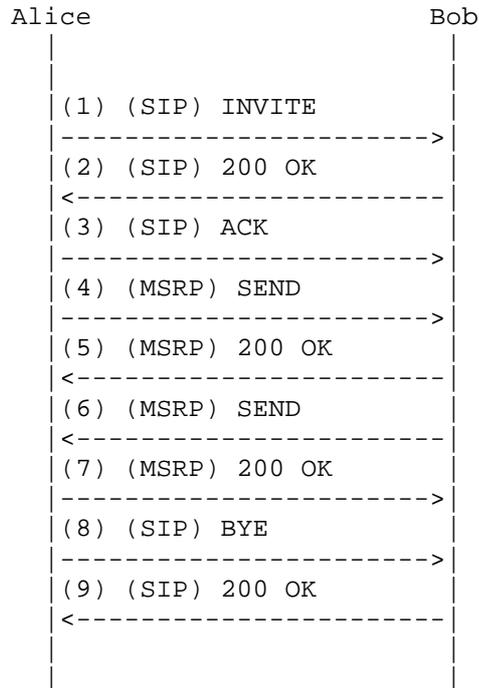
10.10. 506

A 506 response indicates that a request arrived on a session which is already bound to another network connection. The sender should cease sending messages for that session on this connection.

11. Examples

11.1. Basic IM Session

This section shows an example flow for the most common scenario. The example assumes SIP is used to transport the SDP exchange. Details of the SIP messages and SIP proxy infrastructure are omitted for the sake of brevity. In the example, assume that the offerer is sip:alice@example.com and the answerer is sip:bob@example.com.



1. Alice constructs a local URL of
msrp://alicepc.example.com:7777/iau39;tcp .

Alice->Bob (SIP): INVITE sip:bob@example.com

```
v=0
o=alice 2890844557 2890844559 IN IP4 alicepc.example.com
s= -
c=IN IP4 alicepc.example.com
t=0 0
m=message 7777 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://alicepc.example.com:7777/iau39;tcp
```

2. Bob listens on port 8888, and sends the following response:

Bob->Alice (SIP): 200 OK

```
v=0
o=bob 2890844612 2890844616 IN IP4 bob.example.com
s= -
c=IN IP4 bob.example.com
t=0 0
m=message 8888 TCP/MSRP *
```

- a=accept-types:text/plain
a=path:msrp://bob.example.com:8888/9di4ea;tcp
3. Alice->Bob (SIP): ACK sip:bob@example.com
 4. (Alice opens connection to Bob.) Alice->Bob (MSRP):

MSRP d93kswow SEND
To-Path: msrp://bob.example.com:8888/9di4ea;tcp
From-Path: msrp://alicepc.example.com:7777/iau39;tcp
Message-ID: 12339sdqwer
Content-Type: text/plain

Hi, I'm Alice!
-----d93kswow\$
 5. Bob->Alice (MSRP):

MSRP d93kswow 200 OK
To-Path: msrp://alicepc.example.com:7777/iau39;tcp
From-Path: msrp://bob.example.com:8888/9di4ea;tcp
-----d93kswow\$
 6. Bob->Alice (MSRP):

MSRP dkei38sd SEND
To-Path: msrp://alicepc.example.com:7777/iau39;tcp
From-Path: msrp://bob.example.com:8888/9di4ea;tcp
Message-ID: 456
Content-Type: text/plain

Hi, Alice! I'm Bob!
-----dkei38sd\$
 7. Alice->Bob (MSRP):

MSRP dkei38sd 200 OK
To-Path: msrp://alicepc.example.com:7777/iau39;tcp
From-Path: msrp://bob.example.com:8888/9di4ea;tcp
-----dkei38sd\$
 8. Alice->Bob (SIP): BYE sip:bob@example.com

Alice invalidates local session state.
 9. Bob invalidates local state for the session.

Bob->Alice (SIP): 200 OK

11.2. Message with XHTML Content

```
MSRP dsdfoe38sd SEND
To-Path: msrp://alice.example.com:7777/iau39;tcp
From-Path: msrp://bob.example.com:8888/9di4ea;tcp
Message-ID: 456
Content-Type: application/xhtml+xml

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>FY2005 Results</title>
  </head>
  <body>
    <p>See the results at <a
href="http://example.org/">example.org</a>.</p>
  </body>
</html>
-----dsdfoe38sd$
```

11.3. Chunked Message

For an example of a chunked message, see the example in Section 5.1.

11.4. System Message

```
Sysadmin->Alice (MSRP):

MSRP d93kswow SEND
To-Path: msrp://alicepc.example.com:8888/9di4ea;tcp
From-Path: msrp://example.com:7777/iau39;tcp
Message-ID: 12339sdqwer
Failure-Report: no
Success-Report: no
Content-Type: text/plain

This conference will end in 5 minutes
-----d93kswow$
```

11.5. Positive Report

Alice->Bob (MSRP):

```
MSRP d93kswow SEND
To-Path: msrp://bob.example.com:8888/9di4ea;tcp
From-Path: msrp://alicepc.example.com:7777/iau39;tcp
Message-ID: 12339sdqwer
Byte-Range: 1-106/106
Success-Report: yes
Failure-Report: no
Content-Type: text/html
```

```
<html><body>
<p>Here is that important link...
<a href="http://www.example.com/foobar">foobar</a>
</p>
</body></html>
-----d93kswow$
```

Bob->Alice (MSRP):

```
MSRP dkei38sd REPORT
To-Path: msrp://alicepc.example.com:7777/iau39;tcp
From-Path: msrp://bob.example.com:8888/9di4ea;tcp
Message-ID: 12339sdqwer
Byte-Range: 1-106/106
Status: 000 200 OK
-----dkei38sd$
```

11.6. Forked IM

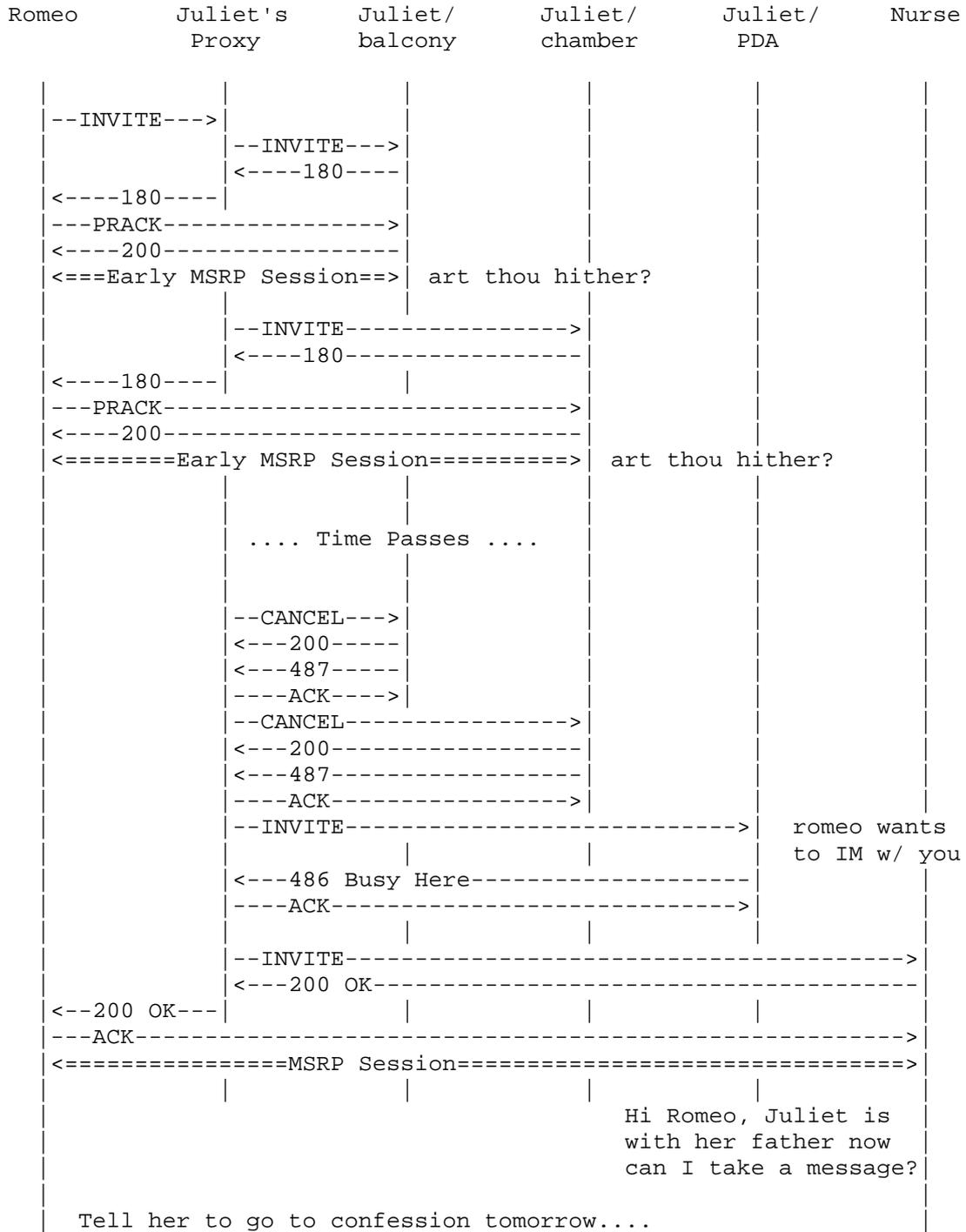
Traditional IM systems generally do a poor job of handling multiple simultaneous IM clients online for the same person. While some do a better job than many existing systems, handling of multiple clients is fairly crude. This becomes a much more significant issue when always-on mobile devices are available, but it is desirable to use them only if another IM client is not available.

Using SIP makes rendezvous decisions explicit, deterministic, and very flexible. In contrast, "page-mode" IM systems use implicit implementation-specific decisions which IM clients cannot influence. With SIP session-mode messaging, rendezvous decisions can be under control of the client in a predictable, interoperable way for any host that implements callee capabilities [30]. As a result, rendezvous policy is managed consistently for each address of record.

The following example shows Juliet with several IM clients where she

can be reached. Each of these has a unique SIP Contact and MSRP session. The example takes advantage of SIP's capability to "fork" an invitation to several Contacts in parallel, in sequence, or in combination. Juliet has registered from her chamber, the balcony, her PDA, and as a last resort, you can leave a message with her Nurse. Juliet's contacts are listed below. The q-values express relative preference (q=1.0 is the highest preference).

When Romeo opens his IM program, he selects Juliet and types the message "art thou hither?" (instead of "you there?"). His client sends a SIP invitation to sip:juliet@thecapulets.example.com. The proxy there tries first the balcony and the chamber simultaneously. A client is running on each of those systems, both of which set up early sessions of MSRP with Romeo's client. The client automatically sends the message over MSRP to the two MSRP URIs involved. After a delay of a several seconds with no reply or activity from Juliet, the proxy cancels the invitation at her first two contacts, and forwards the invitation on to Juliet's PDA. Since her father is talking to her about her wedding, she selects "Do Not Disturb" on her PDA, which sends a "Busy Here" response. The proxy then tries the Nurse, who answers and tells Romeo what is going on.



12. Extensibility

MSRP was designed to be only minimally extensible. New MSRP Methods, header fields, and status codes can be defined in standards track RFCs. MSRP does not contain a version number or any negotiation mechanism to require or discover new features. If an extension is specified in the future that requires negotiation, the specification will need to describe how the extension is to be negotiated in the encapsulating signaling protocol. If a non-interoperable update or extension occurs in the future, it will be treated as a new protocol, and MUST describe how its use will be signaled.

In order to allow extension header fields without breaking interoperability, if an MSRP device receives a request or response containing a header field that it does not understand, it MUST ignore the header field and process the request or response as if the header field was not present. If an MSRP device receives a request with an unknown method, it MUST return a 501 response.

MSRP was designed to use lists of URLs instead of a single URL in the To-Path and From-Path header fields in anticipation of relay or gateway functionality being added. In addition, "msrp" and "msrps" URLs can contain parameters that are extensible.

13. CPIM Compatibility

MSRP sessions may go to a gateway to other CPIM [26] compatible protocols. If this occurs, the gateway MUST maintain session state, and MUST translate between the MSRP session semantics and CPIM semantics, which do not include a concept of sessions. Furthermore, when one endpoint of the session is a CPIM gateway, instant messages SHOULD be wrapped in "message/cpim" [12] bodies. Such a gateway MUST include "message/cpim" as the first entry in its SDP accept-types attribute. MSRP endpoints sending instant messages to a peer that has included "message/cpim" as the first entry in the accept-types attribute SHOULD encapsulate all instant message bodies in "message/cpim" wrappers. All MSRP endpoints MUST support the message/cpim type, and SHOULD support the S/MIME[7] features of that format.

If a message is to be wrapped in a message/cpim envelope, the wrapping MUST be done prior to breaking the message into chunks, if needed.

All MSRP endpoints MUST recognize the From, To, DateTime, and Require header fields as defined in RFC3862. Such applications SHOULD recognize the CC header field, and MAY recognize the Subject header field. Any MSRP application that recognizes any message/cpim header

field MUST understand the NS (name space) header field.

All message/cpim body parts sent by an MSRP endpoint MUST include the From and To header fields. If the message/cpim body part is protected using S/MIME, then it MUST also include the DateTime header field.

The NS, To, and CC header fields may occur multiple times. Other header fields defined in RFC3862 MUST NOT occur more than once in a given message/cpim body part in an MSRP message. The Require header field MAY include multiple values. The NS header field MAY occur zero or more times, depending on how many name spaces are being referenced.

Extension header fields MAY occur more than once, depending on the definition of such header fields.

Using message/cpim envelopes is also useful if an MSRP device wishes to send a message on behalf of some other identity. The device may add a message/cpim envelope with the appropriate From header field value.

14. Security Considerations

Instant Messaging systems are used to exchange a variety of sensitive information ranging from personal conversations, to corporate confidential information, to account numbers and other financial trading information. IM is used by individuals, corporations, and governments for communicating important information. IM systems need to provide the properties of integrity and confidentiality for the exchanged information, the knowledge that you are communicating with the correct party, and allow the possibility of anonymous communication. MSRP pushes many of the hard problems to SIP when SIP sets up the session, but some of the problems remain. Spam and DoS attacks are also very relevant to IM systems.

MSRP needs to provide confidentiality and integrity for the messages it transfers. It also needs to provide assurances that the connected host is the host that it meant to connect to and that the connection has not been hijacked.

14.1. Transport Level Protection

When using only TCP connections, MSRP security is fairly weak. If host A is contacting host B, B passes its hostname and a secret to A using a rendezvous protocol. Although MSRP requires the use of a rendezvous protocol with the ability to protect this exchange, there

is no guarantee that the protection will be used all the time. If such protection is not used, anyone can see this secret. Host A then connects to the provided host name and passes the secret in the clear across the connection to B. Host A assumes that it is talking to B based on where it sent the SYN packet and then delivers the secret in plain text across the connections. Host B assumes it is talking to A because the host on the other end of the connection delivered the secret. An attacker that could ACK the SYN packet could insert itself as a man in the middle in the connection.

When using TLS connections, the security is significantly improved. We assume that the host accepting the connection has a certificate from a well-known certificate authority. Furthermore, we assume that the signaling to set up the session is protected by the rendezvous protocol. In this case, when host A contacts host B, the secret is passed through a confidential channel to A. A connects with TLS to B. B presents a valid certificate, so A knows it really is connected to B. A then delivers the secret provided by B, so that B can verify it is connected to A. In this case, a rogue SIP Proxy can see the secret in the SIP signaling traffic and could potentially insert itself as a man-in-the-middle.

Realistically, using TLS with certificates from well known certificate authorities is difficult for peer-to-peer connections, as the types of hosts that end clients use for sending instant messages are unlikely to have long-term stable IP addresses or DNS names that certificates can bind to. In addition, the cost of server certificates from well-known certificate authorities is currently expensive enough to discourage their use for each client. Using TLS in a peer-to-peer mode without well known certificate is discussed in Section 14.3.

TLS becomes much more practical when some form of relay is introduced. Clients can then form TLS connections to relays, which are much more likely to have TLS certificates. While this specification does not address such relays, they are described by a companion document [22]. That document makes extensive use of TLS to protect traffic between clients and relays, and between one relay and another.

TLS is used to authenticate devices and to provide integrity and confidentiality for the header fields being transported. MSRP elements MUST implement TLS and MUST also implement the TLS ClientExtendedHello extended hello information for server name indication as described in [11]. A TLS cipher-suite of TLS_RSA_WITH_AES_128_CBC_SHA [13] MUST be supported (other cipher-suites MAY also be supported).

14.2. S/MIME

The only strong security for non-TLS connections is achieved using S/MIME.

Since MSRP carries arbitrary MIME content, it can trivially carry S/MIME protected messages as well. All MSRP implementations MUST support the multipart/signed MIME type even if they do not support S/MIME. Since SIP can carry a session key, S/MIME messages in the context of a session could also be protected using a key-wrapped shared secret [27] provided in the session setup. MSRP can carry unencoded binary payloads. Therefore MIME bodies MUST be transferred with a transfer encoding of binary. If a message is both signed and encrypted, it SHOULD be signed first, then encrypted. If S/MIME is supported, SHA-1, RSA, and AES-128 MUST be supported.

This does not actually require the endpoint to have certificates from a well-known certificate authority. When MSRP is used with SIP, the Identity [16] and Certificates [24] mechanisms provide S/MIME based delivery of a secret between A and B. No SIP intermediary except the explicitly trusted authentication service (one per user) can see the secret. The S/MIME encryption of the SDP can also be used by SIP to exchange keying material that can be used in MSRP. The MSRP session can then use S/MIME with this keying material to encrypt and sign messages sent over MSRP. The connection can still be hijacked since the secret is sent in clear text to the other end of the TCP connection, but the consequences are mitigated if all the MSRP content is encrypted and signed with S/MIME. Although out of scope for this document, the SIP negotiation of MSRP session can negotiate symmetric keying material to be used with S/MIME for integrity and privacy.

14.3. Using TLS in Peer to Peer Mode

TLS can be used with a self-signed certificate as long as there is a mechanism for both sides to ascertain that the other side used the correct certificate. When used with SDP and SIP, the correct certificate can be verified by passing a fingerprint of the certificate in the SDP and ensuring that the SDP has suitable integrity protection. When SIP is used to transport the SDP, the integrity can be provided by the SIP Identity mechanism[16]. The rest of this section describes the details of this approach.

If self-signed certificates are used, the content of the subjectAltName attribute inside the certificate MAY use the uniform resource identifier (URI) of the user. In SIP, this URI of the user is the User's Address of Record (AOR). This is useful for debugging purposes only and is not required to bind the certificate to one of

the communication endpoints. Unlike normal TLS operations in this protocol, when doing peer-to-peer TLS, the `subjectAltName` is not an important component of the certificate verification. If the endpoint is also able to make anonymous sessions, a distinct, unique certificate MUST be used for this purpose. For a client that works with multiple users, each user SHOULD have its own certificate. Because the generation of public/private key pairs is relatively expensive, endpoints are not required to generate certificates for each session.

A certificate fingerprint is the output of a one-way hash function computed over the distinguished encoding rules (DER) form of the certificate. The endpoint MUST use the certificate fingerprint attribute as specified in [17] and MUST include this in the SDP. The certificate presented during the TLS handshake needs to match the fingerprint exchanged via the SDP and if the fingerprint does not match the hashed certificate then the endpoint MUST tear down the media session immediately.

When using SIP, the integrity of the fingerprint can be ensured through the SIP Identity mechanism [16]. When a client wishes to use SIP to set up a secure MSRP session with another endpoint it sends an SDP offer in a SIP message to the other endpoint. This offer includes, as part of the SDP payload, the fingerprint of the certificate that the endpoint wants to use. The SIP message containing the offer is sent to the offerer's SIP proxy which will add an Identity header according to the procedures outlined in [16]. When the far endpoint receives the SIP message it can verify the identity of the sender using the Identity header. Since the Identity header is a digital signature across several SIP headers, in addition to the body or bodies of the SIP message, the receiver can also be certain that the message has not been tampered with after the digital signature was added to the SIP message.

An example of SDP with a fingerprint attribute is shown in the following figure. Note the fingerprint is shown spread over two lines due to formatting consideration but should all be on one line.

```
c=IN IP4 atlanta.example.com
m=message 7654 TCP/TLS/MSRP *
a=accept-types:text/plain
a=path:msrps://atlanta.example.com:7654/jshA7we;tcp
a=fingerprint:SHA-1 \
    4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
```

14.4. Other Security Concerns

MSRP cannot be used as an amplifier for DoS attacks, but it can be used to form a distributed attack to consume TCP connection resources on servers. The attacker, Mallory, sends a SIP INVITE with no offer to Alice. Alice returns a 200 with an offer and Mallory returns an answer with SDP indicating that his MSRP address is the address of Tom. Since Alice sent the offer, Alice will initiate a connection to Tom using up resources on Tom's server. Given the huge number of IM clients, and the relatively few TCP connections that most servers support, this is a fairly straightforward attack.

SIP is attempting to address issues in dealing with spam. The spam issue is probably best dealt with at the SIP level when an MSRP session is initiated and not at the MSRP level.

If a sender chooses to employ S/MIME to protect a message, all S/MIME operations apply to the complete message, prior to any breaking of the message into chunks.

The signaling will have set up the session to or from some specific URLs that will often have "im:" or "sip:" URI schemes. When the signaling has been set up to a specific end user, and S/MIME is implemented, then the client needs to verify that the name in the SubjectAltName of the certificate contains an entry that matches the URI that was used for the other end in the signaling. There are some cases, such as IM conferencing, where the S/MIME certificate name and the signaled identity will not match. In these cases, the client should ensure that the user is informed that the message came from the user identified in the certificate and does not assume that the message came from the party they signaled.

In some cases, a sending device may need to attribute a message to some other identity, and may use different identities for different messages in the same session. For example, a conference server may send messages on behalf of multiple users on the same session. Rather than add additional header fields to MSRP for this purpose, MSRP relies on the message/cpim format for this purpose. The sender may envelop such a message in a message/cpim body, and place the actual sender identity in the From field. The trustworthiness of such an attribution is affected by the security properties of the session in the same way that the trustworthiness of the identity of the actual peer is affected, with the additional issue of determining whether the recipient trusts the sender to assert the identity.

This approach can result in nesting of message/cpim envelopes. For example, a message originates from a CPIM gateway, and is then forwarded by a conference server onto a new session. Both the

gateway and the conference server introduce envelopes. In this case, the recipient client SHOULD indicate the chain of identity assertions to the user, rather than allow the user to assume that either the gateway or the conference server originated the message.

It is possible that a recipient might receive messages that are attributed to the same sender via different MSRP sessions. For example, Alice might be in a conversation with Bob via an MSRP session over a TLS protected channel. Alice might then receive a different message from Bob over a different session, perhaps with a conference server that asserts Bob's identity in a message/cpim envelope signed by the server.

MSRP does not prohibit multiple simultaneous sessions between the same pair of identities. Nor does it prohibit an endpoint sending a message on behalf of another identity, such as may be the case for a conference server. The recipient's endpoint should determine its level of trust of the authenticity of the sender independently for each session. The fact that an endpoint trusts the authenticity of the sender on any given session should not affect the level of trust it assigns for apparently the same sender on a different session.

When MSRP clients form or acquire a certificate, they SHOULD ensure that the subjectAltName has a GeneralName entry of type uniformResourceIdentifier for each URL corresponding to this client and should always include an "im:" URI. It is fine if the certificate contains other URIs such as "sip:" or "xmpp:" URIs.

MSRP implementors should be aware of a potential attack on MSRP devices that involves placing very large values in the byte-range header field, potentially causing the device to allocate very large memory buffers to hold the message. Implementations SHOULD apply some degree of sanity checking on byte-range values before allocating such buffers.

15. IANA Considerations

This specification instructs IANA to create a new registry for MSRP parameters. The MSRP Parameter registry is a container for sub-registries. This section further introduces sub-registries for MSRP method names, status codes, and header field names.

Additionally, Section 15.4 through Section 15.7 register new parameters in existing IANA registries.

[NOTE TO IANA/RFC Editor: Please replace all occurrences of RFCXXXX in this section with the actual number assigned to this document.]

15.1. MSRP Method Names

This specification establishes the Method sub-registry under MSRP Parameters and initiates its population as follows. New parameters in this sub-registry must be published in an RFC (either as an IETF submission or RFC Editor submission).

SEND - [RFCXXXX]
REPORT - [RFCXXXX]

The following information MUST be provided in an RFC publication in order to register a new MSRP Method:

The method name.
The RFC number in which the method is registered.

15.2. MSRP Header Fields

This specification establishes the header field-Field sub-registry under MSRP Parameters. New parameters in this sub-registry must be published in an RFC (either as an IETF submission or RFC Editor submission). Its initial population is defined as follows:

To-Path - [RFCXXXX]
From-Path - [RFCXXXX]
Success-Report - [RFCXXXX]
Failure-Report - [RFCXXXX]
Byte-Range - [RFCXXXX]
Status - [RFCXXXX]

The following information MUST be provided in an RFC publication in order to register a new MSRP header field:

The header field name.
The RFC number in which the method is registered.

15.3. MSRP Status Codes

This specification establishes the Status-Code sub-registry under MSRP Parameters. New parameters in this sub-registry must be published in an RFC (either as an IETF submission or RFC Editor submission). Its initial population is defined in Section 10. It takes the following format:

Code [RFC Number]

The following information MUST be provided in an RFC publication in order to register a new MSRP status code:

The status code number.

The RFC number in which the method is registered.

15.4. MSRP Port

MSRP uses TCP port XYZ. Usage of this value is described in Section 6.

[NOTE TO IANA/RFC Editor: Please replace XYZ in this section with the assigned port number.]

15.5. MSRP URL Schemes

This document defines the URL schemes of "msrp" and "msrps".

Syntax: See Section 6.

Character Encoding: See Section 6.

Intended Usage: See Section 6.

Protocols: The Message Session Relay Protocol (MSRP).

Security Considerations: See Section 14.

Relevant Publications: RFCXXXX

15.6. SDP Transport Protocol

MSRP defines the a new SDP protocol field values "TCP/MSRP" and "TCP/TLS/MSRP", which should be registered in the sdp-parameters registry under "proto". This first value indicates the MSRP protocol when TCP is used as an underlying transport. The second indicates that TLS is used.

Specifications defining new protocol values must define the rules for the associated media format namespace. The "TCP/MSRP" and "TCP/TLS/MSRP" protocol values allow only one value in the format field (fmt), which is a single occurrence of "*". Actual format determination is made using the "accept-types" and "accept-wrapped-types" attributes.

15.7. SDP Attribute Names

This document registers the following SDP attribute parameter names in the sdp-parameters registry. These names are to be used in the SDP att-name field.

15.7.1. Accept Types

Contact Information: Ben Campbell (ben@estacado.net)
Attribute-name: accept-types
Long-form Attribute Name: Acceptable MIME Types
Type: Media level
Subject to Charset Attribute: No
Purpose and Appropriate Values: The "accept-types" attribute contains a list of MIME content-types that the endpoint is willing to receive. It may contain zero or more registered MIME types, or "*" in a space delimited string.

15.7.2. Wrapped Types

Contact Information: Ben Campbell (ben@estacado.net)
Attribute-name: accept-wrapped-types
Long-form Attribute Name: Acceptable MIME Types Inside Wrappers
Type: Media level
Subject to Charset Attribute: No
Purpose and Appropriate Values: The "accept-wrapped-types" attribute contains a list of MIME content-types that the endpoint is willing to receive in an MSRP message with multipart content, but may not be used as the outermost type of the message. It may contain zero or more registered MIME types, or "*" in a space delimited string.

15.7.3. Max Size

Contact Information: Ben Campbell (ben@estacado.net)
Attribute-name: max-size
Long-form Attribute Name: Maximum message size.
Type: Media level
Subject to Charset Attribute: No
Purpose and Appropriate Values: The "max-size" attribute indicates the largest message an endpoint wishes to accept. It may take any numeric value, specified in octets.

15.7.4. Path

Contact Information: Ben Campbell (ben@estacado.net)
Attribute-name: path
Long-form Attribute Name: MSRP URL Path
Type: Media level
Subject to Charset Attribute: No
Purpose and Appropriate Values: The "path" attribute indicates a series of MSRP devices that must be visited by messages sent in the session, including the final endpoint. The attribute contains one or more MSRP URIs, delimited by the space character.

16. Contributors and Acknowledgments

In addition to the editors, the following people contributed extensive work to this document: Chris Boulton, Paul Kyzivat, Orit Levin, Hans Persson, Adam Roach, Jonathan Rosenberg, and Robert Sparks.

The following people contributed substantial discussion and feedback to this ongoing effort: Eric Burger, Allison Mankin, Jon Peterson, Brian Rosen, Dean Willis, Aki Niemi, Hisham Khartabil, Pekka Pessi, Miguel Garcia, Peter Ridler, Sam Hartman, and Jean Mahoney.

17. References

17.1. Normative References

- [1] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.
- [2] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", draft-ietf-mmusic-sdp-new-26 (work in progress), July 2006.
- [3] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [4] 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.
- [5] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [6] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.
- [7] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004.
- [8] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.
- [9] Troost, R., Dorner, S., and K. Moore, "Communicating Presentation Information in Internet Messages: The Content-Disposition header field", RFC 2183, August 1997.

- [10] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 3986, January 2005.
- [11] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 3546, June 2003.
- [12] Klyne, G. and D. Atkins, "Common Presence and Instant Messaging (CPIM): Message Format", RFC 3862, August 2004.
- [13] Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, June 2002.
- [14] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, November 2003.
- [15] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", rfc 2046, November 1996.
- [16] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", draft-ietf-sip-identity-06 (work in progress), October 2005.
- [17] Lennox, J., "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", draft-ietf-mmusic-comedia-tls-06 (work in progress), March 2006.

17.2. Informational References

- [18] Johnston, A. and O. Levin, "Session Initiation Protocol Call Control - Conferencing for User Agents", draft-ietf-sipping-cc-conferencing-07 (work in progress), June 2005.
- [19] Rosenberg, J., Peterson, J., Schulzrinne, H., and G. Camarillo, "Best Current Practices for Third Party Call Control in the Session Initiation Protocol", RFC 3725, April 2004.
- [20] Sparks, R., Johnston, A., and D. Petrie, "Session Initiation Protocol Call Control - Transfer", draft-ietf-sipping-cc-transfer-06 (work in progress), March 2006.
- [21] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for

- Instant Messaging", RFC 3428, December 2002.
- [22] Jennings, C., Mahy, R., and A. Roach, "Relay Extensions for Message Sessions Relay Protocol (MSRP)", draft-ietf-simple-msrp-relays-07 (work in progress), February 2006.
 - [23] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.
 - [24] Jennings, C. and J. Peterson, "Certificate Management Service for SIP", draft-ietf-sipping-certs-03 (work in progress), March 2006.
 - [25] Yon, D. and G. Camarillo, "Connection-Oriented Media Transport in SDP", rfc 4145, September 2005.
 - [26] Peterson, J., "A Common Profile for Instant Messaging (CPIM)", rfc 3860, August 2004.
 - [27] Housley, R., "Triple-DES and RC2 Key Wrapping", RFC 3217, December 2001.
 - [28] Camarillo, G. and H. Schulzrinne, "Early Media and Ringing Tone Generation in the Session Initiation Protocol (SIP)", rfc 3960, December 2004.
 - [29] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 3921, October 2004.
 - [30] Rosenberg, J., "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.
 - [31] Peterson, J., "Address Resolution for Instant Messaging and Presence", rfc 3861, August 2004.

Authors' Addresses

Ben Campbell (editor)
Estacado Systems
17210 Campbell Road
Suite 250
Dallas, TX 75252
USA

Email: ben@estacado.net

Rohan Mahy (editor)
Plantronics
345 Encinacal Street
Santa Cruz, CA
USA

Email: rohan@ekabal.com

Cullen Jennings (editor)
Cisco Systems, Inc.
170 West Tasman Dr.
MS: SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM 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.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

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

