                The WebSocket Protocol as a Transport
              for the Message Session Relay Protocol (MSRP)

Abstract

   The WebSocket protocol enables two-way real-time communication
   between clients and servers in situations where direct access to TCP
   and UDP is not available (for example, from within JavaScript in a
   web browser). This document specifies a new WebSocket subprotocol as
   a reliable transport mechanism between Message Session Relay Protocol
   (MSRP) clients and relays to enable usage of MSRP in new scenarios.
   This document normatively updates RFCs 4975 and 4976.

Status of This Memo

Table of Contents

1. Introduction

   The WebSocket [RFC6455] protocol enables message exchange between
   clients and servers on top of a persistent TCP connection (optionally
   secured with Transport Layer Security (TLS) [RFC5246]). The initial
   protocol handshake makes use of HTTP [RFC7230] semantics, allowing
   the WebSocket protocol to reuse existing HTTP infrastructure.

   Modern web browsers include a WebSocket client stack complying with
   the WebSocket API [WS-API] as specified by the W3C. It is expected
   that other client applications (those running in personal computers
   and devices such as smartphones) will also make a WebSocket client
   stack available. The specification in this document enables usage of
   Message Session Relay Protocol [RFC4975] in these scenarios.

   This specification defines a new WebSocket subprotocol (as defined in
   Section 1.9 in [RFC6455]) for transporting MSRP messages between a
   WebSocket client and MSRP relay [RFC4976] containing a WebSocket
   server, a new transport for MSRP, and procedures for MSRP clients and
   relays implementing the WebSocket transport.

   MSRP over WebSocket is well suited for MSRP interactions between
   clients and servers. Common use cases for MSRP over WebSocket
   include:

   o Human-to-machine messaging

   o Client-to-server data exchange (for example, application control
      signaling)

   o Human-to-human messaging where local policy requires
      authentication and/or logging

   MSRP Connection Establishment for Media Anchoring (MSRP-CEMA)
   [RFC6714] is outside of the scope of this document, as this document
   is intended to describe connecting to a WebSocket server that is an
   MSRP relay.

2. Terminology

   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 [RFC2119].

2.1. Definitions

   MSRP WebSocket Client: An MSRP entity capable of opening outbound
       connections to MSRP relays that are WebSocket servers and
       communicating using the WebSocket MSRP subprotocol as defined
       by this document.

   MSRP WebSocket Server: An MSRP entity (specifically an MSRP relay
       [RFC4976]) capable of listening for inbound connections from
       WebSocket clients and communicating using the WebSocket MSRP
       subprotocol as defined by this document.

3. WebSocket Protocol Overview

   The WebSocket protocol [RFC6455] is a transport layer on top of TCP
   (optionally secured with TLS [RFC5246]) in which both the client and
   server exchange message units in both directions. The protocol
   defines a connection handshake, WebSocket subprotocol and extensions
   negotiation, a frame format for sending application and control data,
   a masking mechanism, and status codes for indicating disconnection
   causes.

   The WebSocket connection handshake is based on HTTP [RFC7230] and
   utilizes the HTTP GET method with an "Upgrade" request. This is sent
   by the client and then answered by the server (if the negotiation
   succeeded) with an HTTP 101 status code. Once the handshake is
   completed, the connection upgrades from HTTP to the WebSocket
   protocol. This handshake procedure is designed to reuse the existing
   HTTP infrastructure. During the connection handshake, client and
   server agree on the application protocol to use on top of the
   WebSocket transport. Such application protocol (also known as a
   "WebSocket subprotocol") defines the format and semantics of the
   messages exchanged by the endpoints. This could be a custom protocol
   or a standardized one (such as the WebSocket MSRP subprotocol defined
   in this document). Once the HTTP 101 response is processed, both
   client and server reuse the underlying TCP connection for sending
   WebSocket messages and control frames to each other. Unlike plain
   HTTP, this connection is persistent and can be used for multiple
   message exchanges.

   WebSocket defines message units to be used by applications for the
   exchange of data, so it provides a message boundary-preserving
   transport layer. These message units can contain either UTF-8 text
   or binary data and can be split into multiple WebSocket text/binary
   transport frames as needed by the WebSocket stack.

   The WebSocket API [WS-API] for web browsers only defines callbacks to
   be invoked upon receipt of an entire message unit regardless of
   whether it was received in a single WebSocket frame or split across
   multiple frames.

4. The WebSocket MSRP Subprotocol

   The term WebSocket subprotocol refers to an application-level
   protocol layered on top of a WebSocket connection. This document
   specifies the WebSocket MSRP subprotocol for carrying MSRP requests
   and responses through a WebSocket connection.

4.1. Handshake

   The MSRP WebSocket Client and MSRP WebSocket Server negotiate usage
   of the WebSocket MSRP subprotocol during the WebSocket handshake
   procedure as defined in Section 1.3 of [RFC6455]. The Client MUST
   include the value "msrp" in the Sec-WebSocket-Protocol header in its
   handshake request. The 101 reply from the Server MUST contain "msrp"
   in its corresponding Sec-WebSocket-Protocol header.

   Below is an example of a WebSocket handshake in which the Client
   requests the WebSocket MSRP subprotocol support from the Server:

     GET / HTTP/1.1
     Host: a.example.com
     Upgrade: websocket
     Connection: Upgrade
     Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
     Origin: http://www.example.com
     Sec-WebSocket-Protocol: msrp
     Sec-WebSocket-Version: 13

   The handshake response from the Server accepting the WebSocket MSRP
   subprotocol would look as follows:

     HTTP/1.1 101 Switching Protocols
     Upgrade: websocket
     Connection: Upgrade
     Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
     Sec-WebSocket-Protocol: msrp

   Once the negotiation has been completed, the WebSocket connection is
   established and can be used for the transport of MSRP requests and
   responses. The WebSocket messages transmitted over this connection
   MUST conform to the negotiated WebSocket subprotocol.

4.2. MSRP Encoding

   WebSocket messages can be transported in either UTF-8 text frames or
   binary frames. MSRP [RFC4975] allows both text and binary bodies in
   MSRP requests. Therefore, MSRP WebSocket Clients and Servers MUST
   accept both text and binary frames.

   The WebSocket API [WS-API] does not allow developers to choose
   whether to send UTF-8 text or binary frames but will not send
   non-UTF-8 characters in a text frame. The content of text frames
   MUST be interpreted as binary by WebSocket Clients and Servers.

5. MSRP WebSocket Transport

5.1. General

   WebSocket clients cannot receive WebSocket connections initiated by
   other WebSocket clients or WebSocket servers. This means that it is
   challenging for an MSRP client to communicate directly with other
   MSRP clients. Therefore, all MSRP-over-WebSocket messages MUST be
   routed via an MSRP WebSocket Server. MSRP traffic transported over
   WebSockets MUST be protected by using a Secure WebSocket (WSS)
   connection (using TLS [RFC5246] over TCP).

   MSRP WebSocket Servers can be used to route MSRP messages between
   MSRP WebSocket Clients and between MSRP WebSocket Clients and
   "normal" MSRP clients and relays.

   Each MSRP chunk MUST be carried within a single WebSocket message,
   and a WebSocket message MUST NOT contain more than one MSRP chunk.

   This simplifies parsing of MSRP messages for both clients and
   servers. When large messages are sent by a non-WebSocket peer, MSRP
   chunking (as defined in Section 5.1 of [RFC4975]) MUST be used by the
   WebSocket MSRP Servers to split the message into several smaller MSRP
   chunks.

5.2. Updates to RFC 4975

5.2.1. MSRP URI Transport Parameter

   This document defines the value "ws" as the transport parameter value
   for an MSRP URI [RFC3986] to be contacted using the MSRP WebSocket
   subprotocol as transport.

   The updated ABNF [RFC5234] for this parameter is the following (the
   original BNF for this parameter can be found in [RFC4975]):

     transport = "tcp" / "ws" / 1*ALPHANUM

5.2.2. SDP Transport Protocol

   This document does not define a new Session Description Protocol
   (SDP) transport protocol for MSRP over WebSockets. As all MSRP-over-
   WebSocket messages MUST be routed via an MSRP WebSocket Server, the
   MSRP WebSocket Client MUST specify "TCP/TLS/MSRP" protocols in the
   SDP m-line -- that being the protocol used by non-WebSocket clients
   and between MSRP relays (see Section 8.1 of [RFC4975]).

   The "ws" transport parameter will appear in the endpoint URI in the
   SDP "path" attribute (see Section 8.2 of [RFC4975]). MSRP was
   designed with the possibility of new transport bindings in mind (see
   Section 6 of [RFC4975]), so MSRP implementations are expected to
   allow unrecognized transports, provided that they do not have to
   establish a direct connection to the resource described by the URI.

5.3. Updates to RFC 4976

5.3.1. AUTH Request Authentication

   The MSRP relay specification [RFC4976] states that AUTH requests MUST
   be authenticated. This document modifies this requirement to state
   that all connections between MSRP clients and relays MUST be
   authenticated. In the case of the MSRP WebSocket Clients, there are
   three possible authentication mechanisms:

   1. HTTP Digest authentication in AUTH (as per [RFC4976]).

   2. Cookie-based or HTTP Digest authentication in the WebSocket
       Handshake (see Section 7).

   3. Mutual TLS between the WebSocket-based MSRP client and the
       WebSocket server.

   The AUTH request is a required event when authentication occurs at
   the WebSocket connection level since the "Use-Path:" header required
   to create the SDP offer is included in the 200 OK response to the
   AUTH request.

6. Connection Keepalive

   It is RECOMMENDED that MSRP WebSocket Clients and Servers keep their
   WebSocket connections open by sending periodic WebSocket "Ping"
   frames as described in Section 5.5.2 of [RFC6455].

   The WebSocket API [WS-API] does not provide a mechanism for
   applications running in a web browser to control whether or not
   periodic WebSocket "Ping" frames are sent to the server. The
   implementation of such a keepalive feature is the decision of each
   web browser manufacturer and may also depend on the configuration of
   the web browser.

   A future WebSocket protocol extension providing a similar keepalive
   mechanism could also be used.

   When MSRP WebSocket Clients or Servers cannot use WebSocket "Ping"
   frames to keep connections open, an MSRP implementation MAY use
   bodiless SEND requests as described in Section 7.1 of [RFC4975].
   MSRP WebSocket Clients or Servers MUST be prepared to receive
   bodiless SEND requests.

7. Authentication

   Prior to sending MSRP requests, an MSRP WebSocket Client connects to
   an MSRP WebSocket Server and performs the connection handshake. As
   described in Section 3, the handshake procedure involves a HTTP GET
   method request from the Client and a response from the Server
   including an HTTP 101 status code.

   In order to authorize the WebSocket connection, the MSRP WebSocket
   Server MAY inspect any HTTP headers present (for example, Cookie
   [RFC6265], Host [RFC7230], or Origin [RFC6454]) in the HTTP GET
   request. For many web applications, the value of such a Cookie is
   provided by the web server once the user has authenticated themselves
   to the web server, which could be done by many existing mechanisms.
   As an alternative method, the MSRP WebSocket Server could request
   HTTP authentication by replying to the Client's GET method request
   with a HTTP 401 status code. The WebSocket protocol [RFC6455] covers
   this usage in Section 4.1 and is paraphrased as follows:

      If the status code received from the server is not 101, the
      WebSocket client stack handles the response per HTTP [RFC7230]
      procedures; in particular, the client might perform authentication
      if it receives a 401 status code.

If the HTTP GET request contains an Origin header, the MSRP WebSocket
Server SHOULD indicate Cross-Origin Resource Sharing [CORS] by adding
an Access-Control-Allow-Origin header to the 101 response.

Regardless of whether the MSRP WebSocket Server requires
authentication during the WebSocket handshake, authentication MAY be
requested at the MSRP protocol level by an MSRP Server challenging
AUTH requests using a 401 response. Therefore, an MSRP WebSocket
Client SHOULD support HTTP Digest [RFC7235] authentication as stated
in [RFC4976].

8. Examples

8.1. Authentication

8.1.1. WebSocket Authentication

```
Alice     (MSRP WSS)    a.example.com
 |                           |
 |HTTP GET (WS handshake) F1 |
 |-------------------------->|
 |101 Switching Protocols F2 |
 |<--------------------------|
 |                           |
 |AUTH F3                    |
 |-------------------------->|
 |200 OK F4                  |
 |<--------------------------|
 |                           |
```

Alice loads a web page using her web browser and retrieves JavaScript
code implementing the WebSocket MSRP subprotocol defined in this
document. The JavaScript code (an MSRP WebSocket Client) establishes
a secure WebSocket connection with an MSRP relay (an MSRP WebSocket
Server) at a.example.com. Upon WebSocket connection, Alice
constructs and sends an MSRP AUTH request. Since the JavaScript
stack in a browser has no way to determine the local address from
which the WebSocket connection was made, this implementation uses a
random ".invalid" domain name for the hostpart of the From-Path URI
(see Appendix A).

   In this example, it is assumed that authentication is performed at
   the WebSocket layer (not shown), so no challenge is issued for the
   MSRP AUTH message:

   F1 HTTP GET (WS handshake) Alice -> a.example.com (TLS)

   GET / HTTP/1.1
   Host: a.example.com
   Upgrade: websocket
   Connection: Upgrade
   Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
   Origin: https://www.example.com
   Sec-WebSocket-Protocol: msrp
   Sec-WebSocket-Version: 13

   F2 101 Switching Protocols a.example.com -> Alice (TLS)

   HTTP/1.1 101 Switching Protocols
   Upgrade: websocket
   Connection: Upgrade
   Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
   Sec-WebSocket-Protocol: msrp

   F3 AUTH Alice -> a.example.com (transport WSS)

   MSRP 49fi AUTH
   To-Path: msrps://alice@a.example.com:443;ws
   From-Path: msrps://df7jal23ls0d.invalid:2855/98cjs;ws
   -------49fi$

   F4 200 OK a.example.com -> Alice (transport WSS)

   MSRP 49fi 200 OK
   To-Path: msrps://df7jal23ls0d.invalid:2855/98cjs;ws
   From-Path: msrps://alice@a.example.com:443;ws
   Use-Path: msrps://a.example.com:2855/jui787s2f;tcp
   Expires: 900
   -------49fi$

8.1.2. MSRP Authentication

```
Alice     (MSRP WSS)      a.example.com
|                         |
|HTTP GET (WS handshake) F1   |
|-------------------------->|
|101 Switching Protocols F2   |
|<--------------------------|
|                         |
|AUTH F3                  |
|-------------------------->|
|401 Unauthorized F4                 |
|<--------------------------|
|AUTH F5                  |
|-------------------------->|
|200 OK F6                |
|<--------------------------|
|                         |
```

   This example uses the same scenario as [Section 8.1.1] but with
   authentication performed at the MSRP layer.

   Note that MSRP does not permit line folding. A "\" in the examples
   shows a line continuation due to limitations in line length of this
   document. Neither the backslash nor the extra CRLF is included in
   the actual MSRP message.

   F1 HTTP GET (WS handshake) Alice -> a.example.com (TLS)

   GET / HTTP/1.1
   Host: a.example.com
   Upgrade: websocket
   Connection: Upgrade
   Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
   Origin: https://www.example.com
   Sec-WebSocket-Protocol: msrp
   Sec-WebSocket-Version: 13

   F2 101 Switching Protocols a.example.com -> Alice (TLS)

   HTTP/1.1 101 Switching Protocols
   Upgrade: websocket
   Connection: Upgrade
   Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
   Sec-WebSocket-Protocol: msrp

    F3 AUTH Alice -> a.example.com (transport WSS)

    MSRP 4rsxt9nz AUTH
    To-Path: msrps://alice@a.example.com:443;ws
    From-Path: msrps://df7jal23ls0d.invalid:2855/98cjs;ws
    -------4rsxt9nz$

    F4 401 Unauthorized a.example.com -> Alice (transport WSS)

    MSRP 4rsxt9nz 401 Unauthorized
    To-Path: msrps://df7jal23ls0d.invalid:2855/98cjs;ws
    From-Path: msrps://alice@a.example.com:443;ws
    WWW-Authenticate: Digest realm="example.com", \
     nonce="UvtfpVL7XnnJ63EE244fXDthfLihlMHOY4+dd4A=", qop="auth"
    -------4rsxt9nz$

    F5 AUTH Alice -> a.example.com (transport WSS)

    MSRP qy1hsow5 AUTH
    To-Path: msrps://alice@a.example.com:443;ws
    From-Path: msrps://df7jal23ls0d.invalid:2855/98cjs;ws
    Authorization: Digest username="alice", realm="example.com", \
     nonce="UvtfpVL7XnnJ63EE244fXDthfLihlMHOY4+dd4A=", \
     uri="msrps://alice@a.example.com:443;ws", \
     response="5011d0d58fe975e0d0cdc007ae26f4b7", \
     qop=auth, cnonce="zic5ml401prb", nc=00000001
    -------qy1hsow5$

    F6 200 OK a.example.com -> Alice (transport WSS)

    MSRP qy1hsow5 200 OK
    To-Path: msrps://df7jal23ls0d.invalid:2855/98cjs;ws
    From-Path: msrps://alice@a.example.com:443;ws
    Use-Path: msrps://a.example.com:2855/jui787s2f;tcp
    Expires: 900
    -------qy1hsow5$

8.2. Example Session: MSRP WebSocket Client to MSRP Client

   The following subsections show various message exchanges occurring
   during the course of an MSRP session between a WebSocket client and a
   non-WebSocket client.

8.2.1. SDP Exchange

   The following example shows SDP that could be included in a SIP
   message to set up an MSRP session between Alice and Bob where Alice
   uses a WebSocket MSRP relay and Bob uses a traditional MSRP client
   without a relay.

   A "\" in the examples shows a line continuation due to limitations in
   line length of this document. Neither the backslash nor the extra
   CRLF is included in the actual SDP.

   Alice makes an offer with a path including the relay (having already
   successfully authenticated with the relay):

   c=IN IP4 a.example.com
   m=message 1234 TCP/TLS/MSRP *
   a=accept-types:message/cpim text/plain text/html
   a=path:msrps://a.example.com:2855/jui787s2f;tcp \
          msrps://df7jal23ls0d.invalid:2855/98cjs;ws

   In this offer, Alice wishes to receive MSRP messages via the relay at
   a.example.com. She wants to use TLS as the transport for the MSRP
   session (beyond the relay). She can accept message/cpim, text/plain,
   and text/html message bodies in SEND requests.

   Bob's answer to this offer could look like:

   c=IN IP4 bob.example.com
   m=message 1234 TCP/TLS/MSRP *
   a=accept-types:message/cpim text/plain
   a=path:msrps://bob.example.com:49154/foo;tcp

   Here, Bob wishes to receive the MSRP messages at bob.example.com. He
   can accept only message/cpim and text/plain message bodies in SEND
   requests and has rejected the text/html content offered by Alice. He
   does not need a relay to set up the MSRP session.

8.2.2. SEND (MSRP WebSocket Client to MSRP Client)
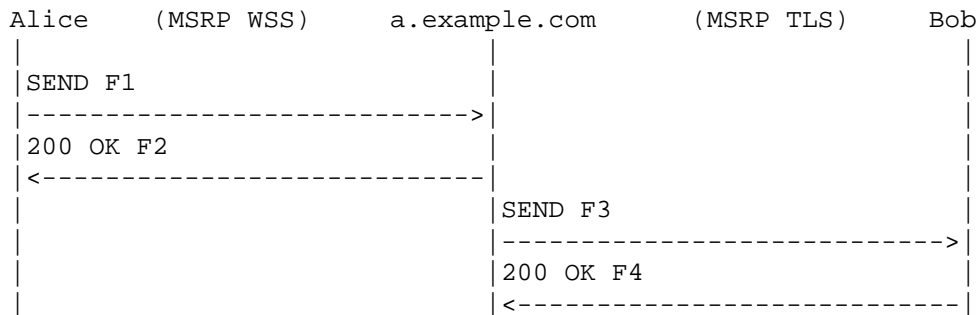
```
Alice    (MSRP WSS)    a.example.com      (MSRP TLS)     Bob
|                         |                         |
|SEND F1                  |                         |
|------------------------>|                         |
|200 OK F2                |                         |
|<------------------------|                         |
|                         |SEND F3                  |
|                         |------------------------>|
|                         |200 OK F4                |
|                         |<------------------------|
```

Later in the session, Alice sends an instant message to Bob. The
MSRP WebSocket Server at a.example.com acts as an MSRP relay, routing
the message to Bob over TLS.

Message details (A "\" in the examples shows a line continuation due
to limitations in line length of this document. Neither the
backslash nor the extra CRLF is included in the actual request or
response):

F1 SEND Alice -> a.example.com (transport WSS)

```
MSRP 6aef SEND
To-Path: msrps://a.example.com:2855/jui787s2f;tcp \
        msrps://bob.example.com:49154/foo;tcp
From-Path: msrps://df7jal23ls0d.invalid:2855/98cjs;ws
Success-Report: no
Byte-Range: 1-*/*
Message-ID: 87652
Content-Type: text/plain

Hi Bob, I'm about to send you file.mpeg
-------6aef$
```

F2 200 OK a.example.com -> Alice (transport WSS)

```
MSRP 6aef 200 OK
To-Path: msrps://df7jal23ls0d.invalid:2855/98cjs;ws
From-Path: msrps://a.example.com:2855/jui787s2f;tcp
-------6aef$
```

      F3 SEND a.example.com -> Bob (transport TLS)

      MSRP juh76 SEND
      To-Path: msrps://bob.example.com:49154/foo;tcp
      From-Path: msrps://a.example.com:2855/jui787s2f;tcp \
                 msrps://df7jal23ls0d.invalid:2855/98cjs;ws
      Success-Report: no
      Byte-Range: 1-*/*
      Message-ID: 87652
      Content-Type: text/plain

      Hi Bob, I'm about to send you file.mpeg
      -------juh76$

      F4 200 OK Bob -> a.example.com (transport TLS)

      MSRP juh76 200 OK
      To-Path: msrps://a.example.com:2855/jui787s2f;tcp
      From-Path: msrps://bob.example.com:49154/foo;tcp
      -------juh76$

8.2.3. SEND (MSRP Client to MSRP WebSocket Client)

      Bob   (MSRP TLS)   a.example.com   (MSRP WSS)   Alice
      | | |
      |SEND F1 | |
      |------------------------>| |
      |200 OK F2 | |
      |<------------------------| |
      | |SEND F3 |
      | |------------------------------>|
      | |200 OK F4 |
      | |<------------------------------|
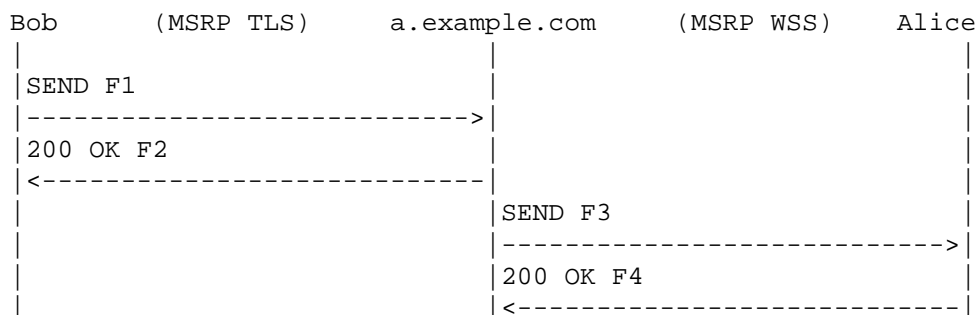
      Later in the session, Bob sends an instant message to Alice. The
      MSRP WebSocket Server at a.example.com acts as an MSRP relay, routing
      the message to Alice over secure WebSocket.

   Message details (A "\" in the examples shows a line continuation due
   to limitations in line length of this document. Neither the
   backslash nor the extra CRLF is included in the actual request or
   response):

   F1 SEND Bob -> a.example.com (transport TLS)

   MSRP xght6 SEND
   To-Path: msrps://a.example.com:2855/jui787s2f;tcp \
           msrps://df7jal23ls0d.invalid:2855/98cjs;ws
   From-Path: msrps://bob.example.com:49154/foo;tcp
   Success-Report: no
   Byte-Range: 1-*/*
   Message-ID: 87652
   Content-Type: text/plain

   Thanks for the file.
   -------xght6$

   F2 200 OK a.example.com -> Bob (transport TLS)

   MSRP xght6 200 OK
   To-Path: msrps://bob.example.com:49154/foo;tcp
   From-Path: msrps://a.example.com:2855/jui787s2f;tcp
   -------xght6$

   F3 SEND a.example.com -> Alice (transport WSS)

   MSRP yh67 SEND
   To-Path: msrps://df7jal23ls0d.invalid:2855/98cjs;ws
   From-Path: msrps://a.example.com:2855/jui787s2f;tcp \
             msrps://bob.example.com:49154/foo;tcp
   Success-Report: no
   Byte-Range: 1-*/*
   Message-ID: 87652
   Content-Type: text/plain

   Thanks for the file.
   -------yh67$

```
   F4 200 OK  Alice -> a.example.com (transport WSS)

   MSRP yh67 200 OK
   To-Path:  msrps://a.example.com:2855/jui787s2f;tcp
   From-Path:  msrps://df7jal23ls0d.invalid:2855/98cjs;ws
   -------yh67$
```

8.3. Example Session: Two MSRP WebSocket Clients

   The following subsections show various message exchanges occurring
   during the course of an MSRP session between two WebSocket clients.

8.3.1. SDP Exchange

   The following example shows SDP that could be included in a SIP
   message to set up an MSRP session between Alice and Carol where both
   of them are using the same WebSocket MSRP relay.

   Alice makes an offer with a path including the relay (having already
   successfully authenticated with the relay):

```
   c=IN IP4 a.example.com
   m=message 1234 TCP/TLS/MSRP *
   a=accept-types:message/cpim text/plain text/html
   a=path:msrps://a.example.com:2855/jui787s2f;tcp \
         msrps://df7jal23ls0d.invalid:2855/98cjs;ws
```

   In this offer, Alice wishes to receive MSRP messages via the relay at
   a.example.com. She wants to use TLS as the transport for the MSRP
   session (beyond the relay). She can accept message/cpim, text/plain,
   and text/html message bodies in SEND requests.

   Carol's answer to this offer could look like:

```
   c=IN IP4 a.example.com
   m=message 1234 TCP/TLS/MSRP *
   a=accept-types:message/cpim text/plain
   a=path:msrps://a.example.com:2855/iwnslt;tcp \
         msrps://jk9awp14vj8x.invalid:2855/76qwe;ws
```

   Here, Carol also wishes to receive the MSRP messages via
   a.example.com. She can accept only message/cpim and text/plain
   message bodies in SEND requests and has rejected the text/html
   content offered by Alice.

8.3.2. SEND

```
Alice    (MSRP WSS)      a.example.com     (MSRP WSS)     Carol
|                          |                          |
|SEND F1                   |                          |
|------------------------->|                          |
|200 OK F2                 |                          |
|<-------------------------|                          |
|                          |SEND F3                   |
|                          |------------------------->|
|                          |200 OK F4                 |
|                          |<-------------------------|
```
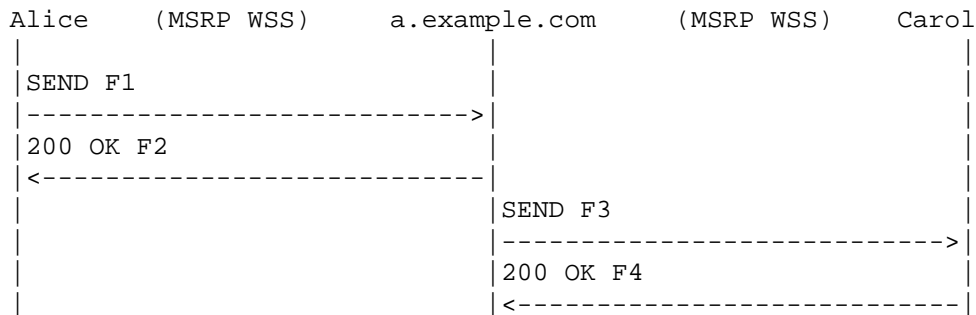
Later in the session, Alice sends an instant message to Carol. The
MSRP WebSocket Server at a.example.com acts as an MSRP relay, routing
the message to Carol over secure WebSocket.

In this example, both Alice and Carol are using MSRP WebSocket
Clients and the same MSRP WebSocket Server. This means that
a.example.com will appear twice in the To-Path in F1. a.example.com
can either handle this internally or loop the MSRP SEND request back
to itself as if it were two separate MSRP relays.

Message details (A "\" in the examples shows a line continuation due
to limitations in line length of this document. Neither the
backslash nor the extra CRLF is included in the actual request or
response):

F1 SEND Alice -> a.example.com (transport WSS)

MSRP kjh6 SEND
To-Path: msrps://a.example.com:2855/jui787s2f;tcp \
         msrps://a.example.com:2855/iwnslt;tcp \
         msrps://jk9awp14vj8x.invalid:2855/76qwe;ws
From-Path: msrps://df7jal23ls0d.invalid:2855/98cjs;ws
Success-Report: no
Byte-Range: 1-*/*
Message-ID: 87652
Content-Type: text/plain

Carol, I sent that file to Bob.
-------kjh6$

```
   F2 200 OK  a.example.com -> Alice (transport WSS)

   MSRP kjh6 200 OK
   To-Path: msrps://df7jal23ls0d.invalid:2855/98cjs;ws
   From-Path: msrps://a.example.com:2855/jui787s2f;tcp
   -------kjh6$


   F3 SEND  a.example.com -> Carol (transport WSS)

   MSRP re58 SEND
   To-Path: msrps://jk9awp14vj8x.invalid:2855/76qwe;ws
   From-Path: msrps://a.example.com:2855/iwnslt;tcp \
           msrps://a.example.com:2855/jui787s2f;tcp \
           msrps://df7jal23ls0d.invalid/98cjs;ws
   Success-Report: no
   Byte-Range: 1-*/*
   Message-ID: 87652
   Content-Type: text/plain

   Carol, I sent that file to Bob.
   -------re58$


   F4 200 OK  Carol -> a.example.com (transport WSS)

   MSRP re58 200 OK
   To-Path: msrps://a.example.com:2855/iwnslt;tcp
   From-Path: msrps://jk9awp14vj8x.invalid:2855/76qwe;ws
   -------re58$
```

8.4. Example Session: MSRP WebSocket Client to MSRP Client Using a
      Relay

   The following subsections show various message exchanges occurring
   during the course of an MSRP session between a WebSocket client and a
   non-WebSocket client, where the latter is also using an MSRP relay.

8.4.1. SDP Exchange

   The following example shows SDP that could be included in a SIP
   message to set up an MSRP session between Alice and Bob where Alice
   uses a WebSocket MSRP relay and Bob uses a traditional MSRP client
   with a separate relay.

Alice makes an offer with a path including the relay (having already
successfully authenticated with the relay):

```
c=IN IP4 a.example.com
m=message 1234 TCP/TLS/MSRP *
a=accept-types:message/cpim text/plain text/html
a=path:msrps://a.example.com:2855/jui787s2f;tcp \
        msrps://df7jal23ls0d.invalid:2855/98cjs;ws
```
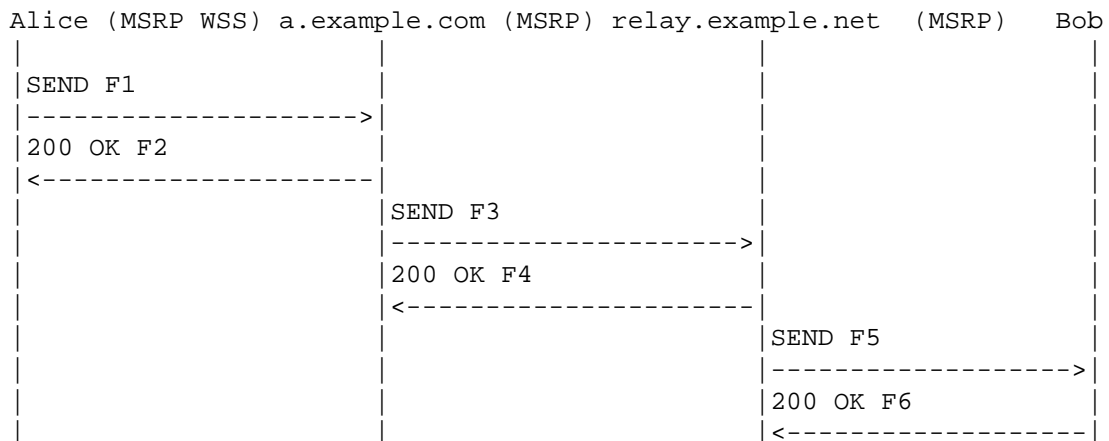
In this offer, Alice wishes to receive MSRP messages via the relay at
a.example.com. She wants to use TLS as the transport for the MSRP
session (beyond the relay). She can accept message/cpim, text/plain,
and text/html message bodies in SEND requests.

Bob's answer to this offer could look like:

```
c=IN IP4 bob.example.com
m=message 1234 TCP/TLS/MSRP *
a=accept-types:message/cpim text/plain
a=path:msrps://relay.example.net:2855/kwvin5f;tcp \
        msrps://bob.example.com:49154/foo;tcp
```

Here, Bob wishes to receive the MSRP messages via the relay at
relay.example.net. He can accept only message/cpim and text/plain
message bodies in SEND requests and has rejected the text/html
content offered by Alice.

8.4.2. SEND

```
Alice (MSRP WSS) a.example.com (MSRP) relay.example.net  (MSRP)    Bob
 |                    |                   |                      |
 |SEND F1             |                   |                      |
 |-------------------->|                   |                      |
 |200 OK F2           |                   |                      |
 |<--------------------|                   |                      |
 |                    |SEND F3            |                      |
 |                    |--------------------->|                      |
 |                    |200 OK F4          |                      |
 |                    |<---------------------|                      |
 |                    |                   |SEND F5               |
 |                    |                   |--------------------->|
 |                    |                   |200 OK F6             |
 |                    |                   |<--------------------|
```

Later in the session, Alice sends an instant message to Bob. The
MSRP WebSocket Server at a.example.com acts as an MSRP relay, routing
the message to Bob via his relay, relay.example.net.

   Message details (A "\" in the examples shows a line continuation due
   to limitations in line length of this document. Neither the
   backslash nor the extra CRLF is included in the actual request or
   response):

   F1 SEND Alice -> a.example.com (transport WSS)

   MSRP Ycwt SEND
   To-Path: msrps://a.example.com:2855/jui787s2f;tcp \
           msrps://relay.example.net:2855/kwvin5f;tcp \
           msrps://bob.example.com:49154/foo;tcp
   From-Path: msrps://df7jal23ls0d.invalid:2855/98cjs;ws
   Success-Report: no
   Byte-Range: 1-*/*
   Message-ID: 87652
   Content-Type: text/plain

   Bob, that was the wrong file - don't watch it!
   -------Ycwt$

   F2 200 OK a.example.com -> Alice (transport WSS)

   MSRP Ycwt 200 OK
   To-Path: msrps://df7jal23ls0d.invalid:2855/98cjs;ws
   From-Path: msrps://a.example.com:2855/jui787s2f;tcp
   -------Ycwt$

   F3 SEND a.example.com -> relay.example.net (transport TLS)

   MSRP 13GA SEND
   To-Path: msrps://relay.example.net:2855/kwvin5f;tcp \
           msrps://bob.example.com:49154/foo;tcp
   From-Path: msrps://a.example.com:2855/jui787s2f;tcp \
             msrps://df7jal23ls0d.invalid/98cjs;ws
   Success-Report: no
   Byte-Range: 1-*/*
   Message-ID: 87652
   Content-Type: text/plain

   Bob, that was the wrong file - don't watch it!
   -------13GA$

```
F4 200 OK  relay.example.net -> a.example.com (transport TLS)

MSRP 13GA 200 OK
To-Path: msrps://a.example.com:2855/iwnslt;tcp
From-Path: msrps://relay.example.net:2855/kwvin5f;tcp
-------13GA$


F5 SEND   relay.example.net -> bob.example.com (transport TLS)

MSRP kXeg SEND
To-Path: msrps://bob.example.com:49154/foo;tcp
From-Path: msrps://relay.example.net:2855/kwvin5f;tcp \
           msrps://a.example.com:2855/jui787s2f;tcp \
           msrps://df7jal23ls0d.invalid/98cjs;ws
Success-Report: no
Byte-Range: 1-*/*
Message-ID: 87652
Content-Type: text/plain

Bob, that was the wrong file - don't watch it!
-------kXeg$


F6 200 OK  bob.example.com -> relay.example.net (transport TLS)

MSRP kXeg 200 OK
To-Path: msrps://relay.example.net:2855/kwvin5f;tcp
From-Path: msrps://bob.example.com:49154/foo;tcp
-------kXeg$
```

9. Security Considerations

   MSRP traffic transported over WebSockets MUST be protected by using a
   secure WebSocket connection (using TLS [RFC5246] over TCP).

   When establishing a connection using MSRP over secure WebSockets, the
   client MUST authenticate the server using the server's certificate
   according to the WebSocket validation procedure in [RFC6455].

   Any security considerations specific to the WebSocket protocol are
   detailed in the relevant specification [RFC6455] and are considered
   outside the scope of this document. The certificate name matching
   (described by [RFC6455]) and cryptosuite selection will be handled by
   the browser, and the browser's procedures will supersede those
   specified in [RFC4975].

   Since the TLS session is always terminated at the MSRP WebSocket
   Server and the WebSocket server can see the plain text, the MSRP
   client (browser) SHOULD NOT indicate end-to-end security to user.

   TLS, as used in this document, should follow the best current
   practices defined in [RFC7525].

10. IANA Considerations

   Per this specification, IANA has registered the WebSocket MSRP
   subprotocol in the "WebSocket Subprotocol Name Registry" with the
   following data:

   Subprotocol Identifier: msrp

   Subprotocol Common Name: WebSocket Transport for MSRP (Message
      Session Relay Protocol)

   Subprotocol Definition: RFC 7977

   Reference: RFC 7977

11. References

11.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119,
              DOI 10.17487/RFC2119, March 1997,
              <http://www.rfc-editor.org/info/rfc2119>.

   [RFC4975] Campbell, B., Ed., Mahy, R., Ed., and C. Jennings, Ed.,
              "The Message Session Relay Protocol (MSRP)", RFC 4975,
              DOI 10.17487/RFC4975, September 2007,
              <http://www.rfc-editor.org/info/rfc4975>.

   [RFC4976] Jennings, C., Mahy, R., and A. Roach, "Relay Extensions
              for the Message Sessions Relay Protocol (MSRP)", RFC 4976,
              DOI 10.17487/RFC4976, September 2007,
              <http://www.rfc-editor.org/info/rfc4976>.

   [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax
              Specifications: ABNF", STD 68, RFC 5234,
              DOI 10.17487/RFC5234, January 2008,
              <http://www.rfc-editor.org/info/rfc5234>.

   [RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol",
              RFC 6455, DOI 10.17487/RFC6455, December 2011,
              <http://www.rfc-editor.org/info/rfc6455>.

11.2. Informative References

   [CORS]   van Kesteren, A., Ed., "Cross-Origin Resource Sharing",
              W3C Recommendation, January 2014,
              <http://www.w3.org/TR/2014/REC-cors-20140116/>.

   [RFC2606] Eastlake 3rd, D. and A. Panitz, "Reserved Top Level DNS
              Names", BCP 32, RFC 2606, DOI 10.17487/RFC2606, June 1999,
              <http://www.rfc-editor.org/info/rfc2606>.

   [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
              Resource Identifier (URI): Generic Syntax", STD 66,
              RFC 3986, DOI 10.17487/RFC3986, January 2005,
              <http://www.rfc-editor.org/info/rfc3986>.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <http://www.rfc-editor.org/info/rfc5246>.

   [RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265,
              DOI 10.17487/RFC6265, April 2011,
              <http://www.rfc-editor.org/info/rfc6265>.

   [RFC6454] Barth, A., "The Web Origin Concept", RFC 6454,
              DOI 10.17487/RFC6454, December 2011,
              <http://www.rfc-editor.org/info/rfc6454>.

   [RFC6714] Holmberg, C., Blau, S., and E. Burger, "Connection
              Establishment for Media Anchoring (CEMA) for the Message
              Session Relay Protocol (MSRP)", RFC 6714,
              DOI 10.17487/RFC6714, August 2012,
              <http://www.rfc-editor.org/info/rfc6714>.

   [RFC7118] Baz Castillo, I., Millan Villegas, J., and V. Pascual,
              "The WebSocket Protocol as a Transport for the Session
              Initiation Protocol (SIP)", RFC 7118,
              DOI 10.17487/RFC7118, January 2014,
              <http://www.rfc-editor.org/info/rfc7118>.

   [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
              Protocol (HTTP/1.1): Message Syntax and Routing",
              RFC 7230, DOI 10.17487/RFC7230, June 2014,
              <http://www.rfc-editor.org/info/rfc7230>.

   [RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
              Protocol (HTTP/1.1): Authentication", RFC 7235,
              DOI 10.17487/RFC7235, June 2014,
              <http://www.rfc-editor.org/info/rfc7235>.

   [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre,
              "Recommendations for Secure Use of Transport Layer
              Security (TLS) and Datagram Transport Layer Security
              (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May
              2015, <http://www.rfc-editor.org/info/rfc7525>.

   [WS-API]   Hickson, I., Ed., "The WebSocket API", W3C Candidate
              Recommendation, September 2012,
              <https://www.w3.org/TR/2012/CR-websockets-20120920/>.

Appendix A.   Implementation Guidelines: MSRP WebSocket Client
              Considerations

   The JavaScript stack in web browsers does not have the ability to
   discover the local transport address used for originating WebSocket
   connections. Therefore, the MSRP WebSocket Client constructs a
   domain name consisting of a random token followed by the ".invalid"
   top-level domain name, as stated in [RFC2606], and uses it within its
   From-Path headers.

   The From-Path URI provided by MSRP clients that use an MSRP relay is
   not used for routing MSRP messages, thus, it is safe to set a random
   domain in the hostpart of the From-Path URI.

Acknowledgements

   Special thanks to Inaki Baz Castillo, Jose Luis Millan Villegas, and
   Victor Pascual, the authors of [RFC7118], which has inspired this
   document.

   Additional thanks to Inaki Baz Castillo, who pointed out that "web
   browser" shouldn't be used all the time, as this specification should
   be valid for smartphones and apps other than browsers and suggested
   clarifications to the SDP handling for MSRP over WebSocket.

   Special thanks to James Wyatt from Crocodile RCS Ltd for helping with
   the JavaScript MSRP-over-WebSockets prototyping.

   Special thanks to Anton Roman who has contributed to this document.

   Thanks to Saul Ibarra Corretge for suggesting that the existing MSRP
   keepalive mechanism may be used when WebSocket pings are not
   available.

   Thanks to Ben Campbell, Inaki Baz Castillo, Keith Drage, Olle
   Johansson, and Christer Holmberg for their thoughtful discussion
   comments and review feedback that led to the improvement of this
   document. Special thanks to Mary Barnes for both her technical
   review and for offering to act as Document Shepherd. Thanks also to
   Stephen Farrell, Alissa Cooper, Mirja Kuehlewind, Allison Mankin,
   Alexey Melnikov, and Kathleen Moriarty for their review comments.

Authors' Addresses

   Peter Dunkley
   Xura
   Lancaster Court
   8 Barnes Wallis Road
   Fareham PO15 5TU
   United Kingdom

   Email: peter.dunkley@xura.com

   Gavin Llewellyn
   Xura
   Lancaster Court
   8 Barnes Wallis Road
   Fareham PO15 5TU
   United Kingdom

   Email: gavin.llewellyn@xura.com

   Victor Pascual
   Oracle

   Email: victor.pascual.avila@oracle.com

   Gonzalo Salgueiro
   Cisco Systems, Inc.
   7200-12 Kit Creek Road
   Research Triangle Park, NC 27709
   United States of America

   Email: gsalguei@cisco.com

   Ram Mohan Ravindranath
   Cisco Systems, Inc.

   Email: rmohanr@cisco.com