Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 16, 2012

I. Baz Castillo
J. Luis Millan
XtraTelecom S.A.
V. Pascual
Acme Packet

September 13, 2011

WebSocket Transport for Session Initiation Protocol (SIP)
draft-ibc-rtcweb-sip-websocket-00

Abstract

This document specifies a WebSocket subprotocol for a new transport in SIP (Session Initiation Protocol). The WebSocket protocol enables two-way realtime communication between clients (typically web-based applications) and servers. The main goal of this specification is to integrate the SIP protocol within web applications.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Conventions 4
3. Scope 5
4. SIP WebSocket Transport 6
4.1. Via Transport Parameter 6
4.2. SIP URI Transport Parameter 6
4.3. Sending Responses 7
5. The WebSocket SIP Subprotocol 8
6. WebSocket Client Usage 9
6.1. WebSocket Disconnection 10
7. WebSocket Server Usage 11
7.1. SIP Proxy Considerations 11
8. WebSocket Connection Keep Alive 12
9. Examples 13
9.1. Registration 13
9.2. INVITE dialog through a proxy 15
9.3. INVITE dialog through two proxies 18
10. Security Considerations 23
11. IANA Considerations 24
11.1. Registration of new Via transports 24
11.2. Registration of new SIP URI transport 24
11.3. Registration of the WebSocket SIP subprotocol 24
12. References 25
12.1. Normative References 25
12.2. Informative References 25
Authors' Addresses 27

1. Introduction

Integrating the SIP protocol [RFC3261] within modern web-based applications has been a hard task historically due to the specification complexity and inherent limitations in web browsers and HTTP protocol [RFC2616]. The arrival of WebSocket [I-D.ietf-hybi-thewebsocketprotocol] and [RTC-Web] (Real Time Collaboration on the World Wide Web) provides a two-way communication technology for web-based applications along with multimedia capabilities for audio and video sessions in web browsers, making feasible the requeriments of the SIP protocol.

This specification defines a new WebSocket subprotocol for transporting SIP messages between a WebSocket client and server, a new transport for the SIP protocol and procedures for SIP proxies when behaving as a bridge between WebSocket and other SIP transports. No changes have been made to the SIP protocol [RFC3261].

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 [RFC2119].

3. Scope

The WebSocket protocol is mostly suitable for web-based applications running in a web browser. Other applications running out of web browsers do not have the constraints of web applications since typically they can directly access to the transport layer.

In the same manner, the WebSocket protocol adds a network overhead since it works as an intermediary layer between the transport and application layers. There is no benefit on using SIP over WebSocket transport between two SIP nodes when none of them runs within a web browser. Even more, the WebSocket protocol is not symmetric since just a WebSocket client can open a connection to a WebSocket server (a WebSocket client does not listen for incoming connections).

Given these arguments, this specification is mostly focused on integrating the SIP protocol within web-based applications or any client application using the WebSocket protocol. Other aspects such as DNS NAPTR/SRV resolution for SIP over WebSocket transport are not covered by this specification since they are mainly useless given the WebSocket protocol nature.

This document just covers SIP as a signalling protocol, leaving multimedia capabilities integration for a separate document once [RTC-Web] (Real Time Collaboration on the World Wide Web) becomes a standard.

4. SIP WebSocket Transport

WebSocket [I-D.ietf-hybi-thewebsocketprotocol] is a reliable protocol and therefore the WebSocket subprotocol for a SIP transport defined by this document is also a reliable transport. Thus, client and server transactions using WebSocket transport MUST follow the procedures and timer values for a reliable transport as defined in [RFC3261].

4.1. Via Transport Parameter

Via header fields carry the transport protocol identifier. This document defines the value "WS" to be used for requests over plain WebSocket protocol and "WSS" for requests over secure WebSocket protocol (in which the WebSocket session is established on top of TLS

[RFC5246] over TCP transport).

The updated augmented BNF (Backus-Naur Form) [RFC5234] for this parameter is the following (the original BNF for this parameter can be found in [RFC3261]):

```
  transport         =  "UDP" / "TCP" / "TLS" / "WS" / "WSS"
                       / other-transport
```

The following are examples of Via header fields using "WS" and "WSS":

```
  Via: SIP/2.0/WS 1.2.3.4:28456
  Via: SIP/2.0/WSS [2001:0:63ba:74c:1806:7ea2:9aab:f892]:32802
```

4.2. SIP URI Transport Parameter

This document defines the value "ws" as the transport parameter value for a SIP URI [RFC3986] to be contacted using WebSocket protocol. Whether to select a plain or secure WebSocket connection depends on the SIP URI schema ("sip" schema means plain WebSocket connection while "sips" schema requires secure WebSocket connection).

The updated augmented BNF (Backus-Naur Form) [RFC5234] for this parameter is the following (the original BNF for this parameter can be found in [RFC3261]):

```
  transport-param   =  "transport="
                       ( "udp" / "tcp" / "sctp" / "tls" / "ws"
                       / other-transport )
```

The following are examples of SIP URI's containing a "ws" transport parameter:

```
  sip:alice@1.2.3.4:28456;transport=ws
  sips:bob@[2001:0:63ba:74c:1806:7ea2:9aab:f892]:32802;transport=ws
```

4.3. Sending Responses

The SIP server transport uses the value of the top Via header field in order to determine where to send a response. If the "sent-protocol" is "WS" or "WSS" the response MUST be sent using the existing WebSocket connection to the source of the original request, if that connection is still open. This requires the server transport to maintain an association between server transactions and transport connections. If that connection is no longer open, the server MUST NOT attempt to open a WebSocket connection to the Via "sent-by"/"received"/"rport".

This is due the nature of the WebSocket protocol in which just the WebSocket client can establish a connection with the WebSocket server. A WebSocket client does not listen for incoming connections.

5. The WebSocket SIP Subprotocol

The term WebSocket subprotocol refers to the application-level protocol layered over a WebSocket connection. This document specifies the WebSocket SIP subprotocol for carrying SIP requests and responses through a WebSocket connection.

WebSocket [I-D.ietf-hybi-thewebsocketprotocol] defines message units as application data exchange for communication endpoints, becoming a message boundary protocol. These messages can contain UTF-8 text or binary data. The WebSocket SIP subprotocol specified in this document mandates messages of type UTF-8 text.

The WebSocket client and WebSocket server send SIP messages to each other. Each SIP message MUST be carried within a single WebSocket message and MUST be a complete SIP message, so a Content-Length header field is not mandatory. Sending more than one SIP message within a single WebSocket message is not allowed, neither sending an incomplete SIP message.

> This makes parsing of SIP messages easier on client side (typically web-based applications with an strict and simple API for receiving WebSocket messages). There is no need to establish boundaries (typically using Content-Length headers) between different messages. Same advantage is present in other message-based SIP transports as UDP or SCTP [RFC4168].

6. WebSocket Client Usage

As stated in [I-D.ietf-hybi-thewebsocketprotocol], a WebSocket URI [RFC3986] is given to the WebSocket client (typically within a web-based application) who resolves the URI destination and establishes a WebSocket connection with the corresponding server (by performing the handshake and negotiation procedures described in [I-D.ietf-hybi-thewebsocketprotocol]).

The client application is supposed to be provided with SIP account configuration values (as an AoR, outbound proxy and so on). Such values are used by the client application when generating SIP messages.

After establishing the WebSocket connection, the client SHOULD discover the source IP and port from which the server has received the TCP connection. Such IP and port are required for constructing the client's SIP local URI (to be used in Contact header during SIP registration and SIP dialogs).

> The mechanism used by the client application in order to discover its source IP and port is currently out of the scope of this specification, although it might be defined in future revisions of this document.

The client's SIP local URI MUST be constructed as follows:

- If the WebSocket connection is secure (given WebSocket URI has "wss" schema) the URI MUST have "sips" schema, "sip" otherwise.
- The URI username is up to the application.
- The URI hostport is determined by the local IP and port previously retrieved.
- A "transport" parameter with value "ws" MUST be added to the URI.

This SIP local URI MUST be used by the client as a registration binding (Contact URI in a REGISTER) and as a local target for SIP dialogs (Contact URI in a request or response) since this URI is the only adddress in which the client can be contacted, and just through the WebSocket server.

Any new request sent by the client MUST contain the discovered local IP and port in the Via "sent-by" field. Via "sent-transport" field MUST be set to "WSS" if the WebSocket connection is secure, to "WS" otherwise.

Due to the nature of the WebSocket protocol, the client sends all the SIP requests to the WebSocket server it is connected to, so the WebSocket server behaves as a de facto outbound SIP proxy.

In case the client application decides to close the WebSocket connection (for example when performing "logout" in a web application) it is recommended to remove the existing SIP registration binding (if present) by specifying an expiration interval of "0" for that contact address in a REGISTER request as described in section 10.2.2 of [RFC3261].

6.1. WebSocket Disconnection

In some circumstances the WebSocket connection could be terminated by the WebSocket server (for example when the server is restarted). If the client application wants to become reachable again it SHOULD reconnect to the WebSocket server and perform the SIP local URI discovery process again followed by a new SIP registration.

The client MAY also remove the previous registration binding in the registrar server, as such address is no longer reachable.

> When the WebSocket server is also the SIP registrar server, it MAY remove the SIP registration bindings associated to a WebSocket connection after such connection has been closed. Such a decision is out of the scope of this specification and depends on the SIP network topology.

7. WebSocket Server Usage

Here we assume that a SIP proxy or UAS (User Agent Server) is also acting as a WebSocket server implementing the WebSocket subprotocol described in this document. The server receives WebSocket connection attempts from clients. How the server authorizes or not those connections is out of the scope of this specification. Once the WebSocket subprotocol defined in this document has been negotiated, both client and server can send SIP messages to each other.

The server can only contact a SIP URI with the parameter "transport=ws" in case the destination address belongs to an existing WebSocket connection established from a WebSocket client. If not, a local transport error MUST be generated (which involves a 500 or 503 SIP response code).

> Such a case could happen when an existing SIP registration binding points to an already closed WebSocket connection which was not removed.

7.1. SIP Proxy Considerations

A SIP proxy implementing WebSocket transport can intercommunicate clients using SIP over WebSocket with other SIP clients or nodes using any other transport.

When the proxy bridges between WebSocket transport and any other SIP transport (including WebSocket transport) it MUST perform Loose Routing as specified in [RFC3261]. Otherwise in-dialog requests would fail since WebSocket clients cannot contact destinations other than their WebSocket server, and non-WebSocket SIP nodes cannot establish a connection to WebSocket clients. It is also recommended that the proxy follows recommendations in [RFC5658] and uses double Record-Route technique in these cases.

In the same way, if the SIP proxy implementing the WebSocket server behaves as an outbound proxy for REGISTER requests, it MUST add a Path header as described in [RFC3327]. Otherwise the WebSocket client would never receive incoming requests from the SIP registrar server after the lookup procedures in the SIP location service.

8. WebSocket Connection Keep Alive

It is recommended that the WebSocket client or server keeps the WebSocket connection open by sending periodic Ping frames as described in [I-D.ietf-hybi-thewebsocketprotocol] section 5.5.2. The mechanisms of decision for a WebSocket endpoint to maintain, or not, the connection over time is out of scope of this document.

In some cases due to transient network errors, the connection with the WebSocket server could be lost without the WebSocket client being aware of it. The WebSocket client would only realize of the network failure when attempting to send new data over the WebSocket connection.

The authors of this specification have requested the W3C (World Wide Web Consortium) to include a mechanism in the WebSocket API [WS-API] for instructing the WebSocket client to supervise the connection by sending periodical Ping frames at the interval requested by the API user.

9. Examples

The flows depicted in this section describe the behavior of an initial prototype which is currently under development.

9.1. Registration

```
   Alice    (SIP WS)     WebSocket SIP Server
   |                             |
   |OPTIONS F1                   |
   |---------------------------->|
   |200 OK F2                    |
   |<----------------------------|
   |                             |
   |REGISTER F3                  |
   |---------------------------->|
   |200 OK F4                    |
   |<---------------------------|
   |                            |
```

Alice is a WebSocket client running on a web browser. Alice establishes a plain WebSocket connection with a WebSocket server (also a SIP proxy/registrar) implementing the SIP subprotocol. Upon connection, Alice sends a SIP OPTIONS request including an empty "rport" parameter [RFC3581] in the Via header and obtains its source IP and port from the Via "received" and "rport" parameters in the response. Alice then forms its SIP local URI and constructs a REGISTER request.

Message details (authentication and SDP bodies are omitted for simplicity):

F1 OPTIONS Alice -> WebSocket SIP Server

```
OPTIONS sip:ws-server.atlanta.com SIP/2.0
Via: SIP/2.0/WS 1.2.3.4;branch=z9hG4bKasudf;rport
From: sip:alice@atlanta.com;tag=ux8asodj
To: sip:ws-server.atlanta.com
Call-ID: 87djahs72kjsd
CSeq: 1 OPTIONS
Max-Forwards: 1
Accept: application/sdp
```

```
   F2 200 OK WebSocket SIP Server -> Alice

   SIP/2.0 200 OK
   Via: SIP/2.0/WS 1.2.3.4;branch=z9hG4bKasudf;received=93.12.40.105;
     rport=19465
   From: sip:alice@atlanta.com;tag=ux8asodj
   To: sip:ws-server.atlanta.com;tag=jcx67hjm
   Call-ID: 87djahs72kjsd
   CSeq: 1 OPTIONS
   Content-Type: application/sdp


   F3 REGISTER Alice -> WebSocket SIP Server

   REGISTER sip:proxy.atlanta.com SIP/2.0
   Via: SIP/2.0/WS 93.12.40.105:19465;branch=z9hG4bKasudf
   From: sip:alice@atlanta.com;tag=65bnmj.34asd
   To: sip:ws-server.atlanta.com
   Call-ID: aiuy7k9njasd
   CSeq: 1 REGISTER
   Max-Forwards: 70
   Contact: <sip:alice@93.12.40.105:19465;transport=ws>


   F4 200 OK WebSocket SIP Server -> Alice

   SIP/2.0 200 OK
   Via: SIP/2.0/WS 93.12.40.105:19465;branch=z9hG4bKasudf
   From: sip:alice@atlanta.com;tag=65bnmj.34asd
   To: sip:ws-server.atlanta.com;tag=12isjljn8
   Call-ID: aiuy7k9njasd
   CSeq: 1 REGISTER
   Contact: <sip:alice@93.12.40.105:19465;transport=ws>
```

9.2. INVITE dialog through a proxy

```
   Alice       (SIP WSS)       SIP Proxy        (SIP UDP)        Carol
   |                             |                             |
   |INVITE F1                    |                             |
   |---------------------------->|                             |
   |100 Trying F2                |                             |
   |<----------------------------|                             |
   |                             |INVITE F3                    |
   |                             |---------------------------->|
   |                             |200 OK F4                    |
   |                             |<----------------------------|
   |200 OK F5                    |                             |
   |<----------------------------|                             |
   |                             |                             |
   |ACK F6                       |                             |
   |---------------------------->|                             |
   |                             |ACK F7                       |
   |                             |---------------------------->|
   |                             |                             |
   |                    Both Way RTP Media                     |
   |<=========================================================>|
   |                             |                             |
   |                             |BYE F8                       |
   |                             |<----------------------------|
   |BYE F9                       |                             |
   |<----------------------------|                             |
   |200 OK F10                   |                             |
   |---------------------------->|                             |
   |                             |200 OK F11                   |
   |                             |---------------------------->|
   |                             |                             |
```

Here the WebSocket server is also a SIP proxy and registrar for the domain atlanta.com. Alice, a WebSocket SIP client, calls Carol's AoR through a secure WebSocket connection. The WebSocket SIP server acts as a SIP proxy routing the INVITE to the UDP location of Carol. The proxy does Loose-Routing. Carol answers the call and terminates it later.

Message details (authentication and SDP bodies are omitted for simplicity):

F1 INVITE Alice -> SIP Proxy (transport WSS)

```
INVITE sip:carol@atlanta.com SIP/2.0
```

```
   Via: SIP/2.0/WSS 93.12.40.105:20565;branch=z9hG4bK56sdasks
   From: sip:alice@atlanta.com;tag=asdyka899
   To: sip:carol@atlanta.com
   Call-ID: asidkj3ss
   CSeq: 1 INVITE
   Max-Forwards: 70
   Contact: <sips:alice@93.12.40.105:20565;transport=ws>
   Content-Type: application/sdp
```

F2 100 Trying SIP Proxy -> Alice (transport WSS)

```
   SIP/2.0 100 Trying
   Via: SIP/2.0/WSS 93.12.40.105:20565;branch=z9hG4bK56sdasks
   From: sip:alice@atlanta.com;tag=asdyka899
   To: sip:carol@atlanta.com
   Call-ID: asidkj3ss
   CSeq: 1 INVITE
```

F3 INVITE SIP Proxy -> Carol (transport UDP)

```
   INVITE sip:carol@77.123.45.23:5060 SIP/2.0
   Via: SIP/2.0/UDP 100.100.100.100;branch=z9hG4bKhjhjqw32c
   Via: SIP/2.0/WSS 93.12.40.105:20565;branch=z9hG4bK56sdasks
   Record-Route: <sip:100.100.100.100;transport=udp>,
     <sips:100.100.100.100:9090;transport=ws>
   From: sip:alice@atlanta.com;tag=asdyka899
   To: sip:carol@atlanta.com
   Call-ID: asidkj3ss
   CSeq: 1 INVITE
   Max-Forwards: 69
   Contact: <sips:alice@93.12.40.105:20565;transport=ws>
   Content-Type: application/sdp
```

F4 200 OK Carol -> SIP Proxy (transport UDP)

```
   SIP/2.0 200 OK
   Via: SIP/2.0/UDP 100.100.100.100;branch=z9hG4bKhjhjqw32c
   Via: SIP/2.0/WSS 93.12.40.105:20565;branch=z9hG4bK56sdasks
   Record-Route: <sip:100.100.100.100;transport=udp>,
     <sips:100.100.100.100:9090;transport=ws>
   From: sip:alice@atlanta.com;tag=asdyka899
   To: sip:carol@atlanta.com;tag=bmqkjhsd
   Call-ID: asidkj3ss
   CSeq: 1 INVITE
   Max-Forwards: 69
```

```
   Contact: <sip:carol@77.123.45.23:5060;transport=udp>
   Content-Type: application/sdp
```

   F5 200 OK SIP Proxy -> Alice (transport WSS)

```
   SIP/2.0 200 OK
   Via: SIP/2.0/WSS 93.12.40.105:20565;branch=z9hG4bK56sdasks
   Record-Route: <sip:100.100.100.100;transport=udp>,
     <sips:100.100.100.100:9090;transport=ws>
   From: sip:alice@atlanta.com;tag=asdyka899
   To: sip:carol@atlanta.com;tag=bmqkjhsd
   Call-ID: asidkj3ss
   CSeq: 1 INVITE
   Max-Forwards: 69
   Contact: <sip:carol@77.123.45.23:5060;transport=udp>
   Content-Type: application/sdp
```

   F6 ACK Alice -> SIP Proxy (transport WSS)

```
   ACK sip:carol@77.123.45.23:5060;transport=udp SIP/2.0
   Via: SIP/2.0/WSS 93.12.40.105:20565;branch=z9hG4bKhgqqp090
   Route: <sips:100.100.100.100:9090;transport=ws>,
     <sip:100.100.100.100;transport=udp>
   From: sip:alice@atlanta.com;tag=asdyka899
   To: sip:carol@atlanta.com;tag=bmqkjhsd
   Call-ID: asidkj3ss
   CSeq: 1 ACK
   Max-Forwards: 70
```

   F7 ACK SIP Proxy -> Carol (transport UDP)

```
   ACK sip:carol@77.123.45.23:5060;transport=udp SIP/2.0
   Via: SIP/2.0/UDP 100.100.100.100;branch=z9hG4bKhwpoc80zzx
   Via: SIP/2.0/WSS 93.12.40.105:20565;branch=z9hG4bKhgqqp090
   From: sip:alice@atlanta.com;tag=asdyka899
   To: sip:carol@atlanta.com;tag=bmqkjhsd
   Call-ID: asidkj3ss
   CSeq: 1 ACK
   Max-Forwards: 69


   F8 BYE Carol -> SIP Proxy (transport UDP)

   BYE sips:alice@93.12.40.105:20565;transport=ws SIP/2.0
   Via: SIP/2.0/UDP 77.123.45.23;branch=z9hG4bKbiuiansd001
```

```
   Route: <sip:100.100.100.100;transport=udp>,
     <sips:100.100.100.100:9090;transport=ws>
   From: sip:carol@atlanta.com;tag=bmqkjhsd
   To: sip:alice@atlanta.com;tag=asdyka899
   Call-ID: asidkj3ss
   CSeq: 1201 BYE
   Max-Forwards: 70


   F9 BYE SIP Proxy -> Alice (transport WSS)

   BYE sips:alice@93.12.40.105:20565;transport=ws SIP/2.0
   Via: SIP/2.0/WSS 100.100.100.100:9090;branch=z9hG4bKmma01m3r5
   Via: SIP/2.0/UDP 77.123.45.23;branch=z9hG4bKbiuiansd001
   From: sip:carol@atlanta.com;tag=bmqkjhsd
   To: sip:alice@atlanta.com;tag=asdyka899
   Call-ID: asidkj3ss
   CSeq: 1201 BYE
   Max-Forwards: 69


   F10 200 OK Alice -> SIP Proxy (transport WSS)

   SIP/2.0 200 OK
   Via: SIP/2.0/WSS 100.100.100.100:9090;branch=z9hG4bKmma01m3r5
   Via: SIP/2.0/UDP 77.123.45.23;branch=z9hG4bKbiuiansd001
   From: sip:carol@atlanta.com;tag=bmqkjhsd
   To: sip:alice@atlanta.com;tag=asdyka899
   Call-ID: asidkj3ss
   CSeq: 1201 BYE


   F11 200 OK SIP Proxy -> Carol (transport UDP)
   SIP/2.0 200 OK
   Via: SIP/2.0/UDP 77.123.45.23;branch=z9hG4bKbiuiansd001
   From: sip:carol@atlanta.com;tag=bmqkjhsd
   To: sip:alice@atlanta.com;tag=asdyka899
   Call-ID: asidkj3ss
   CSeq: 1201 BYE
```

9.3. INVITE dialog through two proxies

```
   Alice    (SIP WS)    Proxy 1    (SIP UDP)   Proxy 2    (SIP WSS)     Bob
   |                     |                     |                     |
   |INVITE F1            |                     |                     |
   |-------------------->|                     |                     |
   |100 Trying F2        |                     |                     |
   |<--------------------|                     |                     |
   |                     |INVITE F3            |                     |
   |                     |-------------------->|                     |
   |                     |100 Trying F4        |                     |
   |                     |<--------------------|                     |
   |                     |                     |INVITE F5            |
   |                     |                     |-------------------->|
   |                     |                     |486 Busy Here F6     |
   |                     |                     |<--------------------|
   |                     |                     |ACK F7               |
   |                     |                     |-------------------->|
   |                     |486 Busy Here F8     |                     |
   |                     |<--------------------|                     |
   |                     |ACK F9               |                     |
   |                     |-------------------->|                     |
   |486 Busy Here F10    |                     |                     |
   |<--------------------|                     |                     |
   |ACK F11              |                     |                     |
   |-------------------->|                     |                     |
   |                     |                     |                     |
```

Alice and Bob are WebSocket clients running on web browsers. Alice belongs to atlanta.com SIP domain while Bob does to biloxi.com. Each domain has its own SIP proxy. Both proxies are also WebSocket servers. Alice calls Bob's AoR through a WebSocket connection. Bob responds the INVITE with a 486 Busy Here. Communication through proxies is made via UDP transport protocol.

Message details (authentication and SDP bodies are omitted for simplicity):

F1 INVITE Alice -> Proxy 1 (transport WS)

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/WS 93.12.40.105:21324;branch=z9hG4bKmmuuq
From: Alice <sip:alice@atlanta.com>;tag=lxtyr
To: Bob <sip:bob@biloxi.com>
Call-ID: aslke3dkj
CSeq: 1 INVITE
Max-Forwards: 70
Contact: <sip:alice@93.12.40.105:21324;transport=ws>
```

```
Content-Type: application/sdp
```

F2 100 Trying Proxy 1 -> Alice (transport WS)

```
SIP/2.0 100 Trying
Via: SIP/2.0/WS 93.12.40.105:21324;branch=z9hG4bKmmuuq
From: Alice <sip:alice@atlanta.com>;tag=lxtyr
To: Bob <sip:bob@biloxi.com>
Call-ID: aslke3dkj
CSeq: 1 INVITE
```

F3 INVITE Proxy 1 -> Proxy 2 (transport UDP)

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP 101.101.101.101;branch=z9hG4bKdkej
Via: SIP/2.0/WS 93.12.40.105:21324;branch=z9hG4bKmmuuq
Record-Route: <sip:101.101.101.101;transport=udp>
```

```
   Record-Route: <sip:101.101.101.101:80;transport=ws>
   From: Alice <sip:alice@atlanta.com>;tag=lxtyr
   To: Bob <sip:bob@biloxi.com>
   Call-ID: aslke3dkj
   CSeq: 1 INVITE
   Max-Forwards: 69
   Contact: <sip:alice@93.12.40.105:21324;transport=ws>
   Content-Type: application/sdp


   F4 100 Trying Proxy 2 -> Proxy 1 (transport UDP)

   SIP/2.0 100 Trying
   Via: SIP/2.0/UDP 101.101.101.101;branch=z9hG4bKdkej
   Via: SIP/2.0/WS 93.12.40.105:21324;branch=z9hG4bKmmuuq
   From: Alice <sip:alice@atlanta.com>;tag=lxtyr
   To: Bob <sip:bob@biloxi.com>
   Call-ID: aslke3dkj
   CSeq: 1 INVITE


   F5 INVITE Proxy 2 -> Bob (transport WSS)

   INVITE sips:bob@85.84.123.222:30142;transport=ws SIP/2.0
   Via: SIP/2.0/WSS 102.102.102.102:443;branch=z9hG4bKqowin
   Via: SIP/2.0/UDP 101.101.101.101;branch=z9hG4bKdkej
   Via: SIP/2.0/WS 93.12.40.105:21324;branch=z9hG4bKmmuuq
   Record-Route: <sips:102.102.102.102:443;transport=ws>
   Record-Route: <sip:102.102.102.102;transport=udp>
```

```
   Record-Route: <sip:101.101.101.101;transport=udp>
   Record-Route: <sip:101.101.101.101:9090;transport=ws>
   From: Alice <sip:alice@atlanta.com>;tag=lxtyr
   To: Bob <sip:bob@biloxi.com>
   Call-ID: aslke3dkj
   CSeq: 1 INVITE
   Max-Forwards: 68
   Contact: <sip:alice@93.12.40.105:21324;transport=ws>
   Content-Type: application/sdp


   F6 486 Busy Here Bob -> Proxy 2 (transport WSS)
```

```
SIP/2.0 486 Busy Here
Via: SIP/2.0/WSS 102.102.102.102:443;branch=z9hG4bKqowin
Via: SIP/2.0/UDP 101.101.101.101;branch=z9hG4bKdkej
Via: SIP/2.0/WS 93.12.40.105:21324;branch=z9hG4bKmmuuq
From: Alice <sip:alice@atlanta.com>;tag=lxtyr
To: Bob <sip:bob@biloxi.com>;tag=dskfjd
Call-ID: aslke3dkj
CSeq: 1 INVITE
```

F7 ACK Proxy 2 -> Bob (transport WSS)

```
ACK sips:bob@85.84.123.222:30142;transport=ws SIP/2.0
Via: SIP/2.0/WSS 102.102.102.102:443;branch=z9hG4bKqowin
From: Alice <sip:alice@atlanta.com>;tag=lxtyr
To: Bob <sip:bob@biloxi.com>;tag=dskfjd
Call-ID: aslke3dkj
CSeq: 1 ACK
```

F8 486 Busy Here Proxy 2 -> Proxy 1 (transport UDP)

```
SIP/2.0 486 Busy Here
Via: SIP/2.0/UDP 101.101.101.101;branch=z9hG4bKdkej
Via: SIP/2.0/WS 93.12.40.105:21324;branch=z9hG4bKmmuuq
From: Alice <sip:alice@atlanta.com>;tag=lxtyr
To: Bob <sip:bob@biloxi.com>;tag=dskfjd
Call-ID: aslke3dkj
CSeq: 1 INVITE
```

F9 ACK Proxy 1 -> Proxy 2 (transport UDP)

```
ACK sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP 101.101.101.101;branch=z9hG4bKdkej
```

```
From: Alice <sip:alice@atlanta.com>;tag=lxtyr
To: Bob <sip:bob@biloxi.com>;tag=dskfjd
Call-ID: aslke3dkj
CSeq: 1 ACK
```

F10 486 Busy Here Proxy 1 -> Alice (transport WS)

```
SIP/2.0 486 Busy Here
Via: SIP/2.0/WS 93.12.40.105:21324;branch=z9hG4bKmmuuq
From: Alice <sip:alice@atlanta.com>;tag=lxtyr
To: Bob <sip:bob@biloxi.com>;tag=dskfjd
Call-ID: aslke3dkj
CSeq: 1 INVITE
```

F11 ACK Alice -> Proxy 1 (transport WS)

```
ACK sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/WS 93.12.40.105:21324;branch=z9hG4bKmmuuq
From: Alice <sip:alice@atlanta.com>;tag=lxtyr
To: Bob <sip:bob@biloxi.com>;tag=dskfjd
Call-ID: aslke3dkj
CSeq: 1 ACK
```

10. Security Considerations

If the client (typically a web-based application) needs to protect the privacy of the SIP traffic through the WebSocket connection, it is encouraged to use a secure WebSocket connection.

11. IANA Considerations

11.1. Registration of new Via transports

This specification registers two new transport identifiers for Via headers:

WS: MUST be used when constructing a SIP request to be sent over a plain WebSocket connection.

WSS: MUST be used when constructing a SIP request to be sent over a secure WebSocket connection (tunneled over TLS [RFC5246]).

11.2. Registration of new SIP URI transport

This specification registers a new value for the "transport" parameter in a SIP URI:

ws: Identifies a SIP URI to be contacted using a WebSocket (plain or secure) connection.

11.3. Registration of the WebSocket SIP subprotocol

If a registry is created for WebSocket subprotocols, the SIP subprotocol defined in this specification will be registered.

12. References

12.1. Normative References

[I-D.ietf-hybi-thewebsocketprotocol]
Fette, I. and A. Melnikov, "The WebSocket protocol", draft-ietf-hybi-thewebsocketprotocol-14 (work in progress), September 2011.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3261] 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.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

12.2. Informative References

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3327] Willis, D. and B. Hoeneisen, "Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts", RFC 3327, December 2002.

[RFC3581] Rosenberg, J. and H. Schulzrinne, "An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing", RFC 3581, August 2003.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform

Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4168] Rosenberg, J., Schulzrinne, H., and G. Camarillo, "The Stream Control Transmission Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP)", RFC 4168, October 2005.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5658] Froment, T., Lebel, C., and B. Bonnaerens, "Addressing Record-Route Issues in the Session Initiation Protocol

(SIP)", RFC 5658, October 2009.

[RTC-Web] IETF and W3C, "Real Time Collaboration on the World Wide Web", October 2010.

[WS-API] Hickson, I., "The Web Sockets API", September 2010.

Authors' Addresses

Inaki Baz Castillo
XtraTelecom S.A.
Barakaldo, Basque Country
Spain

Email: ibc@aliax.net

Jose Luis Millan
XtraTelecom S.A.
Bilbao, Basque Country
Spain

Email: jmillan@aliax.net

Victor Pascual
Acme Packet
Anabel Segura 10

Madrid, Madrid 28108
Spain

Email: vpascual@acmepacket.com