BFCPBIS Working Group
Internet-Draft
Updates: rfc4582bis, rfc4583bis (if approved)
Intended status: Standards Track
Expires: April 15, 2016

V. Pascual
Oracle
A. Roman
Quobis
S. Cazeaux
France Telecom Orange
G. Salgueiro
R. Ravindranath
Cisco
S. Garcia Murillo
Medooze

October 13, 2015

The WebSocket Protocol as a Transport for the Binary Floor Control Protocol (BFCP)
draft-ietf-bfcpbis-bfcp-websocket-05

Abstract

The WebSocket protocol enables two-way realtime communication between clients and servers. This document specifies a new WebSocket sub-protocol as a reliable transport mechanism between Binary Floor Control Protocol (BFCP) entities to enable usage of BFCP in new scenarios.

Table of Contents

1. Introduction

The WebSocket [RFC6455] protocol enables two-way 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 Hypertext Transfer

Protocol (HTTP) [RFC2616] semantics, allowing the WebSocket protocol to reuse existing HTTP infrastructure.

The Binary Floor Control Protocol (BFCP) is a protocol to coordinate access to shared resources in a conference. It is defined in [I-D.ietf-bfcpbis-rfc4582bis] and is used between floor participants and floor control servers, and between floor chairs (i.e., moderators) and floor control servers.

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. This document updates [I-D.ietf-bfcpbis-rfc4582bis] and [I-D.ietf-bfcpbis-rfc4583bis] in order to enable the usage of BFCP in these scenarios.

The transport over which BFCP entities exchange messages depends on how the clients obtain information to contact the floor control server (e.g. using an Session Description Protocol (SDP) offer/answer exchange per [I-D.ietf-bfcpbis-rfc4583bis] or the procedure described in RFC5018 [RFC5018]). [I-D.ietf-bfcpbis-rfc4582bis] defines two transports for BFCP: TCP and UDP. This specification defines a new WebSocket sub-protocol (as defined in section 1.9 in [RFC6455]) for transporting BFCP messages between a WebSocket client and server. This sub-protocol provides a reliable and boundary preserving transport for BFCP when run on top of TCP. Since WebSocket provides a reliable transport, the extensions defined in [I-D.ietf-bfcpbis-rfc4582bis] for sending BFCP over unreliable transports are not applicable. This document normatively updates [I-D.ietf-bfcpbis-rfc4582bis] and [I-D.ietf-bfcpbis-rfc4583bis].

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

BFCP WebSocket Client: Any BFCP entity capable of opening outbound connections to WebSocket servers and communicating using the WebSocket BFCP sub-protocol as defined by this document.

BFCP WebSocket Server: Any BFCP entity capable of listening for inbound connections from WebSocket clients and communicating using the WebSocket BFCP sub-protocol as defined by this document.

3. The WebSocket Protocol

The WebSocket protocol [RFC6455] is a transport layer on top of TCP (optionally secured with TLS [RFC5246]) in which both client and server exchange message units in both directions. The protocol defines a connection handshake, WebSocket sub-protocol 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 [RFC2616] 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 an application protocol (also known as a "WebSocket sub-protocol") defines the format and semantics of the messages exchanged by the endpoints. This could be a custom protocol or a standardized one (as the WebSocket BFCP sub-protocol 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.

The WebSocket protocol 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 BFCP Sub-Protocol

The term WebSocket sub-protocol refers to an application-level protocol layered on top of a WebSocket connection. This document specifies the WebSocket BFCP sub-protocol for carrying BFCP messages over a WebSocket connection.

4.1. Handshake

The BFCP WebSocket Client and BFCP WebSocket Server negotiate usage of the WebSocket BFCP sub-protocol during the WebSocket handshake procedure as defined in section 1.3 of [RFC6455]. The Client MUST include the value "bfcp" in the Sec-WebSocket-Protocol header in its handshake request. The 101 reply from the Server MUST contain "bfcp" in its corresponding Sec-WebSocket-Protocol header.

Below is an example of a WebSocket handshake in which the Client requests the WebSocket BFCP sub-protocol support from the Server:

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

The handshake response from the Server accepting the WebSocket BFCP sub-protocol would look as follows:

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

Once the negotiation has been completed, the WebSocket connection is established and can be used for the transport of BFCP messages. The WebSocket messages transmitted over this connection MUST conform to the negotiated WebSocket sub-protocol.

4.2. BFCP Encoding

BFCP messages use a TLV (Type-Length-Value) binary encoding, therefore BFCP WebSocket Clients and BFCP WebSocket Servers MUST be transported in unfragmented binary WebSocket frames (FIN:1,opcode:%x2) to exchange BFCP messages. The WebSocket frame data MUST be a valid BCFP message, so the length of the payload of the WebSocket frame MUST be lower than the maximum size allowed (2^16 +12 bytes) for a BCFP message as described in [I-D.ietf-bfcpbis-rfc4582bis]. In addition, the encoding rules for reliable protocols defined in [I-D.ietf-bfcpbis-rfc4582bis] MUST be followed.

While this specification assumes that BFCP encoding is only TLV binary, future documents may define other mechanisms like JSON serialization.

5. Transport Reliability

WebSocket [RFC6455] provides a reliable transport and therefore the BFCP WebSocket sub-protocol defined by this document also provides reliable BFCP transport. Thus, client and server transactions using WebSocket for transport MUST follow the procedures for reliable transports as defined in [I-D.ietf-bfcpbis-rfc4582bis] and [I-D.ietf-bfcpbis-rfc4583bis]

BFCP WebSocket clients cannot receive incoming WebSocket connections initiated by any other peer. This means that a BFCP WebSocket client MUST actively initiate a connection towards a BFCP WebSocket server

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

6. SDP Considerations

6.1. Updates to RFC4583bis

Rules to generate an 'm' line for a BFCP stream are described in [I-D.ietf-bfcpbis-rfc4583bis], Section 3

New values are defined for the transport field: TCP/WS/BFCP and TCP/WSS/BFCP.

TCP/WS/BFCP is used when BFCP runs on top of WS, which in turn runs on top of TCP.

TCP/WSS/BFCP is used when BFCP runs on top of WSS, which in turn runs on top of TLS and TCP.

6.2. Updates to RFC4582bis

When TCP is used as the transport, the port field is set following the rules in Section 7 of [I-D.ietf-bfcpbis-rfc4582bis]. Depending on the value of the SDP 'setup' attribute defined in [RFC4145], the port field contains the port to which the remote endpoint will direct BFCP messages or is irrelevant (i.e., the endpoint will initiate the connection towards the remote endpoint) and should be set to a value of 9, which is the discard port. Connection attribute and port MUST follow the rules of [RFC4145]

Some web browsers do not allow non-secure WebSocket connections to be made. So, while the recommendation to use Secure WebSockets (i.e. TCP/WSS) is for security reasons, it is also to achieve maximum compatibility among clients.

6.3. SDP Media Attributes

[I-D.ram-bfcpbis-sdp-ws-uri] defines a new SDP attributes to indicate the connection Uniform Resource Identifier (URI) for the WebSocket Client. The SDP attribute 'ws-uri' defined in section 3.1 of [I-D.ram-bfcpbis-sdp-ws-uri] MUST be used when BFCP runs on top of WS, which in turn runs on top of TCP. The SDP attribute 'wss-uri' defined in section 3.2 of [I-D.ram-bfcpbis-sdp-ws-uri] MUST be used when BFCP runs on top of WSS, which in turn runs on top of TLS and TCP. When the 'ws-uri' or 'wss-uri' attribute is present in the media section of the SDP, the IP and port information provided in the 'c' lines SHALL be ignored and the full URI SHALL be used instead to open the WebSocket connection. The port provided in the 'm' line SHALL be ignored too, as the a=ws-uri or a=wss-uri SHALL provide port number when needed.

7. SDP Offer/Answer Procedures

7.1. General

An endpoint (i.e., both the offerer and the answerer) MUST create an SDP media description ("m=" line) for each BFCP-over-WebSocket media stream and MUST assign a TCP/WSS/BFCP value to the "proto" field of the "m=" line. Furthermore, the SDP answerer (Server) MUST add an "a=ws-uri" or "a=wss-uri" attribute in the "m=" line of each BFCP-over-WebSocket media stream depending on whether it is WS or WSS. This new attribute MUST follow the syntax defined in [I-D.ram-bfcpbis-sdp-ws-uri]. The procedures in this section apply to an "m=" line associated with a BFCP-over-WebSocket media stream.

7.2. Generating the Initial Offer

An SDP offerer in order to negotiate BFCP-over-WebSocket MUST generate an "m=" line which has:

- The SDP attributes as defined in Section 4 of [I-D.ietf-bfcpbis-rfc4583bis]

- The "proto" value in the "m=" line MUST be TCP/WSS/BFCP if WebSocket is over TLS, else it MUST be TCP/WS/BFCP.

The offerer SHOULD assign the SDP "setup" attribute with a value of "active" (the offerer will be the initiator of the outgoing TCP

connection), unless the offerer insists on being a receiver of an incoming connection, in which case the offerer SHOULD use a value of "passive". The offerer MUST NOT assign an SDP "setup" attribute with a "holdconn" value. If the offerer assigns the SDP "setup" attribute with a value of "passive", the offerer MUST be prepared to receive an incoming TCP connection on the IP and port tuple advertised in the "c=" line and audio/video ports of the BFCP media stream before it receives the SDP answer.

The following is an example of an "m=" line for a BFCP connection:

```
Offer (browser):
m=application 9 TCP/WSS/BFCP *
a=setup:active
a=connection:new
a=floorctrl:c-only
m=audio 55000 RTP/AVP 0
m=video 55002 RTP/AVP 31
```

In the above example, the client is intending to setup the TLS /TCP connection and hence the port is set to a value of 9, which is the discard port.

7.3. Generating the Answer

If the answerer accepts the offered BFCP-over-WebSocket transport connection, in the associated SDP answer, the answerer MUST assign an SDP "setup" attribute with a value of either "active" or "passive", according to the procedures in [RFC4145]. The answerer MUST NOT assign an SDP "setup" attribute with a value of "holdconn".

If the answerer assigns an SDP "setup" attribute with a value of "active", the answerer MUST initiate the WebSocket connection handshake by acting as client on the negotiated media stream, towards the IP address and port of the offerer using the procedures described in [RFC6455]. Apart from the SDP attributes of the BFCP media stream, the answer MUST have an "a=ws-uri" or "a=wss-uri" attribute depending on whether BFCP is running over WS or WSS. This attribute MUST follow the syntax defined in [I-D.ram-bfcpbis-sdp-ws-uri]. The "proto" value in the "m=" line MUST be TCP/WSS/BFCP if WebSocket is run on TLS, else it MUST be TCP/WS/BFCP.

The following example shows a case where the server responds with a BFCP media stream over a WebSocket connection running TLS. It shows an answer "m=" line for the BFCP connection. In this example since WebSockets is running over TLS, the server answers back with "a=wss-uri" attribute in SDP indicating the connection URI:

```
Answer (server):
m=application 50000 TCP/WSS/BFCP *
a=setup:passive
a=connection:new
a=wss-uri:wss://bfcp-ws.example.com?token=3170449312
a=floorctrl:s-only
a=confid:4321
a=userid:1234
a=floorid:1 m-stream:10
a=floorid:2 m-stream:11
m=audio 50002 RTP/AVP 0
a=label:10
m=video 50004 RTP/AVP 31
a=label:11
```

7.4. Offerer Processing of the Answer

When the offerer receives an SDP answer, if the offerer ends up being active it MUST initiate the WebSocket connection handshake by sending a GET message on the negotiated media stream, towards the IP address and port of the answerer, as per the procedures described in [RFC6455].

7.5. Modifying the Session

Once an offer/answer exchange has been completed, either endpoint MAY send a new offer in order to modify the session. The endpoints can reuse the existing WebSocket connection if the ws-uri values and the transport parameters indicated by each endpoint are unchanged. Otherwise, following the rules for the initial offer/answer exchange, the endpoints can negotiate and create a new WebSocket connection on top of TLS/TCP or TCP.

8. Authentication

Section 9 of [I-D.ietf-bfcpbis-rfc4582bis] states that BFCP clients and floor control servers SHOULD authenticate each other prior to accepting messages, and RECOMMENDS that mutual TLS/DTLS authentication be used. However, browser-based WebSocket clients have no control over the use of TLS in the WebSocket API [WS-API], so it is RECOMMENDED that standard Web-based methods for client and server authentication are used, as follows.

When a BFCP WebSocket client connects to a BFCP WebSocket server, it SHOULD use TCP/WSS as its transport. The WebSocket client SHOULD inspect the TLS certificate offered by the server and verify that it is valid.

> Since the WebSocket API does not distinguish between certificate errors and other kinds of failure to establish a connection, it is expected that browser vendors will warn end users directly of any kind of problem with the server certificate.

A floor control server that receives a message over TCP/WS can request the use of TCP/WSS by generating an Error message, as described in Section 13.8 of [I-D.ietf-bfcpbis-rfc4582bis], with an Error code with a value of 9 (use TLS).

Prior to sending BFCP requests, a BFCP WebSocket client connects to a BFCP 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 BFCP WebSocket server MAY inspect any cookie [RFC6265] headers present 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 BFCP 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:

> If the status code received from the server is not 101, the WebSocket client stack handles the response per HTTP [RFC2616] procedures, in particular the client might perform authentication if it receives 401 status code.

9. Security Considerations

Considerations from [I-D.ietf-bfcpbis-rfc4582bis], [I-D.ietf-bfcpbis-rfc4583bis] and RFC5018 [RFC5018] apply.

BFCP relies on lower-layer security mechanisms to provide replay and integrity protection and confidentiality. It is RECOMMENDED that the BFCP traffic transported over a WebSocket communication be protected by using a secure WebSocket connection (using TLS [RFC5246] over TCP).

10. IANA Considerations

10.1. Registration of the WebSocket BFCP Sub-Protocol

This specification requests IANA to register the WebSocket BFCP sub-protocol under the "WebSocket Subprotocol Name" Registry with the following data:

Subprotocol Identifier: bfcp

Subprotocol Common Name: WebSocket Transport for BFCP (Binary Floor Control Protocol)

Subprotocol Definition: RFCXXXX

[[NOTE TO RFC EDITOR: Please change XXXX to the number assigned to this specification, and remove this paragraph on publication.]]

10.2. Registration of the 'TCP/WS/BFCP' and 'TCP/WSS/BFCP' SDP 'proto' Values

This document defines two new values for the SDP 'proto' field under the Session Description Protocol (SDP) Parameters registry. The resulting entries are shown in Figure 1 below:

Value	Reference
TCP/WS/BFCP	RFC&rfc.number;
TCP/WSS/BFCP	RFC&rfc.number;

Figure 1: Values for the SDP 'proto' field

[[NOTE TO RFC EDITOR: Please change XXXX to the number assigned to this specification, and remove this paragraph on publication.]]

11. Acknowledgements

The authors want to thank Robert Welbourn, from Acme Packet, who made significant contributions to the first version of this document. This work benefited from the thorough review and constructive comments of Charles Eckel and Christer Holmberg.

12. References

12.1. Normative References

[I-D.ietf-bfcpbis-rfc4582bis]
Camarillo, G., Drage, K., Kristensen, T., Ott, J., and C. Eckel, "The Binary Floor Control Protocol (BFCP)", draft-ietf-bfcpbis-rfc4582bis-14 (work in progress), September 2015.

[I-D.ietf-bfcpbis-rfc4583bis]
Camarillo, G., Kristensen, T., and P. Jones, "Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams", draft-ietf-bfcpbis-rfc4583bis-12 (work in progress), September 2015.

[I-D.ram-bfcpbis-sdp-ws-uri]
R, R. and G. Salgueiro, "Session Description Protocol (SDP) WebSocket Connection URI Attribute", draft-ram-bfcpbis-sdp-ws-uri-00 (work in progress), March 2015.

[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>.

[RFC4145] Yon, D. and G. Camarillo, "TCP-Based Media Transport in the Session Description Protocol (SDP)", RFC 4145, DOI 10.17487/RFC4145, September 2005, <http://www.rfc-editor.org/info/rfc4145>.

[RFC5018] Camarillo, G., "Connection Establishment in the Binary Floor Control Protocol (BFCP)", RFC 5018, DOI 10.17487/RFC5018, September 2007, <http://www.rfc-editor.org/info/rfc5018>.

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

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, DOI 10.17487/RFC2616, June 1999, <http://www.rfc-editor.org/info/rfc2616>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[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>.

[WS-API] W3C and I. Hickson, Ed., "The WebSocket API", May 2012.

Authors' Addresses

Victor Pascual
Oracle

Email: victor.pascual.avila@oracle.com

Anton Roman
Quobis

Email: anton.roman@quobis.com

Stephane Cazeaux
France Telecom Orange

Email: stephane.cazeaux@orange.com

Gonzalo Salgueiro
Cisco Systems, Inc.
7200-12 Kit Creek Road
Research Triangle Park, NC 27709
US

Email: gsalguei@cisco.com

Ram Mohan Ravindranath
Cisco Systems, Inc.
Cessna Business Park,
Kadabeesanahalli Village, Varthur Hobli,
Sarjapur-Marathahalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: rmohanr@cisco.com

Sergio Garcia Murillo
Medooze

Email: sergio.garcia.murillo@gmail.com