Internet Engineering Task Force (IETF)
Request for Comments: 8857
Category: Standards Track
ISSN: 2070-1721

V. Pascual
Nokia
A. Román
Quobis
S. Cazeaux
Orange
G. Salgueiro
R. Ravindranath
Cisco

January 2021

The WebSocket Protocol as a Transport for the Binary Floor Control Protocol (BFCP)

Abstract

The WebSocket protocol enables two-way real-time communication between clients and servers. This document specifies the use of Binary Floor Control Protocol (BFCP) as a new WebSocket subprotocol enabling a reliable transport mechanism between BFCP entities in new scenarios.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8857.

Table of Contents

1. Introduction

The WebSocket (WS) protocol [RFC6455] enables two-way message exchange between clients and servers on top of a persistent TCP connection, optionally secured with Transport Layer Security (TLS) [RFC8446]. The initial protocol handshake makes use of Hypertext Transfer Protocol (HTTP) [RFC7230] 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 [RFC8855] 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 extends the applicability of [RFC8855] and [RFC8856] 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 a Session Description Protocol (SDP) offer/answer exchange per [RFC8856] or the procedure described in RFC 5018 [RFC5018]). [RFC8855] defines two transports for BFCP: TCP and UDP. This specification defines a new WebSocket subprotocol (as defined in Section 1.9 of [RFC6455]) for transporting BFCP messages between a WebSocket client and server. This subprotocol 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 [RFC8855] for sending BFCP over unreliable transports are not applicable.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.1. Definitions

BFCP WebSocket Client: Any BFCP entity capable of opening outbound connections to WebSocket servers and communicating using the WebSocket BFCP subprotocol 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 subprotocol 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 [RFC8446]) in which both 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 header field. 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, the 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 subprotocol") 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 subprotocol defined in this document). Once the HTTP 101 response is processed, both the 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.

4. The WebSocket BFCP Subprotocol

The term WebSocket subprotocol refers to an application-level protocol layered on top of a WebSocket connection. This document specifies the WebSocket BFCP subprotocol 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 subprotocol 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 field in its handshake request. The 101 reply from the server MUST contain "bfcp" in its corresponding Sec-WebSocket-Protocol header field.

Below is an example of a WebSocket handshake in which the client requests the WebSocket BFCP subprotocol 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 subprotocol 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.

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 BFCP 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 BFCP message as described in [RFC8855]. In addition, the encoding rules for reliable protocols defined in [RFC8855] MUST be followed.

While this specification assumes that BFCP encoding is only TLV binary, future documents may define other mechanisms, like JSON serialization. If encoding changes, a new subprotocol identifier would need to be selected.

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

5. Transport Reliability

The WebSocket protocol [RFC6455] provides a reliable transport, and therefore the BFCP WebSocket subprotocol defined by this document also provides reliable BFCP transport. Thus, client and server transactions using the WebSocket protocol for transport MUST follow the procedures for reliable transports as defined in [RFC8855] and [RFC8856].

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. The BFCP server will have a globally routable address and thus does not require ICE, as clients always initiate connections to it.

6. SDP Considerations

6.1. Transport Negotiation

Rules to generate an "m=" line for a BFCP stream are described in [RFC8856], Section 4.

New values are defined for the SDP "proto" 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 secure WebSocket (WSS), which in turn runs on top of TLS and TCP.

The "port" field is set following the rules in Section 4 and Section 7.1 of [RFC8856]. 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 it 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. The 'connection' attribute and port MUST follow the rules of [RFC4145].

While this document recommends the use of secure WebSocket (i.e., TCP/WSS) for security reasons, TCP/WS is also permitted so as to achieve maximum compatibility among clients.

6.2. SDP Media Attributes

[RFC8124] defines a new SDP attribute to indicate the connection Uniform Resource Identifier (URI) for the WebSocket client. The SDP attribute 'websocket-uri' defined in Section 3 of [RFC8124] MUST be used when BFCP runs on top of WS or WSS. When the 'websocket-uri' attribute is present in the media section of the SDP, the procedures mentioned in Section 4 of [RFC8124] MUST be followed.

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 either a 'TCP/WSS/BFCP' or 'TCP/WS/BFCP' value to the "proto" field of the "m=" line depending on whether the endpoint wishes to use secure WebSocket or WebSocket. Furthermore, the server side, which could be either the offerer or answerer, MUST add a 'websocket-uri' attribute in the media section depending on whether it wishes to use WebSocket or secure WebSocket. This new attribute MUST follow the syntax defined in [RFC8124]. Additionally, the SDP offer/answer procedures defined in Section 4 of [RFC8124] MUST be followed for the "m=" line associated with a BFCP-over-WebSocket media stream.

7.2. Example Usage of 'websocket-uri' SDP Attribute

The following is an example of an "m=" line for a BFCP connection. In this example, the offerer sends the SDP with the "proto" field having a value of 'TCP/WSS/BFCP', indicating that the offerer wishes to use secure WebSocket as a transport for the media stream, and the "fmt" field having a value of '*' (for details on the "fmt" field, see Section 4 of [RFC8856]).

```
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

Answer (server):
m=application 50000 TCP/WSS/BFCP *
a=setup:passive
a=connection:new
a=websocket-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
```

It is possible that an endpoint (e.g., a browser) sends an offerless INVITE to the server. In such cases, the server will act as SDP offerer. The server MUST assign the SDP 'setup' attribute with a value of 'passive'. The server MUST have a 'websocket-uri' attribute with a 'ws-URI' or 'wss-URI' value depending on whether the server wishes to use WebSocket or secure WebSocket. This attribute MUST follow the syntax defined in Section 3 of [RFC8124]. For BFCP

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

8. Authentication

Section 9 of [RFC8855] 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. If the signaling or control protocol traffic used to set up the conference is authenticated and confidentiality and integrity protected, secure WebSocket (WSS) MUST be used, and the floor control server MUST authenticate the client. The WebSocket client MUST follow the procedures in [RFC7525] while setting up TLS connection with the WebSocket server. The BFCP client validates the server by means of verifying the server certificate. This means the 'websocket-uri' value MUST contain a hostname. The verification process does not use "a=fingerprint".

A floor control server that receives a message over TCP/WS can mandate the use of TCP/WSS by generating an Error message, as described in Section 13.8 of [RFC8855], 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 4.1, the handshake procedure involves an 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 SHOULD inspect any cookie header fields [RFC6265] 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 an 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 [RFC7230] procedures; in particular, the client might perform authentication if it receives an 401 status code. The WebSocket clients are vulnerable to the attacks of basic authentication (mentioned in Section 4 of [RFC7617]) and digest authentication (mentioned in Section 5 of [RFC7616]). To overcome some of these weaknesses, WebSocket clients can use the HTTP Origin-Bound Authentication (HOBA) mechanism mentioned in [RFC7486], for example.

9. Security Considerations

Considerations from [RFC8855], [RFC8856], and [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 WebSocket be protected by using a Secure WebSocket connection (using TLS [RFC8446] over TCP). The security considerations in [RFC6455] apply for BFCP over WebSocket as well. The security model here is a typical webserver-client model where the client validates the server certificate and then connects to the server. Section 8 describes the authentication procedures between client and server.

When using BFCP over WebSocket, the security mechanisms defined in [RFC8855] are not used. Instead, the application is required to build and rely on the security mechanisms in [RFC6455].

The rest of this section analyses the threats described in Section 14 of [RFC8855] when WebSocket is used as a transport protocol for BFCP.

An attacker attempting to impersonate a floor control server is avoided by having servers accept BFCP messages over WSS only. As with any other web connection, the clients will verify the server's certificate. The BFCP WebSocket client MUST follow the procedures in [RFC7525] (including hostname verification as per Section 6.1 of [RFC7525]) while setting up a TLS connection with floor control WebSocket server.

An attacker attempting to impersonate a floor control client is avoided by having servers accept BFCP messages over WSS only. As described in Section 10.5 of [RFC6455] the floor control server can use any client authentication mechanism and follow the steps in Section 8 of this document.

Attackers may attempt to modify messages exchanged by a client and a floor control server. This can be prevented by having WSS between client and server.

An attacker trying to replay the messages is prevented by having floor control servers check that messages arriving over a given WSS connection use an authorized user ID.

Attackers may eavesdrop on the network to get access to confidential information between the floor control server and a client (e.g., why a floor request was denied). In order to ensure that BFCP users are getting the level of protection that they would get using BFCP directly, applications need to have a way to control the WebSocket libraries to use encryption algorithms specified in Section 7 of [RFC8855]. Since the WebSocket API [WS-API] does not have a way to allow an application to select the encryption algorithm to be used, the protection level provided when WSS is used is limited to the underlying TLS algorithm used by the WebSocket library.

10. IANA Considerations

10.1. Registration of the WebSocket BFCP Subprotocol

IANA has registered the WebSocket BFCP subprotocol under the "WebSocket Subprotocol Name Registry" as follows:

Subprotocol Identifier: bfcp

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

Subprotocol Definition: RFC 8857

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" subregistry within the "Session Description Protocol (SDP) Parameters" registry. The resulting entries are shown in Table 1:

Value	Reference
TCP/WS/BFCP	RFC 8857
TCP/WSS/BFCP	RFC 8857

Table 1: Values for the SDP "proto" Field

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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc5018>.

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

[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, <https://www.rfc-editor.org/info/rfc7525>.

[RFC8124] Ravindranath, R. and G. Salgueiro, "The Session Description Protocol (SDP) WebSocket Connection URI Attribute", RFC 8124, DOI 10.17487/RFC8124, March 2017, <https://www.rfc-editor.org/info/rfc8124>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8855] Camarillo, G., Drage, K., Kristensen, T., Ott, J., and C. Eckel, "The Binary Floor Control Protocol (BFCP)", RFC 8855, DOI 10.17487/RFC8855, January 2021, <https://www.rfc-editor.org/info/rfc8855>.

[RFC8856] Camarillo, G., Kristensen, T., and C. Holmberg, "Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams", RFC 8856, DOI 10.17487/RFC8856, January 2021, <https://www.rfc-editor.org/info/rfc8856>.

11.2. Informative References

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

[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, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7486] Farrell, S., Hoffman, P., and M. Thomas, "HTTP Origin-Bound Authentication (HOBA)", RFC 7486, DOI 10.17487/RFC7486, March 2015, <https://www.rfc-editor.org/info/rfc7486>.

[RFC7616] Shekh-Yusef, R., Ed., Ahrens, D., and S. Bremer, "HTTP Digest Access Authentication", RFC 7616, DOI 10.17487/RFC7616, September 2015, <https://www.rfc-editor.org/info/rfc7616>.

[RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <https://www.rfc-editor.org/info/rfc7617>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

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

Acknowledgements

The authors want to thank Robert Welbourn from Acme Packet and Sergio Garcia Murillo, who made significant contributions to the first draft version of this document. This work benefited from the thorough review and constructive comments of Charles Eckel, Christer Holmberg, Paul Kyzivat, Dan Wing, and Alissa Cooper. Thanks to Bert Wijnen, Robert Sparks, and Mirja Kühlewind for their reviews and comments on this document.

Thanks to Spencer Dawkins, Ben Campbell, Kathleen Moriarty, Alexey Melnikov, Jari Arkko, and Stephen Farrell for their feedback and comments during IESG reviews.

Authors' Addresses

Victor Pascual
Nokia
Barcelona
Spain

Email: victor.pascual_avila@nokia.com

Antón Román
Quobis
Pol. Ind. A Granxa, Casa de Pedra
36475 O Porriño
Spain

Email: anton.roman@quobis.com

Stéphane Cazeaux
Orange
42 rue des Coutures
14000 Caen
France

Email: stephane.cazeaux@orange.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.
Cessna Business Park
Kadabeesanahalli Village, Varthur Hobli,
Sarjapur-Marathahalli Outer Ring Road
Bangalore 560103
Karnataka

India

Email: rmohanr@cisco.com