The WebSocket Protocol as a Transport for the Remote Framebuffer Protocol (RFB)
draft-garnero-rfb-websocket-00

Abstract

The WebSocket protocol enables two-way realtime communication between clients and servers in web-based applications. This document specifies a WebSocket sub-protocol as a reliable transport mechanism between RFB (Remote Framebuffer Protocol) entities to enable usage of RFB in web-oriented deployments.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 31, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

The WebSocket [RFC6455] protocol enables message exchange between clients and servers on top of a persistent TCP connection (optionally secured with TLS [RFC5246]). The initial protocol handshake makes use of HTTP [RFC2616] 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 RFB in these scenarios.
This specification defines a WebSocket sub-protocol (as defined in section 1.9 in [RFC6455]) for transporting RFB messages between a WebSocket client and server, a reliable and (as far as possible) message-boundary preserving transport for RFB and procedures for RFB entities implementing the WebSocket transport.

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

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

RFB WebSocket Server: A RFB entity capable of listening for inbound connections from WebSocket clients and communicating using the WebSocket RFB 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 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 RFB 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.

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 RFB 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 RFB sub-protocol for carrying RFB messages through a WebSocket connection.

4.1. Handshake

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

Below is an example of a WebSocket handshake in which the Client requests the WebSocket RFB sub-protocol support from the Server:
GET / HTTP/1.1
Host: rfb-ws.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBzeXN0ZWQ=
Origin: http://www.example.com
Sec-WebSocket-Protocol: rfb
Sec-WebSocket-Version: 13

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

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

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

4.2. RFB Encoding

WebSocket messages can be transported in either UTF-8 text frames or binary frames. RFB [RFC6143] defines binary bodies in RFB messages. Therefore RFB WebSocket Clients and RFB WebSocket Servers MUST accept binary frames and MUST NOT accept text frames.

Encoding of RFB messages over WebSocket transport remains identical to the encoding defined in [RFC6143]

5. RFB WebSocket Transport

5.1. General

WebSocket [RFC6455] is a reliable protocol and therefore the RFB WebSocket sub-protocol defined by this document is a reliable RFB transport.

Each RFB message, excepted FramebufferUpdate message, MUST be carried within a single WebSocket message, and a WebSocket message MUST NOT contain more than one RFB message.
Because the WebSocket transport preserves message boundaries, this simplifies decoding of (most) RFB messages for both clients and servers.

However, FramebufferUpdate message SHOULD be handled differently for performance reasons.

5.2. FramebufferUpdate Server-to-Client message handling

According to [RFC6143] section 7.6.1 "FramebufferUpdate", a framebuffer update consists of a sequence of rectangles of pixel data that the client should put into its framebuffer. Therefore a RFB Websocket Server should have to transfer a certain number of rectangles of pixel data to a remote RFB Websocket Client. The amount of data could be quite large depending on the number of rectangles, the number of pixel in each rectangle and the encoding of the pixels. It could make sense for the RFB Websocket Server to avoid to wait to gather the full amount of rectangles before starting to transfer the data to the remote RFB Websocket Client. That way, the remote RFB Websocket Client could start to process "small" parts of the framebuffer update in a timely manner instead of processing the full amount of data in one time, which could be a really CPU intensive process. Then, preservation of message boundaries is no more required for this RFB message and such an RFB message could be transferred by mean of several WebSocket messages.

Of course, due to the fact that [RFC6143] define a TCP transport, the nature of the Remote Framebuffer Protocol let it easily support a non preservation of message boundaries.

However, the RFB Websocket Server should respect the following principles:

- The RFB Websocket Server SHOULD elect to send an RFB FramebufferUpdate in a single WebSocket message
- Or the RFB Websocket Server SHOULD elect to send an RFB FramebufferUpdate in multiple WebSocket messages. In that case:

  It MUST send the RFB FramebufferUpdate content in a set of consecutive WebSocket messages. That is, It MUST NOT transfer any other RFB message until the RFB FramebufferUpdate full message is transfered.

  It MUST send the RFB FramebufferUpdate header (see [RFC6143] section 7.6.1 "FramebufferUpdate") in a single WebSocket
message. This WebSocket message should be the first message related to the transfer of the full RFB FramebufferUpdate content. The RFB FramebufferUpdate header should be alone in the WebSocket message, without any rectangle information.

It MUST send RFB FramebufferUpdate rectangle header ([RFC6143] section 7.6.1 "FramebufferUpdate") in a single WebSocket message, potentially with a part or all of the pixel data associated to this rectangle.

It MUST send rectangle pixel data in a single WebSocket message or several consecutive WebSocket messages.

It MUST NOT mix information related to different rectangles in the same WebSocket message.

6. Connection Keep-Alive

RFB WebSocket Clients and Servers may keep their WebSocket connections open by sending periodic WebSocket "Ping" frames as described in [RFC6455] section 5.5.2.

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 keep-alive feature is the decision of each web browser manufacturer and may also depend on the configuration of the web browser.

7. Authentication

Authentication process is fully described in [RFC6143] and remains valid if WebSocket transport is used.

8. Security Considerations

8.1. Secure WebSocket Connection

It is recommended that the RFB traffic transported over a WebSocket communication be protected by using a secure WebSocket connection (using TLS [RFC5246] over TCP).

When establishing a connection using RFB over secure WebSocket transport, the client MUST authenticate the server using the server’s certificate according to the WebSocket validation procedure in [RFC6455].
9. IANA Considerations

RFC Editor Note: Please set the RFC number assigned for this document in the sub-sections below and remove this note.

9.1. Registration of the WebSocket RFB Sub-Protocol

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

Subprotocol Identifier: rfb

Subprotocol Common Name: WebSocket Transport for RFB (Remote Framebuffer Protocol)

Subprotocol Definition: Included in this document.

10. References

10.1. Normative References


10.2. Informative References


Author’s Address

Pierre Garnero
Villamicro
Grasse 06130
France

Email: pierre.garnero@villamicro.fr