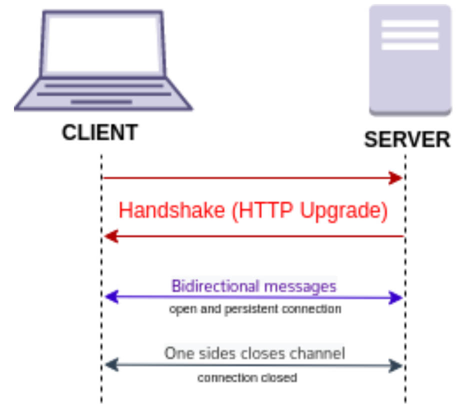
WIKIPEDIA
The Free Encyclopedia

WebSocket

WebSocket



A diagram describing a connection using WebSocket

International standard	RFC 6455 (https://datatracker.ietf.org/doc/html/rfc6455)
Developed by	IETF
Industry	Computer science
Connector type	TCP
Website	Official website (https://datatracker.ietf.org/doc/html/rfc6455)

WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 (https://datatracker.ietf.org/doc/html/rfc6455) in 2011. The current API specification allowing web applications to use this protocol is known as *WebSockets*.[1] It is a living standard maintained by the WHATWG and a successor to *The WebSocket API* from the W3C.[2]

WebSocket is distinct from HTTP. Both protocols are located at layer 7 in the OSI model and depend on TCP at layer 4. Although they are different, RFC 6455 (https://datatracker.ietf.org/doc/html/rfc6455) states that WebSocket "is designed to work over HTTP ports 443 and 80 as well as to support HTTP proxies and intermediaries", thus making it compatible with HTTP. To achieve compatibility, the WebSocket handshake uses the HTTP Upgrade header[3] to change from the HTTP protocol to the WebSocket protocol.

The WebSocket protocol enables interaction between a web browser (or other client application) and a web server with lower overhead than half-duplex alternatives such as HTTP polling, facilitating real-time data transfer from and to the server. This is made possible by providing a standardized way for the server to send content to the client without being first requested by the client, and allowing messages to be passed back and forth while keeping the connection open. In this way, a two-way ongoing conversation can take place between the client and the server. The communications are usually done over TCP port number 443 (or 80 in the case of unsecured connections), which is beneficial for environments that block non-web Internet connections using a firewall. Similar two-way browser–server communications have been achieved in non-standardized ways using stopgap technologies such as Comet or Adobe Flash Player.[4]

Most browsers support the protocol, including Google Chrome, Firefox, Microsoft Edge, Internet Explorer, Safari and Opera.[5]

Unlike HTTP, WebSocket provides full-duplex communication.[6][7] Additionally, WebSocket enables streams of messages on top of TCP. TCP alone deals with streams of bytes with no inherent concept of a message. Before WebSocket, port 80 full-duplex communication was attainable using Comet channels; however, Comet implementation is nontrivial, and due to the TCP handshake and

HTTP header overhead, it is inefficient for small messages. The WebSocket protocol aims to solve these problems without compromising the security assumptions of the web.

The WebSocket protocol specification defines `ws` (WebSocket) and `wss` (WebSocket Secure) as two new uniform resource identifier (URI) schemes[8] that are used for unencrypted and encrypted connections respectively. Apart from the scheme name and fragment (i.e. # is not supported), the rest of the URI components are defined to use URI generic syntax.[9]

Using browser developer tools, developers can inspect the WebSocket handshake as well as the WebSocket frames.[10]

History

WebSocket was first referenced as TCPConnection in the HTML5 specification, as a placeholder for a TCP-based socket API.[11] In June 2008, a series of discussions were led by Michael Carter that resulted in the first version of the protocol known as WebSocket.[12]

The name "WebSocket" was coined by Ian Hickson and Michael Carter shortly thereafter through collaboration on the #whatwg IRC chat room,[13] and subsequently authored for inclusion in the HTML5 specification by Ian Hickson. In December 2009, Google Chrome 4 was the first browser to ship full support for the standard, with WebSocket enabled by default.[14] Development of the WebSocket protocol was subsequently moved from the W3C and WHATWG group to the IETF in February 2010, and authored for two revisions under Ian Hickson.[15]

After the protocol was shipped and enabled by default in multiple browsers, the RFC 6455 (https://datatracker.ietf.org/doc/html/rfc6455) was finalized under Ian Fette in December 2011.

RFC 7692 (https://datatracker.ietf.org/doc/html/rfc7692) introduced compression extension to WebSocket using the DEFLATE algorithm on a per-message basis.

Browser implementation

A secure version of the WebSocket protocol is implemented in Firefox 6,[16] Safari 6, Google Chrome 14,[17] Opera 12.10 and Internet Explorer 10.[18] A detailed protocol test suite report[19] lists the conformance of those browsers to specific protocol aspects.

An older, less secure version of the protocol was implemented in Opera 11 and Safari 5, as well as the mobile version of Safari in iOS 4.2.[20] The BlackBerry Browser in OS7 implements WebSockets.[21] Because of vulnerabilities, it was disabled in Firefox 4 and 5,[22] and Opera 11.[23]

Implementation status

Protocol, version	Draft date	Internet Explorer	Firefox[24] (PC)	Firefox (Android)	Chrome (PC, Mobile)	Safari (Mac, iOS)	Opera (PC, Mobile)	Android Browser
hixie-75 (https://tools.ietf.org/html/draft-hixie-thewebsocketprotocol-75)	February 4, 2010				4	5.0.0		
hixie-76 (https://tools.ietf.org/html/draft-hixie-thewebsocketprotocol-76) **hybi-00 (https://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-00)**	May 6, 2010 May 23, 2010		4.0 (disabled)		6	5.0.1	11.00 (disabled)	
hybi-07 (https://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-07), v7	April 22, 2011		6[25][a]					
hybi-10 (https://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-10), v8	July 11, 2011		7[27][a]	7	14[28]			
RFC 6455 (https://datatracker.ietf.org/doc/html/rfc6455), v13	December, 2011	10[29]	11	11	16[30]	6	12.10[31]	4.4

JavaScript client example

```
// Creates new WebSocket object with a wss URI as the parameter
const socket = new WebSocket('wss://game.example.com/ws/updates');

// Fired when a connection with a WebSocket is opened
socket.onopen = function () {
  setInterval(function() {
    if (socket.bufferedAmount == 0)
      socket.send(getUpdateData());
  }, 50);
};

// Fired when data is received through a WebSocket
```

```
socket.onmessage = function(event) {
  handleUpdateData(event.data);
};

// Fired when a connection with a WebSocket is closed
socket.onclose = function(event) {
  onSocketClose(event);
};

// Fired when a connection with a WebSocket has been closed because of an error
socket.onerror = function(event) {
  onSocketError(event);
};
```

Web server implementation

Nginx has supported WebSockets since 2013, implemented in version 1.3.13[32] including acting as a reverse proxy and load balancer of WebSocket applications.[33]

Apache HTTP Server has supported WebSockets since July, 2013, implemented in version 2.4.5[34][35]

Internet Information Services added support for WebSockets in version 8 which was released with Windows Server 2012.[36]

lighttpd has supported WebSockets since 2017, implemented in version 1.4.46.[37] lighttpd mod_proxy can act as a reverse proxy and load balancer of WebSocket applications. lighttpd mod_wstunnel can construct WebSocket tunnels to transmit arbitrary data, including in JSON format, to a backend application.

Tempesta FW supports WebSockets for HTTP/1.1 and HTTPS connections since 2022.[38] WebSockets over HTTP/2 by RFC 8441 (https://datatracker.ietf.org/doc/html/rfc8441) were considered by the developers as not widely enough deployed and were not implemented.

Protocol

Protocol handshake

To establish a WebSocket connection, the client sends a WebSocket handshake request, for which the server returns a WebSocket handshake response, as shown in the example below.[39]

Client request (just like in HTTP, each line ends with `\r\n` and there must be an extra blank line at the end):

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

Server response:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

The handshake starts with an HTTP request/response, allowing servers to handle HTTP connections as well as WebSocket connections on the same port. Once the connection is established, communication switches to a bidirectional binary protocol which does not conform to the HTTP protocol.

In addition to `Upgrade` headers, the client sends a `Sec-WebSocket-Key` header containing base64-encoded random bytes, and the server replies with a hash of the key in the `Sec-WebSocket-Accept` header. This is intended to prevent a caching proxy from re-sending a previous WebSocket conversation,[40] and does not provide any authentication, privacy, or integrity. The hashing function appends the fixed string `258EAFA5-E914-47DA-95CA-C5AB0DC85B11` (a UUID) to the value from `Sec-WebSocket-Key` header (which is not decoded from base64), applies the SHA-1 hashing function, and encodes the result using base64.[41]

The RFC6455 requires the key MUST be a nonce consisting of a randomly selected 16-byte value that has been base64-encoded,[42] that is 24 bytes in base64 (with last two bytes to be ==). Though some relaxed HTTP servers do allow shorter keys to present, many modern HTTP servers will reject the request with error "invalid Sec-WebSocket-Key header".

Base Framing Protocol

Once the connection is established, the client and server can send WebSocket data or text frames back and forth in full-duplex mode. The data is minimally framed, with a small header followed by payload.[43] WebSocket transmissions are described as "messages", where a single message can optionally be split across several data frames. This can allow for sending of messages where initial data is available but the complete length of the message is unknown (it sends one data frame after another until the end is reached and committed with the FIN bit).

FIN

Indicates the final fragment in a message. 1b.

RSV

MUST be 0 unless defined by an extension. 1b.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
FIN	RSV1	RSV2	RSV3	Opcode				Mask	Payload length						
Extended payload length (optional)															
Masking key (optional)															
Payload data															

Opcode
Operation code. 4b.

0
Continuation frame
1
Text frame
2
Binary frame
8
Connection close
9
Ping
A
Pong
etc.
Reserved

Mask
Set to 1 if the payload data is masked. 1b.

Payload length
The length of the payload data. 7b.

126
The following 2 bytes are interpeted as the payload length.
127
The following 8 bytes are interpeted as the payload length.
0-125
This is the payload length.

Masking key
All frames sent from the client should be masked by this key. This field is absent if the mask bit is set to 0. 4B.
Payload data
The payload data of the fragment.

With extensions to the protocol, this can also be used for multiplexing several streams simultaneously (for instance to avoid monopolizing use of a socket for a single large payload).[44]

Client to Server Masking

The payload data sent from the client should be masked by the masking key. The masking key is a 4 bytes random value chosen by the client and should be unpredictable. The unpredictability of the masking key is essential to prevent malicious applications from selecting the bytes that already appear. The following algorithm is applied to mask the payload data.

```
j = i MOD 4
transformed-octet-i = original-octet-i XOR masking-key-octet-j
```

Security considerations

Unlike regular cross-domain HTTP requests, WebSocket requests are not restricted by the same-origin policy. Therefore, WebSocket servers must validate the "Origin" header against the expected origins during connection establishment, to avoid cross-site WebSocket hijacking attacks (similar to cross-site request forgery), which might be possible when the connection is authenticated with cookies or HTTP authentication. It is better to use tokens or similar protection mechanisms to authenticate the WebSocket connection when sensitive (private) data is being transferred over the WebSocket.[45] A live example of vulnerability was seen in 2020 in the form of Cable Haunt.

Proxy traversal

WebSocket protocol client implementations try to detect whether the user agent is configured to use a proxy when connecting to destination host and port, and if it is, uses HTTP CONNECT method to set up a persistent tunnel.

While the WebSocket protocol itself is unaware of proxy servers and firewalls, it features an HTTP-compatible handshake, thus allowing HTTP servers to share their default HTTP and HTTPS ports (80 and 443 respectively) with a WebSocket gateway or server. The WebSocket protocol defines a ws:// and wss:// prefix to indicate a WebSocket and a WebSocket Secure connection respectively. Both schemes use an HTTP upgrade mechanism to upgrade to the WebSocket protocol. Some proxy servers are transparent and work fine with WebSocket; others will prevent WebSocket from working correctly, causing the connection to fail. In some cases, additional proxy-server configuration may be required, and certain proxy servers may need to be upgraded to support WebSocket.

If unencrypted WebSocket traffic flows through an explicit or a transparent proxy server without WebSockets support, the connection will likely fail.[46]

If an encrypted WebSocket connection is used, then the use of Transport Layer Security (TLS) in the WebSocket Secure connection ensures that an `HTTP CONNECT` command is issued when the browser is configured to use an explicit proxy server. This sets up a tunnel, which provides low-level end-to-end TCP communication through the HTTP proxy, between the WebSocket Secure client and the WebSocket server. In the case of transparent proxy servers, the browser is unaware of the proxy server, so no `HTTP CONNECT` is sent. However, since the wire traffic is encrypted, intermediate transparent proxy servers may simply allow the encrypted traffic through, so there is a much better chance that the WebSocket connection will succeed if WebSocket Secure is used. Using encryption is not free of resource cost, but often provides the highest success rate, since it would be travelling through a secure tunnel.

A mid-2010 draft (version hixie-76) broke compatibility with reverse proxies and gateways by including eight bytes of key data after the headers, but not advertising that data in a `Content-Length: 8` header.[47] This data was not forwarded by all intermediates, which could lead to protocol failure. More recent drafts (e.g., hybi-09[48]) put the key data in a `Sec-WebSocket-Key` header, solving this problem.

See also

- BOSH
- Comparison of WebSocket implementations
- Network socket
- Push technology
- Server-sent events
- XMLHttpRequest
- HTTP/2
- Internet protocol suite
- WebRTC

Notes

a. Gecko-based browsers versions 6–10 implement the WebSocket object as "MozWebSocket",[26] requiring extra code to integrate with existing WebSocket-enabled code.

References

1. "WebSockets Standard" (https://websockets.spec.whatwg.org/). *websockets.spec.whatwg.org*. Retrieved 2022-05-16.
2. "The WebSocket API" (https://www.w3.org/TR/2021/NOTE-websockets-20210128/Overview.html). *www.w3.org*. Retrieved 2022-05-16.
3. Ian Fette; Alexey Melnikov (December 2011). "Relationship to TCP and HTTP" (https://datatracker.ietf.org/doc/html/rfc6455#section-1.7). *RFC 6455 The WebSocket Protocol* (https://datatracker.ietf.org/doc/html/rfc6455). IETF. sec. 1.7. doi:10.17487/RFC6455 (https://doi.org/10.17487%2FRFC6455). RFC 6455 (https://datatracker.ietf.org/doc/html/rfc6455).
4. "Adobe Flash Platform – Sockets" (https://help.adobe.com/en_US/as3/dev/WSb2ba3b1aad8a27b0-181c51321220efd9d1c-8000.html). *help.adobe.com*. Retrieved 2021-07-28. "TCP connections require a "client" and a "server". Flash Player can create client sockets."
5. "The WebSocket API (WebSockets)" (https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API). *MDN Web Docs*. Mozilla Developer Network. 2023-04-06. Archived (https://web.archive.org/web/20210728161324/https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API) from the original on 2021-07-28. Retrieved 2021-07-26.
6. "Glossary: WebSockets" (https://developer.mozilla.org/en-US/docs/Glossary/WebSockets). Mozilla Developer Network. 2015.
7. "HTML5 WebSocket – A Quantum Leap in Scalability for the Web" (https://web.archive.org/web/20210401044556/http://www.websocket.org/quantum.html). *www.websocket.org*. Archived from the original (http://www.websocket.org/quantum.html) on 2021-04-01. Retrieved 2016-01-08.
8. Graham Klyne, ed. (2011-11-14). "IANA Uniform Resource Identifer (URI) Schemes" (https://www.iana.org/assignments/uri-schemes.html). Internet Assigned Numbers Authority. Retrieved 2011-12-10.

9. Ian Fette; Alexey Melnikov (December 2011). "WebSocket URIs" (https://datatracker.ietf.org/doc/html/rfc6455#section-3). *RFC 6455 The WebSocket Protocol* (https://datatracker.ietf.org/doc/html/rfc6455). IETF. sec. 3. doi:10.17487/RFC6455 (https://doi.org/10.17487%2FRFC6455). RFC 6455 (https://datatracker.ietf.org/doc/html/rfc6455).
10. Wang, Vanessa; Salim, Frank; Moskovits, Peter (February 2013). "APPENDIX A: WebSocket Frame Inspection with Google Chrome Developer Tools" (https://web.archive.org/web/20151231184436/http://my.safaribooksonline.com/book/-/9781430247401/appendix-a-inspecting-websocket-traffic/sec1_xhtml). *The Definitive Guide to HTML5 WebSocket*. Apress. ISBN 978-1-4302-4740-1. Archived from the original (http://my.safaribooksonline.com/book/-/9781430247401/appendix-a-inspecting-websocket-traffic/sec1_xhtml) on 31 December 2015. Retrieved 7 April 2013.
11. "HTML 5" (https://www.w3.org/TR/2008/WD-html5-20080610/comms.html#tcp-connections). *www.w3.org*. Retrieved 2016-04-17.
12. "[whatwg] TCPConnection feedback from Michael Carter on 2008-06-18 (whatwg.org from June 2008)" (https://lists.w3.org/Archives/Public/public-whatwg-archive/2008Jun/0165.html). *lists.w3.org*. Retrieved 2016-04-17.
13. "IRC logs: freenode / #whatwg / 20080618" (http://krijnhoetmer.nl/irc-logs/whatwg/20080618#l-1145). *krijnhoetmer.nl*. Retrieved 2016-04-18.
14. "Web Sockets Now Available In Google Chrome" (https://blog.chromium.org/2009/12/web-sockets-now-available-in-google.html). *Chromium Blog*. Retrieved 2016-04-17.
15. <ian@hixie.ch>, Ian Hickson (6 May 2010). "The WebSocket protocol" (https://tools.ietf.org/html/draft-hixie-thewebsocketprotocol-75). *Ietf Datatracker*. Retrieved 2016-04-17.
16. Dirkjan Ochtman (May 27, 2011). "WebSocket enabled in Firefox 6" (https://web.archive.org/web/20120526230019/https://developer.mozilla.org/en/WebSockets). *Mozilla.org*. Archived from the original (https://developer.mozilla.org/en/WebSockets) on 2012-05-26. Retrieved 2011-06-30.
17. "Chromium Web Platform Status" (https://www.chromium.org/developers/web-platform-status). Retrieved 2011-08-03.
18. "WebSockets (Windows)" (https://msdn.microsoft.com/en-us/library/ie/hh673567(v=vs.85).aspx). Microsoft. 2012-09-28. Retrieved 2012-11-07.
19. "WebSockets Protocol Test Report" (https://web.archive.org/web/20160922040928/http://autobahn.ws/testsuite/reports/clients/index.html). Tavendo.de. 2011-10-27. Archived from the original (http://autobahn.ws/testsuite/reports/clients/index.html) on 2016-09-22. Retrieved 2011-12-10.
20. Katie Marsal (November 23, 2010). "Apple adds accelerometer, WebSockets support to Safari in iOS 4.2" (https://www.appleinsider.com/articles/10/11/23/apple_adds_accelerometer_websockets_support_to_safari_in_ios_4_2.html). *AppleInsider.com*. Retrieved 2011-05-09.
21. "Web Sockets API" (https://web.archive.org/web/20110610191150/http://docs.blackberry.com/en/developers/deliverables/29271/Web_Sockets_support_1582781_11.jsp). BlackBerry. Archived from the original (http://docs.blackberry.com/en/developers/deliverables/29271/Web_Sockets_support_1582781_11.jsp) on June 10, 2011. Retrieved 8 July 2011.
22. Chris Heilmann (December 8, 2010). "WebSocket disabled in Firefox 4" (https://hacks.mozilla.org/2010/12/websockets-disabled-in-firefox-4/). *Hacks.Mozilla.org*. Retrieved 2011-05-09.
23. Aleksander Aas (December 10, 2010). "Regarding WebSocket" (https://web.archive.org/web/20101215010748/http://my.opera.com/chooseopera/blog/2010/12/10/regarding-websocket). *My Opera Blog*. Archived from the original (http://my.opera.com/chooseopera/blog/2010/12/10/regarding-websocket) on 2010-12-15. Retrieved 2011-05-09.

24. "WebSockets (support in Firefox)" (https://web.archive.org/web/20120526230019/https://developer.mozilla.org/en/WebSockets). *developer.mozilla.org*. Mozilla Foundation. 2011-09-30. Archived from the original (https://developer.mozilla.org/en/WebSockets) on 2012-05-26. Retrieved 2011-12-10.
25. "Bug 640003 - WebSockets - upgrade to ietf-06" (https://bugzilla.mozilla.org/show_bug.cgi?id=640003). Mozilla Foundation. 2011-03-08. Retrieved 2011-12-10.
26. "WebSockets - MDN" (https://web.archive.org/web/20120526230019/https://developer.mozilla.org/en/WebSockets). *developer.mozilla.org*. Mozilla Foundation. 2011-09-30. Archived from the original (https://developer.mozilla.org/en/WebSockets) on 2012-05-26. Retrieved 2011-12-10.
27. "Bug 640003 - WebSockets - upgrade to ietf-07(comment 91)" (https://bugzilla.mozilla.org/show_bug.cgi?id=640003#c91). Mozilla Foundation. 2011-07-22.
28. "Chromium bug 64470" (https://code.google.com/p/chromium/issues/detail?id=64470). *code.google.com*. 2010-11-25. Retrieved 2011-12-10.
29. "WebSockets in Windows Consumer Preview" (https://blogs.msdn.com/b/ie/archive/2012/03/19/websockets-in-windows-consumer-preview.aspx). *IE Engineering Team*. Microsoft. 2012-03-19. Retrieved 2012-07-23.
30. "WebKit Changeset 97247: WebSocket: Update WebSocket protocol to hybi-17" (https://trac.webkit.org/changeset/97249). *trac.webkit.org*. Retrieved 2011-12-10.
31. "A hot Opera 12.50 summer-time snapshot" (https://web.archive.org/web/20120805234006/http://my.opera.com/ODIN/blog/2012/08/03/a-hot-opera-12-50-summer-time-snapshot). Opera Developer News. 2012-08-03. Archived from the original (http://my.opera.com/ODIN/blog/2012/08/03/a-hot-opera-12-50-summer-time-snapshot) on 2012-08-05. Retrieved 2012-08-03.
32. "Welcome to nginx!" (https://archive.today/20120717014311/http://nginx.org/en/CHANGES). *nginx.org*. Archived from the original (http://nginx.org/en/CHANGES) on 17 July 2012. Retrieved 3 February 2022.
33. "Using NGINX as a WebSocket Proxy" (https://www.nginx.com/blog/websocket-nginx/). *NGINX*. May 17, 2014.
34. "Overview of new features in Apache HTTP Server 2.4" (http://httpd.apache.org/docs/trunk/new_features_2_4.html). *Apache*.
35. "Changelog Apache 2.4" (https://www.apachelounge.com/Changelog-2.4.html). *Apache Lounge*.
36. "IIS 8.0 WebSocket Protocol Support" (https://docs.microsoft.com/en-us/iis/get-started/whats-new-in-iis-8/iis-80-websocket-protocol-support). *Microsoft Docs*. 28 November 2012. Retrieved 2020-02-18.
37. "Release-1 4 46 - Lighttpd - lighty labs" (https://redmine.lighttpd.net/projects/lighttpd/wiki/Release-1_4_46).
38. "Tempesta FW Handling clients WebSockets" (https://github.com/tempesta-tech/tempesta/wiki/Handling-clients#websockets). *Tempesta FW wiki*. Retrieved 6 June 2022.
39. Ian Fette; Alexey Melnikov (December 2011). "Protocol Overview" (https://datatracker.ietf.org/doc/html/rfc6455#section-1.2). *RFC 6455 The WebSocket Protocol* (https://datatracker.ietf.org/doc/html/rfc6455). IETF. sec. 1.2. doi:10.17487/RFC6455 (https://doi.org/10.17487%2FRFC6455). RFC 6455 (https://datatracker.ietf.org/doc/html/rfc6455).
40. "Main Goal of WebSocket protocol" (https://trac.tools.ietf.org/wg/hybi/trac/wiki/FAQ). IETF. Retrieved 25 July 2015. "The computation [...] is meant to prevent a caching intermediary from providing a WS-client with a cached WS-server reply without actual interaction with the WS-server."

41. Ian Fette; Alexey Melnikov (December 2011). "Opening Handshake" (https://datatracker.ietf.org/doc/html/rfc6455#section-1.3). *RFC 6455 The WebSocket Protocol* (https://datatracker.ietf.org/doc/html/rfc6455). IETF. p. 8. sec. 1.3. doi:10.17487/RFC6455 (https://doi.org/10.17487%2FRFC6455). RFC 6455 (https://datatracker.ietf.org/doc/html/rfc6455).
42. Ian Fette; Alexey Melnikov (December 2011). "Opening Handshake" (https://datatracker.ietf.org/doc/html/rfc6455#section-1.3). *RFC 6455 The WebSocket Protocol* (https://datatracker.ietf.org/doc/html/rfc6455). IETF. p. 21. sec. 1.3. doi:10.17487/RFC6455 (https://doi.org/10.17487%2FRFC6455). RFC 6455 (https://datatracker.ietf.org/doc/html/rfc6455).
43. Ian Fette; Alexey Melnikov (December 2011). "Base Framing Protocol" (https://datatracker.ietf.org/doc/html/rfc6455#section-5.2). *RFC 6455 The WebSocket Protocol* (https://datatracker.ietf.org/doc/html/rfc6455). IETF. sec. 5.2. doi:10.17487/RFC6455 (https://doi.org/10.17487%2FRFC6455). RFC 6455 (https://datatracker.ietf.org/doc/html/rfc6455).
44. John A. Tamplin; Takeshi Yoshino (2013). *A Multiplexing Extension for WebSockets* (https://datatracker.ietf.org/doc/html/draft-ietf-hybi-websocket-multiplexing). IETF. I-D draft-ietf-hybi-websocket-multiplexing.
45. Christian Schneider (August 31, 2013). "Cross-Site WebSocket Hijacking (CSWSH)" (https://www.christian-schneider.net/CrossSiteWebSocketHijacking.html#main). *Web Application Security Blog*.
46. Peter Lubbers (March 16, 2010). "How HTML5 Web Sockets Interact With Proxy Servers" (http://www.infoq.com/articles/Web-Sockets-Proxy-Servers). *Infoq.com*. C4Media Inc. Retrieved 2011-12-10.
47. Willy Tarreau (2010-07-06). "WebSocket -76 is incompatible with HTTP reverse proxies" (https://www.ietf.org/mail-archive/web/hybi/current/msg02149.html). *ietf.org* (email). Internet Engineering Task Force. Retrieved 2011-12-10.
48. Ian Fette (June 13, 2011). "Sec-WebSocket-Key" (https://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-09#section-11.4#section-11.4). *The WebSocket protocol, draft hybi-09* (https://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-09#section-11.4). sec. 11.4. Retrieved June 15, 2011.

External links

- IETF Hypertext-Bidirectional (HyBi) working group (https://datatracker.ietf.org/wg/hybi/charter/)
  - RFC 6455 (https://datatracker.ietf.org/doc/html/rfc6455) The WebSocket protocol - Proposed Standard published by the IETF HyBi Working Group
  - The WebSocket protocol (https://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol) - Internet-Draft published by the IETF HyBi Working Group
  - The WebSocket protocol (https://tools.ietf.org/html/draft-hixie-thewebsocketprotocol-76) - Original protocol proposal by Ian Hickson
- The WebSocket API (https://dev.w3.org/html5/websockets/) Archived (https://web.archive.org/web/20150607035919/http://dev.w3.org/html5/websockets/) 2015-06-07 at the Wayback Machine - W3C Working Draft specification of the API
- The WebSocket API (http://www.w3.org/TR/websockets/) - W3C Candidate Recommendation specification of the API
- WebSocket.org (https://www.websocket.org/) Archived (https://web.archive.org/web/20180916101105/http://websocket.org/) 2018-09-16 at the Wayback Machine WebSocket demos, loopback tests, general information and community