HYBI Working Group
Internet-Draft
Intended status: Informational
Expires: September 15, 2011

G. Montenegro, Ed.
Microsoft Corporation

March 14, 2011

HyBi WebSocket Requirements and Features
draft-ietf-hybi-websocket-requirements-02

Abstract

This document states the requirements of the WebSocket Protocol. The goal of the document is to provide a stable base for protocol design and related discussion.

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 September 15, 2011.

Copyright Notice

Table of Contents

1. Introduction

HTTP [RFC2616] is a client/server protocol, where the HTTP servers store the data and provide it when it is requested by clients. When used to retrieve data from an HTTP server, the client sends HTTP requests to the server, and the server returns the requested data in HTTP responses. So the client has to poll the server continuously in order to receive new data.

Recently, techniques that enable bidirectional communication over HTTP have become more pervasive. Those techniques reduce the need to poll continuously the server thanks to the usage of HTTP hanging requests and multiple connections between the client and the server [I-D.ietf-hybi-design-space].

The goal of HyBi is to provide an efficient and clean two-way communication channel between client and server.

The communication channel will:

- Allow each side to, independently from the other, send data when it is willing and ready to do so.
- Rely on a single TCP connection for traffic in both directions.
- Reduce the high overhead produced by HTTP headers in each request/response.

The goal of this work is to provide the set of requirements for the WebSocket Protocol.

In the following sections we list and analyse the requirements from the perspective of clients and servers.

2. Terminology

This document uses the following HyBi-related terms:

connection: A transport layer virtual circuit established between a client and a server for the purpose of communication.

frame: The basic unit of WebSocket communication, consisting of a structured sequence of octets matching the syntax defined in the actual protocol and transmitted on the established communication channel.

message: user message: a block of related data with identified boundaries. A message may comprise multiple frames.

origin server: The server on which a given resource resides or is to be created.

WebSocket handshake: The process (and associated capability negotiation) that sets up the WebSocket communication channel.

WebSocket communication channel: After the WebSocket handshake is complete, the resultant bi-directional communication path between client and server over the transport (e.g., TCP, or SSL over TCP).

WebSocket sub-protocol: The negotiated sub-protocol for use on a WebSocket communication channel that dictates framing, encoding, etc.

2.1. Requirements Language

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 RFC 2119 [RFC2119].

3. WebSocket Requirements

The following requirements for the WebSocket Protocol have been identified both in the HyBi wg input document [I-D.ietf-hybi-thewebsocketprotocol] and in the HyBi mailing list dicussion.

REQ. 1: The WebSocket Protocol MUST run directly on top of the transport protocol (over which the communication was running up to and including the WebSocket handshake). The transport protocol is limited to TCP.

REQ. 2: The WebSocket Protocol MUST be able to handle (send and receive) messages on the transport protocol (over which the communication was running up to and including the WebSocket handshake).

Reason: transfer data as message obviates the need for the receiver to parse/handle partial content.

REQ. 3: The protocol MUST support the ability to fragment a message into frames of a given length.

REQ. 4: It MUST be possible to send a message when the total size is either unknown or exceeds a fixed buffer size.

Reason: This will allow dynamic messages to be constructed and sent without the need to buffer the entire message.

REQ. 5: Textual data MUST be encoded as UTF-8.

REQ. 6: The protocol MUST support and clearly distinguish between textual and binary data types (e.g. binary) via a common framing with explicit length indication.

REQ. 7: The WebSocket protocol MUST allow HTTP and WebSocket connections to be served from the same port. Consideration MUST be given:

* to provide WebSocket services via modules that plug in to existing web infrastructure.

* to making it possible and practical to implement standalone implementations of the protocol without requiring a fully conforming HTTP implementation.

Reason: Some server developers would like to integrate WebSocket support into existing HTTP servers. In addition, the default HTTP and HTTPS ports are ofter favoured for traffic that has to go through a firewall, so service providers will likely want to be able to use WebSocket over ports 80 and 443, even when running a Web server on the same host. However there could be scenarios where it is not opportune or possible to setup a proxy on the same HTTP server.

REQ. 8: If using an HTTP Upgrade exchange in the WebSocket handshake, the protocol MUST be HTTP compatible up to and including the Upgrade exchange.

REQ. 9: The protocol SHOULD make it possible and practical to reuse existing HTTP components where appropriate.

Reason: Reusing existing well-debugged software decreases the number of implementation errors as well as the possibility to introduce security holes, and increases development speed, especially when the WebSocket server is implemented as modules that plug in to existing popular Web servers.

3.1. WebSocket Client Requirements

REQ. 10: The WebSocket Client MUST be able to set up a communication channel with a WebSocket Server using a well-defined handshake.

REQ. 11: The WebSocket Protocol MUST provide for graceful close of an active WebSocket connection on request from the user Application.

Reason: a clean shutdown signals that the other endpoint has definitely received all the messages sent prior to the close, so there is no protocol uncertainty about what has been processed and what can be retried on another connection.

REQ. 12: WebSocket Protocol MUST also allow ungraceful close, either on request from the user application or as a result of a detected error condition.

REQ. 13: The WebSocket Client MUST be able to request the server, during the handshake, to use a specific WebSocket sub-protocol.

REQ. 14: The WebSocket Client MUST have the ability to send and clearly distinguish between arbitrary text or binary content to the server on the established communication channel.

3.2. WebSocket Server Requirements

REQ. 15: The WebSocket Server that accepts to set up a communication channel with a WebSocket Client MUST use a well-defined handshake.

REQ. 16: The WebSocket Server MUST have the ability to send and clearly distinguish between arbitrary text or binary content to the client on the established communication channel.

3.3. WebSocket Proxies Requirements

Todo

REQ. 17: The WebSocket protocol MUST work over existing proxies to the same extent as HTTP or HTTPS already does.

Reason: This is in line with Req on HTTP compliance.

3.4. WebSocket Security Requirements

REQ. 18: The WebSocket Protocol MUST use the Origin-based security model commonly used by Web browsers to restrict which Web pages can contact a WebSocket sever when the WebSocket protocol is used from a Web page.

REQ. 19: When used directly (not from a Web page), the WebSocket Protocol MUST use a security model equivalent to that of direct HTTP or HTTPS usage.

REQ. 20: WebSocket should be designed to be robust against cross-protocol attacks. The protocol design should consider and mitigate the risk presented by WebSocket clients to existing servers (including HTTP servers). It should also consider and mitigate the risk to WebSocket servers presented by clients for other protocols (including HTTP).

Reason: As the WebSocket protocol is expected to be often used in browsers, a careful design is necessary to mitigate the chances for hostile JavaScript to use WebSocket for a cross-protocol attack against vanilla HTTP resources or non-HTTP servers. More the design should prevent the possibility for cross-site XMLHttpRequest (using CORS or XDomainRequest) to be used for a cross-protocol attack against WebSocket resources, potentially violating integrity (though not confidentiality).

Subsequent discussion in the working group has determined consensus on the use of masking as one of the mechanisms to mitigate this concern.

4. Security Considerations

5. IANA Considerations

This requirements document does not mandate any immediate IANA actions. However, such IANA considerations may arise from future HyBi specification documents which try to meet the requirements given here.

6. Acknowledgments

The initial requirements were created by Salvatore Loreto. Thanks to Greg Wilkins and Maciej Stachowiak for fulfilling previous editing duties.

7. References

7.1. Normative References

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

[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.

[I-D.ietf-hybi-thewebsocketprotocol] Fette, I., "The WebSocket protocol", draft-ietf-hybi-thewebsocketprotocol-06 (work in progress), February 2011.

7.2. Informative References

[I-D.ietf-hybi-design-space] Moffitt, J., Loreto, S., Saint-Andre, P., and S. Salsano, "Design Space for Bidirectional Protocols", draft-ietf-hybi-design-space-00 (work in progress), June 2010.

Appendix A. Change Log (to be removed by RFC Editor before publication)

From version -01 to version -02:

- In Req. 1, clarified that the transport protocol is TCP.
- Explicit mention of masking as one of the mechanisms to use in order to mitigate cross-protocol attacks.
- Moved Requirements Language to terminology section.
- Got rid of MUST in intro.

From version -00 to version -01:

- Modified definition of a Message to reflect recent consensus that it may comprise multiple frames.
- Added definitions for WebSocket handshake, WebSocket communication channel and WebSocket sub-protocol.

- Updated references to official IETF documents, moved "design-space" to informational.
- Added a new requirement to support the ability to fragment a message into frames of a given length
- Reworded Req 5 to reflect recent consensus on "binary+data" as well as common framing with explicit length indication
- Reworded Req 7 to reflect recent consensus (declared in Maastricht) on "HTTP compliance"
- Reworded Req 12 and 13 into a single Req on the client being able to send text or binary content. Elided mention of "discrete blocks" as the structure of messages is captured elsewhere.
- Reworded Req 15 and 16 into a single Req on the server being able to send text or binary content. Elided mention of "discrete blocks" as the structure of messages is captured elsewhere.
- Added a requirement in the proxies requirements section along the lines of the HTTP compliance requirement, per consensus declared in Maastricht.
- Modified security requirement when used from outside a browser to avoid wording in terms of a security model equivalent to that of browser-based usage.
- Various editorial changes.

Author's Address

Gabriel Montenegro (editor)
Microsoft Corporation
One Microsoft Way
Redmond 98052
USA

Email: gabriel.montenegro@microsoft.com