XMPP Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 21, 2014

L. Stout, Ed.
&yet
J. Moffitt
Mozilla
E. Cestari
cstar industries
April 19, 2014

An XMPP Sub-protocol for WebSocket draft-ietf-xmpp-websocket-03

Abstract

This document defines a binding for the XMPP protocol over a WebSocket transport layer. A WebSocket binding for XMPP provides higher performance than the current HTTP binding for XMPP.

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 October 21, 2014.

Copyright Notice

Table of Contents

1. Introduction

Applications using the Extensible Messaging and Presence Protocol (XMPP) (see [RFC6120] and [RFC6121]) on the Web currently make use of BOSH (see [XEP-0124] and [XEP-0206]), an XMPP binding to HTTP. BOSH is based on the HTTP long polling technique, and it suffers from high transport overhead compared to XMPP's native binding to TCP. In addition, there are a number of other known issues with long polling [RFC6202], which have an impact on BOSH-based systems.

It would be much better in most circumstances to avoid tunneling XMPP over HTTP long polled connections and instead use the XMPP protocol directly. However, the APIs and sandbox that browsers have provided do not allow this. The WebSocket protocol [RFC6455] exists to solve these kinds of problems and is a bidirectional protocol that provides

a simple message-based framing layer over raw sockets, allowing for more robust and efficient communication in web applications.

The WebSocket protocol enables two-way communication between a client and a server, effectively emulating TCP at the application layer and therefore overcoming many of the problems with existing long-polling techniques for bidirectional HTTP. This document defines a WebSocket sub-protocol for XMPP.

2. Terminology

The basic unit of framing in the WebSocket protocol is called a message. In XMPP, the basic unit is the stanza, which is a subset of the first-level children of each document in an XMPP stream (see Section 9 of [RFC6120]). XMPP also has a concept of messages, which are stanzas with a top-level element of <message/>. In this document, the word "message" will mean a WebSocket message, not an XMPP message stanza, unless otherwise noted.

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 [RFC2119].

3. XMPP Sub-Protocol

3.1. Handshake

The XMPP sub-protocol is used to transport XMPP over a WebSocket connection. The client and server agree to this protocol during the WebSocket handshake (see Section 1.3 of [RFC6455]).

During the WebSocket handshake, the client MUST include the |Sec-WebSocket-Protocol| header in its handshake, and the value |xmpp| MUST be included in the list of protocols. The reply from the server MUST also contain |xmpp| in its own |Sec-WebSocket-Protocol| header in order for an XMPP sub-protocol connection to be established.

Once the handshake is complete, WebSocket messages sent or received will conform to the protocol defined in the rest of this document.

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


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

[WebSocket connection established]

C:  <open xmlns="urn:ietf:params:xml:ns:xmpp-framing"
          to="example.com"
          version="1.0" />
```

3.2. WebSocket Messages

Data frame messages in the XMPP sub-protocol MUST be of the text type and contain UTF-8 encoded data.

3.3. XMPP Framing

The WebSocket XMPP sub-protocol deviates from the standard method of constructing and using XML streams as defined in [RFC6120] by adopting the message framing provided by WebSocket to delineate the stream open and close headers, stanzas, and other top-level stream elements.

3.3.1. Framed XML Stream

The start of a framed XML stream is marked by the use of an opening "stream header" which is an <open/> element with the appropriate attributes and namespace declarations (see Section 3.3.2). The attributes of the <open/> element are the same as those of the <stream/> element defined in [RFC6120], and with the same semantics.

The end of a framed XML stream is denoted by the closing "stream header" which is a <close/> element with its associated attributes and namespace declarations (see Section 3.3.2).

The introduction of the <open/> and <close/> elements is motivated by the parsable XML document framing restriction in Section 3.3.3. As a consequence, note that a framed XML stream does not provided a wrapping <stream:stream/> element encompassing the entirety of the XML stream, as in [RFC6120].

3.3.2. Framed Stream Namespace

The XML stream "headers" (the <open/> and <close/> elements) MUST be qualified by the namespace 'urn:ietf:params:xml:ns:xmpp-framing' (the "framed stream namespace"). If this rule is violated, the entity that receives the offending stream header MUST close the stream with an error, which SHOULD be <invalid-namespace> (see Section 4.9.3.10 of [RFC6120]).

3.3.3. Stream Frames

The individual frames of a framed XML stream have a one-to-one correspondence with WebSocket messages, and MUST be parsable as standalone XML documents, complete with all relevant namespace and language declarations. The inclusion of XML declarations, however, is NOT RECOMMENDED as WebSocket messages are already mandated to be UTF-8 encoded and therefore would only add a constant size overhead to each message.

The first character of each frame MUST be a '<' character.

Every XMPP stanza or other XML element (including the stream open and close headers) sent directly over the XML stream MUST be sent in its own frame.

Examples of WebSocket messages that contain independently parsable XML documents (note that for stream features and errors, there is no parent context element providing the "stream" namespace prefix as in [RFC6120], and thus the stream namespace MUST be declared):

```
-- WS Message boundary --
<stream:features xmlns:stream="http://etherx.jabber.org/streams">
  <bind xmlns="urn:ietf:params:xml:ns:xmpp-bind"/>
</stream:features>
-- WS Message boundary --
<error xmlns="http://etherx.jabber.org/streams">
  <host-unknown xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
</error>
-- WS Message boundary --
<message xmlns="jabber:client" xml:lang="en">
  <body>Every WebSocket message is parsable by itself.</body>
</message>
```

3.4. Stream Initiation

The first message sent by the initiating entity after the WebSocket opening handshake is complete MUST be an <open/> element qualified by the "urn:ietf:params:xml:ns:xmpp-framing" namespace. The 'from', 'id', 'to', and 'version' attributes of this element mirror those of the XMPP opening stream tag as defined for the 'http://etherx.jabber.org/streams' namespace in XMPP [RFC6120].

The receiving entity MUST respond with an <open /> element, or a <close /> element (see Section 3.6.1).

Clients MUST NOT multiplex XMPP streams over the same WebSocket.

3.5. Stream Errors

Stream level errors in XMPP are terminal. Should such an error occur, the server MUST send the stream error as a complete element in a message to the client.

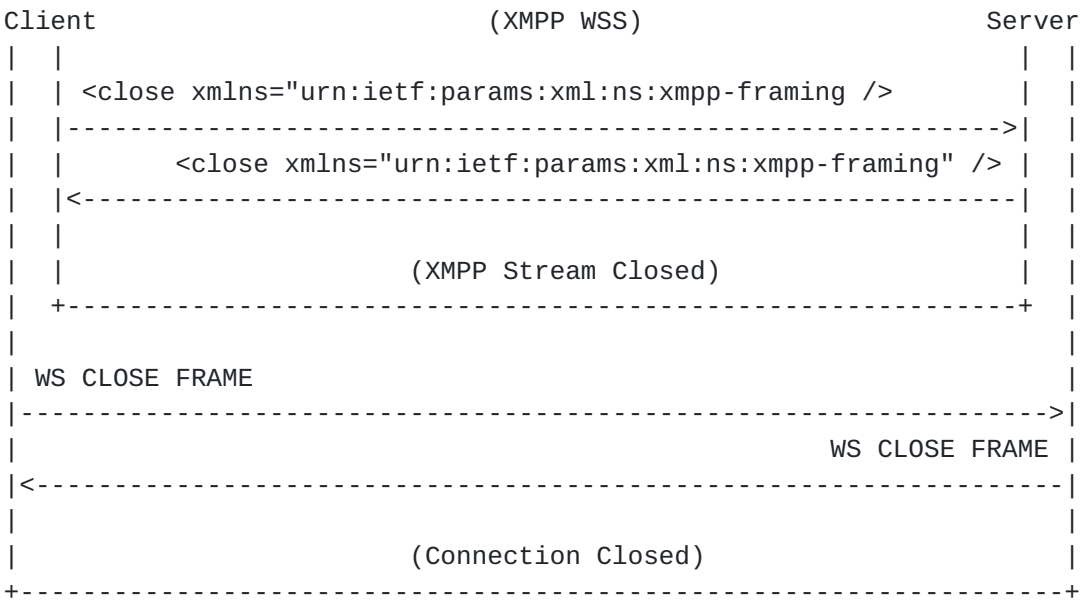
If the error occurs during the opening of a stream, the server MUST send the initial open element response, followed by the stream level error in a second WebSocket message frame. The server MUST then close the connection as specified in Section 3.6.

3.6. Closing the Connection

Either the server or the client may close the connection at any time. Before closing the connection, the closing party SHOULD close the XMPP stream, if it has been established, by sending a message with the <close/> element, qualified by the "urn:ietf:params:xml:ns:xmpp-framing" namespace. The stream is considered closed when a corresponding <close/> element is received from the other party.

To close the WebSocket connection, the closing party MUST initiate the WebSocket closing handshake (see Section 7.1.2 of [RFC6455]).

An example of ending an XMPP over WebSocket session by first closing the XMPP stream layer and then the WebSocket connection layer:



If a client closes the WebSocket connection without closing the XMPP stream after having enabled stream management (see Section 3.10), the server SHOULD keep the XMPP session alive for a period of time based on server policy, as specified in [XEP-0198]. If the client has not negotiated the use of [XEP-0198], there is no distinction between a stream that was closed as described above and a simple disconnection; the stream is then considered implicitly closed and the XMPP session ended.

3.6.1. see-other-uri

If the server (or a connection manager intermediary) wishes to instruct the client to move to a different WebSocket endpoint (e.g. for load balancing purposes), the server MAY send a <close/> element and set the "see-other-uri" attribute to the URI of the new connection endpoint (which MAY be for a different transport method, such as BOSH (see [XEP-0124] and [XEP-0206]).

Clients MUST NOT accept suggested endpoints with a lower security context (e.g. moving from a "wss://" endpoint to a "ws://" or "http://" endpoint).

An example of the server closing a stream and instructing the client to connect at a different WebSocket endpoint:

```
S: <close xmlns="urn:ietf:params:xml:ns:xmpp-framing"
          see-other-uri="wss://otherendpoint.example/xmpp-bind" />
```

3.7. Stream Restarts

Whenever a stream restart is mandated, both the server and client streams are implicitly closed and new streams MUST be opened, using the same process as in Section 3.4. The client MUST send a new stream <open/> element and MUST NOT send a closing <close/> element.

An example of restarting the stream after successful SASL negotiation:

```
S: <success xmlns="urn:ietf:params:xml:ns:xmpp-sasl" />

[Streams implicitly closed]

C: <open xmlns="urn:ietf:params:xml:ns:xmpp-framing"
         to="example.com"
         version="1.0" />
```

3.8. Pings and Keepalives

XMPP servers often send "whitespace keepalives" (see Section 4.6.1 of [RFC6120]) between stanzas to maintain an XML stream, and XMPP clients can do the same as these extra whitespace characters are not significant in the protocol. Servers and clients SHOULD use WebSocket ping control frames instead for this purpose.

In some cases, the WebSocket connection might be served by an intermediary connection manager and not the XMPP server. In these situations, the use of WebSocket ping messages are insufficient to test that the XMPP stream is still alive. Both the XMPP Ping extension [XEP-0199] and the XMPP Stream Management extension [XEP-0198] provide mechanisms to ping the XMPP server, and either extension (or both) MAY be used to determine the state of the connection.

3.9. Use of TLS

TLS cannot be used at the XMPP sub-protocol layer because the sub-protocol does not allow for raw binary data to be sent. Instead, enabling TLS SHOULD be done at the WebSocket layer using secure WebSocket connections via the |wss| URI scheme. (See Section 10.6 of [RFC6455].)

Because TLS is to be provided outside of the XMPP sub-protocol layer, a server MUST NOT advertise TLS as a stream feature (see Section 4.6 of [RFC6120]), and a client MUST ignore any advertised TLS stream feature, when using the XMPP sub-protocol.

3.10. Stream Management

In order to alleviate the problems of temporary disconnections, the XMPP Stream Management extension [XEP-0198] MAY be used to confirm when stanzas have been received by the server.

In particular, the use of session resumption in [XEP-0198] MAY be used to allow for recreating the same stream session state after a temporary network unavailability or after navigating to a new URL in a browser.

4. Discovering the WebSocket Connection Method

Section 3 of [RFC6120] defines a procedure for connecting to an XMPP server, including ways to discover the TCP/IP address and port of the server. When using the WebSocket binding as specified in this document (instead of the TCP binding as specified in [RFC6120]), a client needs an alternative way to discover information about the server's connection methods, since web browsers and other WebSocket-capable software applications typically cannot obtain such information from the Domain Name System.

The alternative lookup process uses Web Host Metadata [RFC6415] and Web Linking [RFC5988], where the link relation type is "urn:xmpp:alt-connections:websocket" as described in Discovering Alternate XMPP Connection Methods [XEP-0156]. An example follows.

```
<XRD xmlns='http://docs.oasis-open.org/ns/xri/xrd-1.0'>
  <Link rel="urn:xmpp:alt-connections:websocket"
        href="wss://webcm.example.net:443/ws" />
</XRD>
```

Servers MAY expose discovery information using host-meta documents, and clients MAY use such information to determine the WebSocket endpoint for a server.

Use of web-host metadata MAY be used to establish trust between the XMPP server domain and the WebSocket endpoint, particularly in multi-tenant situations where the same WebSocket endpoint is serving multiple XMPP domains.

5. IANA Considerations

5.1. WebSocket Subprotocol Name

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

Subprotocol Identifier: xmpp

Subprotocol Common Name: WebSocket Transport for the Extensible Messaging and Presence Protocol (XMPP)

Subprotocol Definition: this document

5.2. URN Sub-Namespace

A URN sub-namespace for framing of Extensible Messaging and Presence Protocol (XMPP) streams is defined as follows.

URI: urn:ietf:params:xml:ns:xmpp-framing

Specification: this document

Description: This is the XML namespace name for framing of Extensible Messaging and Presence Protocol (XMPP) streams as defined by RFC XXXX.

Registrant Contact: IESG <iesg@ietf.org>

6. Security Considerations

Since application level TLS cannot be used (see Section 3.9), applications need to protect the privacy of XMPP traffic at the WebSocket or other appropriate layer.

Browser based applications are not able to inspect and verify at the application layer the certificate used for the WebSocket connection to ensure that it corresponds to the domain specified as the "to" address of the XMPP stream. For hosts whose domain matches the origin for the WebSocket connection, that check is already performed by the browser. However, in situations where the domain of the XMPP server might not match the origin for the WebSocket endpoint (especially multi-tenant hosting situations), the web host metadata method (see [RFC6415] and [XEP-0156]) MAY be used to delegate trust from the XMPP server domain to the WebSocket origin.

When presented with a new WebSocket endpoint via the "see-other-uri" attribute of a <close/> element, clients MUST NOT accept the suggestion if the security context of the new endpoint is lower than the current one in order to prevent downgrade attacks from a "wss://" endpoint to "ws://".

The Security Considerations for both WebSocket (see Section 10 of [RFC6455] and XMPP (see Section 13 of [RFC6120]) apply to the WebSocket XMPP sub-protocol.

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.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

7.2. Informative References

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, October 2010.

[RFC6121] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 6121, March 2011.

[RFC6202] Loreto, S., Saint-Andre, P., Salsano, S., and G. Wilkins, "Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP", RFC 6202, April 2011.

[RFC6415] Hammer-Lahav, E. and B. Cook, "Web Host Metadata", RFC 6415, October 2011.

[XEP-0124] Paterson, I., Smith, D., Saint-Andre, P., Moffitt, J., and L. Stout, "Bidirectional-streams Over Synchronous HTTP (BOSH)", XSF XEP 0124, November 2013.

[XEP-0156] Hildebrand, J., Saint-Andre, P., and L. Stout, "Discovering Alternative XMPP Connection Methods", XSF XEP 0156, January 2014.

[XEP-0198] Karneges, J., Saint-Andre, P., Hildebrand, J., Forno, F., Cridland, D., and M. Wild, "Stream Management", XSF XEP 0198, June 2011.

[XEP-0199] Saint-Andre, P., "XMPP Ping", XSF XEP 0199, June 2009.

[XEP-0206]
Paterson, I., Saint-Andre, P., and L. Stout, "XMPP Over BOSH", XSF XEP 0206, November 2013.

[XML-SCHEMA]
Thompson, H., Maloney, M., Mendelsohn, N., and D. Beech, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>.

Appendix A. XML Schema

The following schema formally defines the 'urn:ietf:params:xml:ns :xmpp-framing' namespace used in this document, in conformance with W3C XML Schema [XML-SCHEMA]. Because validation of XML streams and stanzas is optional, this schema is not normative and is provided for descriptive purposes only.

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:ietf:params:xml:ns:xmpp-framing'
    xmlns='urn:ietf:params:xml:ns:xmpp-framing'
    elementFormDefault='unqualified'>

  <xs:element name='open'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='empty'>
          <xs:attribute name='from' type='xs:string'
                        use='optional'/>
          <xs:attribute name='id' type='xs:string'
                        use='optional'/>
          <xs:attribute name='to' type='xs:string'
                        use='optional'/>
          <xs:attribute name='version' type='xs:decimal'
                        use='optional'/>
          <xs:attribute ref='xml:lang'
                        use='optional'/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name='close'>
    <xs:complexType>
```

```
        <xs:simpleContent>
          <xs:extension base='empty'>
            <xs:attribute name='from' type='xs:string'
                          use='optional'/>
            <xs:attribute name='id' type='xs:string'
                          use='optional'/>
            <xs:attribute name='see-other-uri' type='xs:anyURI'
                          use='optional'/>
            <xs:attribute name='to' type='xs:string'
                          use='optional'/>
            <xs:attribute name='version' type='xs:decimal'
                          use='optional'/>
            <xs:attribute ref='xml:lang'
                          use='optional'/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>

    <xs:simpleType name='empty'>
      <xs:restriction base='xs:string'>
        <xs:enumeration value=''/>
      </xs:restriction>
    </xs:simpleType>

  </xs:schema>
```

Authors' Addresses

Lance Stout (editor)
&yet

Email: lance@andyet.net

Jack Moffitt
Mozilla

Email: jack@metajack.im

Eric Cestari
cstar industries

Email: eric@cstar.io