Advanced Message Queuing Protocol (AMQP) WebSocket Binding (WSB) Version 1.0

Working Draft 03

06 August 2013

Technical Committee:

[OASIS Advanced Message Queuing Protocol (AMQP) Bindings and Mappings (AMQP-BINDMAP) TC](https://www.oasis-open.org/committees/amqp-bindmap/)

Chair:

Steve Huston ([shuston@riverace.com](mailto:shuston@riverace.com)), Individual

Editors:

Robert Godfrey ([robert.godfrey@jpmorgan.com](mailto:robert.godfrey@jpmorgan.com)), [JPMorgan Chase & Co.](http://www.jpmorganchase.com/)

David Ingham ([David.Ingham@microsoft.com](mailto:David.Ingham@microsoft.com)), [Microsoft](http://www.microsoft.com/)

Rob Dolin ([RobDolin@microsoft.com](mailto:RobDolin@microsoft.com)), [Microsoft](http://www.microsoft.com/)

Additional artifacts:

This prose specification is one component of a Work Product that also includes:

* Other parts (list titles and/or file names)

Related work:

This specification is related to:

* *OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0 Part 0: Overview*. 29 October 2012. OASIS Standard. <http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-overview-v1.0-os.html>.

Abstract:

AMQP WebSocket Binding is layered below the AMQP protocol as a mechanism for sending and receiving AMQP frames as binary payloads of WebSocket messages.

This specification describes how to map from AMQP concepts to WebSocket concepts, how to open a connection, how to send data, how to close a connection, and how to handle errors. The specification also describes two authentication and security sub-protocols: Raw AMQP over WebSocket and AMQP over SASL.

Status:

This [Working Draft](http://www.oasis-open.org/committees/process.php#dWorkingDraft) (WD) has been produced by one or more TC Members; it has not yet been voted on by the TC or [approved](http://www.oasis-open.org/committees/process.php#committeeDraft) as a Committee Draft (Committee Specification Draft or a Committee Note Draft). The OASIS document [Approval Process](http://www.oasis-open.org/committees/process.php#standApprovProcess) begins officially with a TC vote to approve a WD as a Committee Draft. A TC may approve a Working Draft, revise it, and re-approve it any number of times as a Committee Draft.

Initial URI pattern:

http://docs.oasis-open.org/amqp-bindmap/amqp-wsb/v1.0/csd01/amqp-wsb-v1.0-csd01.doc

(Managed by OASIS TC Administration; please don’t modify.)

Copyright © OASIS Open 2013. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](http://www.oasis-open.org/policies-guidelines/ipr) may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Table of Contents

[1 Introduction 4](#_Toc362615125)

[1.1 Terminology 4](#_Toc362615126)

[1.2 Normative References 4](#_Toc362615127)

[1.3 Non-Normative References 5](#_Toc362615128)

[2 Opening a WebSocket Connection 6](#_Toc362615129)

[2.1 WebSocket Opening 6](#_Toc362615130)

[2.2 AMQP Opening 6](#_Toc362615131)

[2.3 Example 7](#_Toc362615132)

[3 Opening a WebSocket Connection (with SASL authentication) 9](#_Toc362615133)

[3.1 WebSocket Opening 9](#_Toc362615134)

[3.2 SASL Authentication Exchange 9](#_Toc362615135)

[3.3 AMQP Opening 9](#_Toc362615136)

[3.4 Example 10](#_Toc362615137)

[4 Sending Data 11](#_Toc362615138)

[4.1 AMQP as WebSocket Binary 11](#_Toc362615139)

[4.1.1 Masking 12](#_Toc362615140)

[4.2 AMQP Frame Mapping to WebSocket Message 12](#_Toc362615141)

[4.3 Connection Keep-Alive 13](#_Toc362615142)

[5 Authentication and Security 13](#_Toc362615143)

[5.1 Raw AMQP over WebSocket 13](#_Toc362615144)

[5.2 Raw AMQP over Secure WebSocket 13](#_Toc362615145)

[5.3 Ports 80 and 443 13](#_Toc362615146)

[6 Normal Closing of a Connection 14](#_Toc362615147)

[6.1 Example 15](#_Toc362615148)

[7 IANA Considerations 16](#_Toc362615149)

[8 # Conformance 17](#_Toc362615150)

[Appendix A. Acknowledgments 18](#_Toc362615151)

[Appendix B. Example Title 19](#_Toc362615152)

[B.1 Subsidiary section 19](#_Toc362615153)

[B.1.1 Sub-subsidiary section 19](#_Toc362615154)

[Appendix C. Revision History 20](#_Toc362615155)

# Introduction

[All text is normative unless otherwise labeled]

This specification describes how to transfer AMQP messages as the binary payload of WebSocket messages.

Each AMQP message consists of one or more AMQP frames. Each AMQP frame maps to a WebSocket message which in-turn maps to one or more WebSocket frames. Thus, there is a one-to-many mapping between an AMQP frame and WebSocket frames. This allows for intermediaries along the communication path to split a WebSocket message into potentially multiple WebSocket frames.

The WebSocket Protocol is particularly useful in that:

* Its initial handshake appears as HTTP traffic and it uses the same ports (80 and 443) as HTTP traffic so it is often able to pass-through network security devices without requiring special configuration or opening of additional ports.
* Many web browsers have built-in infrastructure for sending and receiving WebSocket protocol messages.

Each AMQP frame maps to a Web Sockets message.

Each AMQP frame can be transmitted as one (left) or more (right) WebSocket frames:

[ AMQP frame 1 of 2 ] [ AMQP frame 2 of 2 ]

| / / \ \_\_\_\_\_

| | / \ \_\_\_\_

| / / \

[ WS frame 1 of 3] [ WS frame 2 of 3 ] [ WS frame 3 of 3 ]

## 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].

## Normative References

[AMQP] Godfrey, Robert; Ingham, David; Schloming, Rafael, “Advanced Message Queueing Protocol (AMQP) Version 1.0”, October 2012. OASIS Standard. <https://www.oasis-open.org/standards#amqpv1.0>

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., "Hypertext Transfer Protocol -- HTTP/1.1", RFC2616, June 1999. <http://www.w3.org/Protocols/rfc2616/rfc2616.html>.

[RFC4422] Melnikov, A., and Zeilenga, K., “Simple Authentication and Security Layer (SASL)", RFC4422, June 2006. <http://tools.ietf.org/html/rfc4422>.

[RFC6455] Fette, I., and Melinkov, A., “The WebSocket Protocol”, December 2011. RFC 6455, December 2011. <http://tools.ietf.org/html/rfc6455>.

## Non-Normative References

**[SIPWS]** Baz Castillo, I., Millan Villegas, J., Pascual, V., “The WebSocket Protocol as a Transport for the Session Initiation Protocol (SIP)”, 13 June 2013. http://tools.ietf.org/html/draft-ietf-sipcore-sip-websocket-09

[XMPPWS] Stout, L., Moffitt, J., and Cestari, E., “An XMPP Sub-Protocol for WebSocket”, 18 May 2013. <http://tools.ietf.org/html/draft-moffitt-xmpp-over-websocket-03>.

**NOTE: The proper format for citation of technical work produced by an OASIS TC (whether Standards Track or Non-Standards Track) is:**

**[Citation Label]**

Work Product [title](http://docs.oasis-open.org/specGuidelines/ndr/namingDirectives.html#workProductName) (italicized). Approval date (DD Month YYYY). OASIS [Stage](http://docs.oasis-open.org/specGuidelines/ndr/namingDirectives.html#stage) Identifier and [Revision](http://docs.oasis-open.org/specGuidelines/ndr/namingDirectives.html#revision) Number (*e.g.*, OASIS Committee Specification Draft 01). Principal URI ([version-specific URI](http://docs.oasis-open.org/specGuidelines/ndr/namingDirectives.html#this-version), *e.g*., with filename component: somespec-v1.0-csd01.html).

For example:

**[OpenDoc-1.2]** *Open Document Format for Office Applications (OpenDocument) Version 1.2*. 19 January 2011. OASIS Committee Specification Draft 07. <http://docs.oasis-open.org/office/v1.2/csd07/OpenDocument-v1.2-csd07.html>.

**[CAP-1.2]** *Common Alerting Protocol Version 1.2*. 01 July 2010. OASIS Standard. <http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.html>.

1. Opening a WebSocket Connection

To establish a connection, first the WebSocket Protocol connection MUST be opened, followed by the AMQP connection.

## WebSocket Opening

The WebSocket Protocol connection MUST be opened as described in **[RFC6455]** section 4. The initiating AMQP endpoint (WebSocket Client) sends a HTTP GET request to the receiving AMQP endpoint (WebSocket Server.) The WebSocket Server provides a HTTP 101 (“Switching Protocols”) response including the HTTP header: “Upgrade: websocket”.

The Client MUST include a value referencing a version of [AMQP] in the Sec-WebSocket-Protocol header in its handshake request. For regular AMQP over WebSocket Protocol, the value for this HTTP header should be the US-ASCII text string “AMQP0100”.

If the Server agrees to communicate using the requested protocol, the 101 reply from the Server MUST include a matching reference to a version of [AMQP] in its Sec-WebSocket-Protocol header.

If the Server does not agree to the sub-protocol requested by the Client, the Server MUST NOT return a Sec-WebSocket-Protocol header. The Client MUST then close the connection.

If the Client receives a HTTP 3XX redirect response from the Server, the Client MAY follow the redirect.

See **[RFC6455]** sections 4.1 and 4.2 for additional details on the WebSocket Opening Handshake.

## AMQP Opening

Once the WebSocket connection has been established, the AMQP Connection, Session, and Links MUST be negotiated. This is done using AMQP frames with the Open, Begin, and Attach performatives as described in **[AMQP]** section 2.7.

A single WebSocket connection maps to a single AMQP connection. As is normal for AMQP, there MAY be potentially many AMQP sessions over a single WebSocket connection / AMQP connection.

## Example

The below example section is non-normative.

WS Client WS Service

/ AMQP Endpoint / AMQP Endpoint

| |

| HTTP GET (WS handshake) F1 |

|------------------------------->|

| 101 Switching Protocols F2 |

|<-------------------------------|

| |

| OPEN (AMQP Connection) F3 |

|------------------------------->|

| OPEN |

|<-------------------------------|

| |

| BEGIN (AMQP Session) |

|------------------------------->|

| BEGIN |

|<-------------------------------|

| |

| ATTACH (AMQP Link from C to S) |

|------------------------------->|

| ATTACH |

|<-------------------------------|

| |

| ATTACH (AMQP Link from S to C) |

|<-------------------------------|

| ATTACH |

|------------------------------->|

| |

Note: The AMQP frames with Open, Begin, and Attach performatives sent from one AMQP endpoint do not necessarily need to wait for the other AMQP endpoint. They could be sent in the order: Open, Begin, Attach, Attach; then Open, Begin, Attach, Attach.

Figure 1: Example request:

GET /examplepath HTTP/1.1

Host: server.example.com

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Key: ...

Sec-WebSocket-Protocol: AMQP0100

Sec-WebSocket-Version: 13

...

Figure 2: Example response:

HTTP/1.1 101 Switching Protocols

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Accept: ...

Sec-WebSocket-Protocol: AMQP0100

...

Additional example detail showing AMQP Open, Begin, and Attach performatives can be found in **[AMQP]**.

Opening a WebSocket Connection   
(with SASL authentication)

To establish a connection, first the WebSocket Protocol connection MUST be opened, followed by the SASL authentication exchange, followed by the AMQP connection.

## WebSocket Opening

The WebSocket Protocol connection MUST be opened as described in **[RFC6455]** section 4. The initiating AMQP endpoint (WebSocket Client) sends a HTTP GET request to the receiving AMQP endpoint (WebSocket Server.) The WebSocket Server provides a HTTP 101 (“Switching Protocols”) response including the HTTP header: “Upgrade: websocket”.

The Client MUST include a value referencing a version of [AMQP] in the Sec-WebSocket-Protocol header in its handshake request. For AMQP SASL layer over WebSocket Protocol, the value for this HTTP header should be the US-ASCII text string “AMQP3100AMQP0100”.

If the Server agrees to communicate using the requested protocol, the 101 reply from the Server MUST include a matching reference to a version of [AMQP] in its Sec-WebSocket-Protocol header.

If the Server does not agree to the sub-protocol requested by the Client, the Server MUST NOT return a Sec-WebSocket-Protocol header. The Client MUST then close the connection.

If the Client receives a HTTP 3XX redirect response from the Server, the Client MAY follow the redirect.

See **[RFC6455]** sections 4.1 and 4.2 for additional details on the WebSocket Opening Handshake.

## SASL Authentication Exchange

Next, a SASL authentication exchange must take place. As described in [RFC4422], “consists of a message from the client to the server requesting authentication via a particular mechanism, followed by one or more pairs of challenges from the server and responses from the client, followed by a message from the server indicating the outcome of the authentication exchange.”

## AMQP Opening

Once the WebSocket connection has been established, and the SASL layer has been established, the AMQP Connection, Session, and Links MUST be negotiated. This is done using AMQP frames with the Open, Begin, and Attach performatives as described in **[AMQP]** section 2.7.

A single WebSocket connection maps to a single AMQP connection. As is normal for AMQP, there MAY be potentially many AMQP sessions over a single WebSocket connection / AMQP connection.

## Example

The below example section is non-normative.

WS Client WS Service

/ AMQP Endpoint / AMQP Endpoint

| |

| HTTP GET (WS handshake) F3 |

|------------------------------->|

| 101 Switching Protocols F4 |

|<-------------------------------|

| |

| SASL Authentication Exchange |

|<------------------------------>|

| |

| OPEN (AMQP Connection) |

|------------------------------->|

| OPEN |

|<-------------------------------|

| |

| BEGIN (AMQP Session) |

|------------------------------->|

| BEGIN |

|<-------------------------------|

| |

| ATTACH (AMQP Link from C to S) |

|------------------------------->|

| ATTACH |

|<-------------------------------|

| |

| ATTACH (AMQP Link from S to C) |

|<-------------------------------|

| ATTACH |

|------------------------------->|

| |

They could be sent in the order: Open, Begin, Attach, Attach; then Open, Begin, Attach, Attach.

Figure 3: Example request:

GET /examplepath HTTP/1.1

Host: server.example.com

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Key: ...

Sec-WebSocket-Protocol: AMQP3100AMQP0100

Sec-WebSocket-Version: 13

...

Figure 4: Example response:

HTTP/1.1 101 Switching Protocols

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Accept: ...

Sec-WebSocket-Protocol: AMQP3100AMQP0100

...

Additional example detail showing the SASL authentication exchange can be found in **[RFC4422]**.

Additional example detail showing AMQP Open, Begin, and Attach performatives can be found in **[AMQP]**.

1. Sending Data

## AMQP as WebSocket Binary

AMQP content MUST be sent as binary data payloads of WebSocket messages.

WebSocket messages are framed as illustrated below from **[RFC6455]** section 5.2:

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-------+-+-------------+-------------------------------+

|F|R|R|R| opcode|M| Payload len | Extended payload length |

|I|S|S|S| (4) |A| (7) | (16/64) |

|N|V|V|V| |S| | (if payload len==126/127) |

| |1|2|3| |K| | |

+-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +

| Extended payload length continued, if payload len == 127 |

+ - - - - - - - - - - - - - - - +-------------------------------+

| |Masking-key, if MASK set to 1 |

+-------------------------------+-------------------------------+

| Masking-key (continued) | Payload Data |

+-------------------------------- - - - - - - - - - - - - - - - +

: Payload Data continued ... :

+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +

| Payload Data continued ... |

+---------------------------------------------------------------+

|  |  |
| --- | --- |
| FIN | Indicates that this is the final WebSocket fragment in a message.  The first fragment MAY also be the final fragment. |
| RSV1, RSV2, RSV3 | Not used for AMQP; these MUST be 0 (zero) unless defined elsewhere. |
| Opcode | %x2 indicating binary if this is the first WebSocket frame for an AMQP frame  %x0 indicating continuation if this is a WebSocket frame containing a continuation of data for an AMQP frame  WebSocket Ping and Pong Opcodes MAY also be sent as described later in this specification. |
| Mask | 1 indicating masked payload data if the sender is the WebSocket Client  0 indicating non-masked payload data if the sender is the WebSocket Server |
| Payload length | The length of the Payload Data in bytes and using 7 bits, 7+16 bits, or 7+64 bits as described in [RFC6455] section 5.2 |
| Masking-key | If Mask==1, this field MUST contain a 32-bit mask applied to the payload data  If Mask==0, this field MUST NOT be present |
| Payload data | The AMQP frame payload |

### Masking

As mentioned in section 2.1 of this specification, the AMQP endpoint initiating the initial WebSocket connection MUST behave as the WebSocket Client and thus MUST mask its payload data. The AMQP endpoint receiving the initial WebSocket connection MUST behave as the WebSocket Server and thus MUST NOT mask its payload. WebSocket Protocol Payload Data Masking is done by applying the 32-bit Masking-key to the Payload data. WebSockets masking is further described in **[RFC6455]** section 5.2 and 10.3.

## AMQP Frame Mapping to WebSocket Message

Each AMQP message consists of one or more AMQP frames. Each AMQP frame maps to a WebSocket message which in-turn maps to one or more WebSocket frames.

In the example below, there is an AMQP message consisting of two AMQP frames. AMQP frame 1 maps to WebSockets frame 1. AMQP frame 2 maps to WebSockets frame 2 and WebSockets frame 3:

[ AMQP frame 1 of 2 ] [ AMQP frame 2 of 2 ]

| / / \ \_\_\_\_\_

| | / \ \_\_\_\_

| / / \

[ WS frame 1 of 3] [ WS frame 2 of 3 ] [ WS frame 3 of 3 ]

A single AMQP frame MAY be split into one or more WebSocket frames, but a single WebSocket frame MUST NOT carry more than one AMQP frame.

The below table illustrates how the WebSocket FIN bit (indicating the final frame) and the WebSocket Opcode should be set depending on how many WebSocket frames are used for a single AMQP frame:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Number of WebSocket frames**  **per single AMQP frame** | **First frame Opcode** | **First frame FIN bit** | **Middle frame(s) Opcode** | **Middle frame(s) FIN bit** | **Last frame Opcode** | **Last frame FIN bit** |
| 1 | %x2 | 1 |  |  |  |  |
| 2 | %x2 | 0 |  |  | %x0 | 1 |
| >2 | %x2 | 0 | %x0 | 0 | %x0 | 1 |

Nodes receiving AMQP messages over the WebSocket protocol MUST combine one or more WebSocket frames into a single WebSocket message if indicated to do so by the WebSocket FIN and Opcode.

## Connection Keep-Alive

WebSocket Clients and Servers MAY keep their WebSocket connections open by sending periodic WebSocket “Ping” and “Pong” frames as described in **[RFC6455]** section 5.5.2.

AMQP messages (as binary payloads of WebSocket messages) MAY also be used for connection keep-alive.

1. Authentication and Security

## Raw AMQP over WebSocket

Raw AMQP messages SHOULD be sent over port 80. These MUST also have an AMQP type code of 0x00 as described in **[AMQP]** section 2.3.1.

This should use the WebSocket (“ws”) as the transport protocol and regular AMQP over WebSocket as the sub-protocol (“AMQP0100”).AMQP with SASL over WebSocket

Raw AMQP messages SHOULD be sent over port 80. These MUST also have an AMQP type code of 0x01 as described in **[AMQP]** section 2.3.1.

This should use the WebSocket (“ws”) as the transport protocol and AMQP with SASL over WebSocket as the sub-protocol (“AMQP3100”).

## Raw AMQP over Secure WebSocket

AMQP over Secure WebSocket SHOULD be sent over port 443. These MUST also have an AMQP type code of 0x00 as described in **[AMQP]** section 2.3.1.

This should use WebSocket over SSL or WebSocket Secure (“wss”) as the transport protocol and regular AMQP as the sub-protocol (“AMQP0100”).AMQP with SASL over Secure WebSocket

AMQP over Secure WebSocket SHOULD be sent over port 443. These MUST also have an AMQP type code of 0x01 as described in **[AMQP]** section 2.3.1.

This should use the WebSocket over SSL or WebSocket Secure (“wss”) as the transport protocol and AMQP with SASL over WebSocket as the sub-protocol (“AMQP3100”).No SSL Upgrade

Since the WebSocket Protocol supports usage either over a basic socket (HTTP / WS) or a secure socket (HTTPS / WSS), SSL upgrade is not supported.

## Ports 80 and 443

(This section is non-normative)

The above sections recommend using ports 80 and 443 for AMQP over WebSocket but this is not an explicit requirement.

1. Normal Closing of a Connection

In the non-error case, the AMQP connection SHOULD be closed first, followed by the WebSocket connection.

Once the AMQP closing handshake has completed, the WebSocket closing handshake should be initiated. As described in **[RFC6455]** section 5.5.1, the peer node desiring to close the connection sends a WebSocket Close frame (with Opcode 0x8). Once the other peer node receives this, it MAY finish transmitting any majority finished transmissions, and then MUST send a WebSocket Close frame (with Opcode 0x8) in return.

## Example

The below example section is non-normative.

Node A Node B

| |

| DETACH (AMQP Link from A to B) |

|------------------------------->|

| DETACH |

|<-------------------------------|

| |

| DETACH (AMQP Link from B to A) |

|<-------------------------------|

| DETACH |

|------------------------------->|

| |

|------------------------------->|

| END (AMQP Session) |

|------------------------------->|

| END |

|<-------------------------------|

| |

| CLOSE (AMQP Connection) |

|------------------------------->|

| CLOSE |

|<-------------------------------|

| |

| WebSocket Close Frame |

|------------------------------->|

| WebSocket Close Frame |

|<-------------------------------|

| |

Note: The AMQP frames with Detach, End, and Close performatives sent from one AMQP endpoint do not necessarily need to wait for the other AMQP endpoint. The could be sent in the order: Detach, Detach, End, Close; then Detach, Detach, End, Close.

Additional example detail showing AMQP Detach, End, and Close performatives can be found in **[AMQP]**.

Figure X: Example WebSocket Close Frame without masking

0x88 0x02 0x03 0xE8

(0x03 0xE8 = 1000, the WebSocket value for normal closure)

Figure Y: Example WebSocket Close Frame with unrealistic mask of 0x00 0x00

0x88 0x82 0x00 0x00 0x03 0xE8

# IANA Considerations

This specification requests IANA to register the WebSocket AMQP sub-protocol under the “WebSocket Subprotocol Name” registry with the following data:

|  |  |
| --- | --- |
| Subprotocol Identifier | AMQP0100 |
| Subprotocol Common Name | WebSocket Transport for Advanced Message Queueing Protocol (AMQP) |
| Subprotocol Definition | TBD: URL of this document (when available) |

|  |  |
| --- | --- |
| Subprotocol Identifier | AMQP3100AMQP0100 |
| Subprotocol Common Name | WebSocket Transport for Advanced Message Queueing Protocol (AMQP) with SASL authentication |
| Subprotocol Definition | TBD: URL of this document (when available) |

TODO: Register with IANA:

* Reference: <http://tools.ietf.org/html/rfc6455#section-11.5>
* Registry: <http://www.iana.org/assignments/websocket/websocket.xml>
* Registration Form: <http://www.iana.org/protocols/apply>

# # Conformance

The last numbered section in the specification must be the Conformance section. Conformance Statements/Clauses go here. [Remove # marker]

1. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

* Raphael Cohn, StormMQ
* Rob Dolin, Microsoft
* Robert Godfrey, JP Morgan
* Steve Huston, Riverace
* David Ingham, Microsoft
* Alex Kritikos, Software AG
* Rafael Schloming, RedHat
* Jakub Scholz, Deutsche Boerse AG

The following individuals were members of the OASIS Advanced Message Queueing Protocol (AMQP) Binding and Mappings (AMQP-BINDMAP) Technical Committee during the creation of this specification and their contributions are gratefully acknowledged:

* Sanjay Aiyagari, VMware, Inc.
* Matthew Arrott, Individual
* Allan Beck, JPMorgan Chase Bank, N.A.
* Mark Blair, Credit Suisse
* Laurie Bryson, JPMorgan Chase Bank, N.A.
* Raphael Cohn, Individual
* Allan Cornish, INETCO Systems Ltd.
* Rob Dolin, Microsoft
* Robert Gemmell, JPMorgan Chase Bank, N.A.
* Rob Godfrey, JPMorgan Chase Bank, N.A.
* William Henry, Red Hat
* Steve Huston, Individual
* David Ingham, Microsoft
* Ram Jeyaraman, Microsoft
* Alex Kritikos, Software AG, Inc.
* Shawn McAllister, Solace Systems
* Dale Moberg, Axway Software
* Andreas Moravec, Deutsche Boerse AG
* John O'Hara, Individual
* Jonathan Poulter, Kaazing
* Oleksandr Rudyy, JPMorgan Chase Bank, N.A.
* Rafael Schloming, Red Hat
* Jakub Scholz, Deutsche Boerse AG
* Angus Telfer, INETCO Systems Ltd.
* Wolf Tombe, US Department of Homeland Security

1. Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Editor** | **Changes Made** |
| [Rev number] | [Rev Date] | [Modified By] | [Summary of Changes] |