

JMAP  
Internet-Draft  
Intended status: Standards Track  
Expires: January 6, 2020

K. Murchison  
Fastmail  
July 5, 2019

A JSON Meta Application Protocol (JMAP) Subprotocol for WebSocket  
draft-ietf-jmap-websocket-02

Abstract

This document defines a binding for the JSON Meta Application Protocol (JMAP) over a WebSocket transport layer. The WebSocket binding for JMAP provides higher performance than the current HTTP binding for JMAP.

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 <https://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 January 6, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1.	Introduction	2
2.	Conventions Used in This Document	3
3.	Discovering Support for JMAP over WebSocket	3
4.	JMAP Subprotocol	3
4.1.	Handshake	4
4.2.	WebSocket Messages	4
4.2.1.	JMAP Requests	4
4.2.2.	JMAP Responses	5
4.2.3.	JMAP Request-level Errors	5
4.2.4.	JMAP Push Notifications	5
4.3.	Examples	6
5.	Security Considerations	10
6.	IANA Considerations	10
6.1.	Registration of the WebSocket JMAP Subprotocol	10
7.	Acknowledgments	10
8.	References	10
8.1.	Normative References	10
8.2.	Informative References	11
8.3.	URIs	11
Appendix A. Change History (To be removed by RFC Editor before publication)		11
Author's Address		12

## 1. Introduction

JMAP [I-D.ietf-jmap-core] over HTTP [RFC7235] requires that every JMAP API request be authenticated. Depending on the type of authentication used by the JMAP client and the configuration of the JMAP server, authentication could be an expensive operation both in time and resources. In such circumstances, authenticating every JMAP API request may harm performance.

The WebSocket binding for JMAP eliminates this performance hit by authenticating just the WebSocket handshake request and having those credentials remain in effect for the duration of the WebSocket connection. This binding supports JMAP API requests and responses, with optional support for push notifications.

Furthermore, the WebSocket binding for JMAP can optionally compress [RFC7692] both JMAP API requests and responses. Although compression of HTTP responses is ubiquitous, compression of HTTP requests has very low, if any deployment, and therefore isn't a viable option for JMAP API requests over HTTP.

## 2. Conventions Used in This Document

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 BCP 14 [1] [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The same terminology is used in this document as in the core JMAP specification.

## 3. Discovering Support for JMAP over WebSocket

The JMAP capabilities object is returned as part of the standard JMAP Session object (see Section 2 of [I-D.ietf-jmap-core]). Servers supporting this specification MUST add a property named "urn:ietf:params:jmap:websocket" to the capabilities object. The value of this property is an object which MUST contain the following information on server capabilities:

`websocketUrl`: "String" The URL to use for initiating a JMAP over WebSocket handshake.

`supportsWebSocketPush`: "Boolean" This is "true" if the server supports push notifications over the WebSocket, as described in Section 4.2.4.

Example:

```
"urn:ietf:params:jmap:websocket": {  
  "websocketUrl": "/jmap/ws/",  
  "supportsWebSocketPush": true  
}
```

## 4. JMAP Subprotocol

The term WebSocket subprotocol refers to an application-level protocol layered on top of a WebSocket connection. This document specifies the WebSocket JMAP subprotocol for carrying JMAP API requests, responses, and optional push notifications through a WebSocket connection. Binary data MUST NOT be uploaded or downloaded through a WebSocket JMAP connection. Binary data is handled per Section 6 of [I-D.ietf-jmap-core]) via a separate HTTP connection or stream.

#### 4.1. Handshake

The JMAP WebSocket client and JMAP WebSocket server negotiate the use of the WebSocket JMAP subprotocol during the WebSocket handshake, either via a HTTP/1.1 Upgrade request (see Section 1.3 of [RFC6455]) or a HTTP/2 Extended CONNECT request (see Section 5 of [RFC8441]).

Regardless of the method used for the WebSocket handshake, the client MUST make an authenticated [RFC7235] HTTP request on the JMAP "websocketUrl" (Section 3), and the client MUST include the value 'jmap' in the list of protocols for the 'Sec-WebSocket-Protocol' header field. The reply from the server MUST also contain 'jmap' in its corresponding 'Sec-WebSocket-Protocol' header field in order for a JMAP subprotocol connection to be established.

If a client receives a handshake response that does not include 'jmap' in the 'Sec-WebSocket-Protocol' header, then a JMAP subprotocol WebSocket connection was not established and the client MUST close the WebSocket connection.

Once the handshake has successfully completed, the WebSocket connection is established and can be used for JMAP API requests, responses, and optional push notifications. Other message types MUST NOT be transmitted over this connection.

The credentials used for authenticating the HTTP request to initiate the handshake remain in effect for the duration of the WebSocket connection.

#### 4.2. WebSocket Messages

Data frame messages in the JMAP subprotocol MUST be of the text type and contain UTF-8 encoded data. The messages MUST be in the form of a single JMAP Request object (see Section 3.2 of [I-D.ietf-jmap-core]) or JMAP WebSocketPushEnable object (see Section 4.2.4) when sent from the client to the server, and in the form of a single JMAP Response object, JSON Problem Details object, or JMAP StateChange object (see Sections 3.3, 3.5.1, and 7.1 respectively of [I-D.ietf-jmap-core]) when sent from the server to the client.

##### 4.2.1. JMAP Requests

This specification adds two extra arguments to the Request object:

@type: "String" This MUST be the string "Request".

id: "String" (default: ) A client-specified identifier for the request.

JMAP over WebSocket allows the server to process requests out of order. The client-specified identifier is used as a mechanism for the client to correlate requests and responses.

Additionally, the "maxConcurrentRequests" field in the "capabilities" object (see Section 2 of [I-D.ietf-jmap-core]) limits the number of inflight requests over the WebSocket.

#### 4.2.2. JMAP Responses

This specification adds two extra arguments to the Response object:

@type: "String" This MUST be the string "Response".

requestId: "String|null" The client-specified identifier in the corresponding request. If "null", no identifier was provided in the request.

#### 4.2.3. JMAP Request-level Errors

This specification adds two extra arguments to the Problem Details object:

@type: "String" This MUST be the string "RequestError".

requestId: "String|null" The client-specified identifier in the corresponding request. If "null", no identifier was provided in the request.

#### 4.2.4. JMAP Push Notifications

JMAP over WebSocket servers that support push notifications on the WebSocket will advertise a "supportsWebSocketPush" property with a value of "true" in the server capabilities object.

A client enables push notifications from the server by sending a WebSocketPushEnable object to the server. A WebSocketPushEnable object has the following properties:

@type: "String" This MUST be the string "WebSocketPushEnable".

dataTypes: "String[]|null" A list of data type names (e.g. "Mailbox", "Email") that the client is interested in. A StateChange notification will only be sent if the data for one of these types changes. Other types are omitted from the TypeState

object. If "null", changes will be pushed for all supported data types.

pushState: "String" Optional. The last "pushState" token that the client received from the server. Upon receipt of a "pushState" token, the server SHOULD immediately send all changes since that state token.

All push notifications take the form of a standard StateChange object (see Section 7.1 of [I-D.ietf-jmap-core]).

This specification adds one extra argument to the StateChange object:

pushState: "String" Optional. A (preferably short) string representing the state on the server for ALL of the data types in the account (not just the objects returned in this call).

#### 4.3. Examples

The following examples show WebSocket JMAP opening handshakes, a JMAP Core/echo request and response, and a subsequent closing handshake. The examples assume that the JMAP "websocketUrl" has been advertised in the JMAP Session object as "/jmap/ws/". Note that folding of header fields is for editorial purposes only.

WebSocket JMAP connection via HTTP/1.1 with push notifications enabled:

[[ From Client ]]

[[ From Server ]]

```
GET /jmap/ws/ HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Authorization: Basic Zm9vOmJhcg==
Sec-WebSocket-Key:
  dGhlIHhnbXBsZSBub25jZQ==
Sec-WebSocket-Protocol: jmap
Sec-WebSocket-Version: 13
Origin: http://www.example.com
```

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

```
[WebSocket connection established]
```

```
WS_DATA
```

```
{
  "@type": "WebSocketPushEnable",
  "dataTypes": [ "Mailbox", "Email" ],
  "pushState": "aaa"
}
```

```
WS_DATA
```

```
{
  "@type": "StateChange",
  "changed": {
    "a456": {
      "Email": "d35ecb040aab"
    }
  },
  "pushState": "bbb"
}
```

```
WS_DATA
```

```
{
  "@type": "Request",
  "id": "R1",
  "using": [ "urn:ietf:params:jmap:core" ],
  "methodCalls": [
    [
      "Core/echo", {
        "hello": true,
        "high": 5
      },
      "b3ff"
    ]
  ]
}
```

```
WS_DATA
```

```
{
  "@type": "Response",
  "requestId": "R1",
  "methodResponses": [
    [
      "Core/echo", {
        "hello": true,
        "high": 5
      },
      "b3ff"
    ]
  ]
}
```

```
]
}
```

```
WS_DATA
The quick brown fox jumps
over the lazy dog.
```

```
WS_DATA
{
  "@type": "RequestError",
  "requestId": "null",
  "type":
"urn:ietf:params:jmap:error:notJSON",
  "status": 400,
  "detail":
"The request did not parse as I-JSON."
}
```

```
WS_DATA
{
  "@type": "StateChange",
  "changed": {
    "a123": {
      "Mailbox": "0af7a512ce70"
    }
  }
  "pushState": "ccc"
}
```

```
WS_CLOSE
```

```
WS_CLOSE
```

```
[WebSocket connection closed]
```



WebSocket JMAP connection on a HTTP/2 stream which also negotiates compression [RFC7692]:

[[ From Client ]]

[[ From Server ]]

SETTINGS

SETTINGS\_ENABLE\_CONNECT\_PROTOCOL = 1

HEADERS + END\_HEADERS

:method = CONNECT

:protocol = websocket

:scheme = https

:path = /jmap/ws/

:authority = server.example.com

authorization = Basic Zm9vOmJhcg==

sec-websocket-protocol = jmap

sec-websocket-version = 13

sec-websocket-extensions =

permessage-deflate

origin = http://www.example.com

HEADERS + END\_HEADERS

:status = 200

sec-websocket-protocol = jmap

sec-websocket-extensions =

permessage-deflate

[WebSocket connection established]

DATA

WS\_DATA

[compressed text]

DATA

WS\_DATA

[compressed text]

...

DATA + END\_STREAM

WS\_CLOSE

DATA + END\_STREAM

WS\_CLOSE

[WebSocket connection closed]

[HTTP/2 stream closed]

## 5. Security Considerations

The security considerations for both WebSocket (see Section 10 of [RFC6455]) and JMAP (see Section 8 of [I-D.ietf-jmap-core]) apply to the WebSocket JMAP subprotocol.

## 6. IANA Considerations

### 6.1. Registration of the WebSocket JMAP Subprotocol

This specification requests IANA to register the WebSocket JMAP subprotocol under the "WebSocket Subprotocol Name" Registry with the following data:

Subprotocol Identifier: JMAP

Subprotocol Common Name: WebSocket Transport for JMAP (JSON Meta Application Protocol)

Subprotocol Definition: RFCXXXX (this document)

## 7. Acknowledgments

The author would like to thank the following individuals for contributing their ideas and support for writing this specification: Neil Jenkins, Robert Mueller, and Chris Newman.

## 8. References

### 8.1. Normative References

- [I-D.ietf-jmap-core] Jenkins, N. and C. Newman, "JSON Meta Application Protocol", draft-ietf-jmap-core-17 (work in progress), March 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, December 2011, <<https://www.rfc-editor.org/info/rfc6455>>.

- [RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <<https://www.rfc-editor.org/info/rfc7235>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8441] McManus, P., "Bootstrapping WebSockets with HTTP/2", RFC 8441, DOI 10.17487/RFC8441, September 2018, <<https://www.rfc-editor.org/info/rfc8441>>.

## 8.2. Informative References

- [I-D.ietf-jmap-mail] Jenkins, N. and C. Newman, "JMAP (JSON Meta Application Protocol) for Mail", draft-ietf-jmap-mail-16 (work in progress), March 2019.
- [RFC7692] Yoshino, T., "Compression Extensions for WebSocket", RFC 7692, DOI 10.17487/RFC7692, December 2015, <<https://www.rfc-editor.org/info/rfc7692>>.

## 8.3. URIs

- [1] <https://tools.ietf.org/html/bcp14>

## Appendix A. Change History (To be removed by RFC Editor before publication)

### Changes since ietf-01:

- o Changed 'wsURL' to 'websocketUrl' and removed push query option.
- o Added 'supportsWebSocketPush' capability.
- o Added '@type' argument to Request object.
- o Added 'WebSocketPushEnable' object.
- o Added 'pushState' argument to StateChange object.
- o Updated example.
- o Minor Editorial changes.

### Changes since ietf-00:

- o Added text describing advertisement of and selection of optional push notifications.
- o Minor Editorial changes.

Changes since murchison-02:

- o Renamed as a JMAP WG document.
- o Allow out of order processing.
- o Allow push notifications.
- o Modified examples.
- o Add Security Considerations text.
- o Minor Editorial changes.

Changes since murchison-01:

- o Updated WebSocket over HTTP/2 reference to RFC8144.

Changes since murchison-00:

- o Fleshed out section on discovery of support for JMAP over WebSocket.
- o Allow JSON Problem Details objects to be returned by the server for toplevel errors.
- o Mentioned the ability to compress JMAP API requests.
- o Minor Editorial changes.

#### Author's Address

Kenneth Murchison  
Fastmail US LLC  
1429 Walnut Street - Suite 1201  
Philadelphia, PA 19102  
USA

Email: [murch@fastmailteam.com](mailto:murch@fastmailteam.com)  
URI: <http://www.fastmail.com/>