ï»¿

        A JSON Meta Application Protocol (JMAP) Subprotocol for WebSocket

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 is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF).  It represents the consensus of the IETF community.  It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG).  Further information on
   Internet Standards is available in Section 2 of RFC 7841.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   https://www.rfc-editor.org/info/rfc8887.

Table of Contents

1.  Introduction

    JMAP [RFC8620] 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, reauthenticating for every JMAP
    API request may harm performance.

    The WebSocket [RFC6455] 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 [RFC2119] [RFC8174] when, and only when, they appear in all
    capitals, as shown here.

    This document uses the terminology defined in the core JMAP
    specification [RFC8620].

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 [RFC8620]).  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 that MUST contain the following
    information on server capabilities:

    *  url: "String"

       The wss-URI (see Section 3 of [RFC6455]) to use for initiating a
       JMAP-over-WebSocket handshake (the "WebSocket URL endpoint"
       colloquially).

    *  supportsPush: "Boolean"

       This is true if the server supports push notifications over the
       WebSocket, as described in Section 4.3.5.

    Example:

    "urn:ietf:params:jmap:websocket": {
      "url": "wss://server.example.com/jmap/ws/",
      "supportsPush": 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 is handled per Section 6 of
    [RFC8620] (via a separate HTTP connection or stream) or per a future
    extension to JMAP or this specification.

## 4.1.  Authentication

A JMAP WebSocket connection is authenticated by presenting a user's
credentials in the HTTP request [RFC7235] that initiates the
WebSocket handshake.  See Section 8.2 of [RFC8620] for
recommendations regarding the selection of HTTP authentication
schemes.

## 4.2.  Handshake

The JMAP WebSocket client and JMAP WebSocket server negotiate the use
of the WebSocket JMAP subprotocol during the WebSocket handshake,
either via an HTTP/1.1 Upgrade request (see Section 4 of [RFC6455])
or an HTTP/2 Extended CONNECT request (see Section 5 of [RFC8441]).
The WebSocket JMAP subprotocol is also intended to run over future
bindings of HTTP (e.g., HTTP/3) provided that there is a defined
mechanism for performing a WebSocket handshake over that binding.

Regardless of the method used for the WebSocket handshake, the client
MUST first perform a TLS handshake on a JMAP WebSocket URL endpoint
(Section 3) having the "wss://" scheme (WebSocket over TLS) in
accordance with the requirements of running the particular binding of
HTTP over TLS (see [RFC2818] and Section 4.1 of [RFC6455] for
HTTP/1.1 and Section 9.2 of [RFC7540] for HTTP/2).  If the TLS
handshake fails, the client MUST close the connection.  Otherwise,
the client MUST make an authenticated HTTP request [RFC7235] on the
encrypted connection and 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 a corresponding "Sec-
WebSocket-Protocol" header field with a value of "jmap" in order for
a JMAP subprotocol connection to be established.

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.  If the authentication credentials for the user expire,
the server can either treat subsequent requests as if they are
unauthenticated or close the WebSocket connection.  In the latter
case, the server MAY send a Close frame with a status code of 1008
(Policy Violation), as defined in Section 7.4.1 of [RFC6455].

## 4.3.  WebSocket Messages

Data frame messages in the JMAP subprotocol MUST be text frames and
contain UTF-8 encoded data.  The messages MUST be in the form of a
single JMAP Request object (see Section 3.3 of [RFC8620]), JMAP
WebSocketPushEnable object (see Section 4.3.5.2), or JMAP
WebSocketPushDisable object (see Section 4.3.5.3) when sent from the
client to the server, and MUST be in the form of a single JMAP
Response object, JSON Problem Details object, or JMAP StateChange
object (see Sections 3.4, 3.6.1, and 7.1 of [RFC8620], respectively)
when sent from the server to the client.

Note that fragmented WebSocket messages (split over multiple text
frames) MUST be coalesced prior to parsing them as JSON objects.

### 4.3.1.  Handling Invalid Data

If a client or server receives a binary frame, the endpoint can
either ignore the frame or close the WebSocket connection.  In the
latter case, the endpoint MAY send a Close frame with a status code
of 1003 (Unsupported Data), as defined in Section 7.4.1 of [RFC6455].

If a client receives a message that is not in the form of a JSON
Problem Details object, a JMAP Response object, or a JMAP StateChange

object, the client can either ignore the message or close the
WebSocket connection.  In the latter case, the endpoint MAY send a
Close frame with a status code of 1007 (Invalid frame payload data),
as defined in Section 7.4.1 of [RFC6455].

A server MUST return an appropriate JSON Problem Details object
(Section 4.3.4) for any request-level errors (e.g., an invalid JMAP
object, an unsupported capability or method call, or exceeding a
server request limit).

4.3.2.  JMAP Requests

The specification extends the Request object with two additional
arguments when used over a WebSocket:

*  @type: "String"

   This MUST be the string "Request".

*  id: "String" (optional)

   A client-specified identifier for the request to be echoed back in
   the response to this 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" limit in the "capabilities"
object (see Section 2 of [RFC8620]) also applies to requests made on
the WebSocket connection.  When using the WebSocket JMAP subprotocol
over a binding of HTTP that allows multiplexing of requests (e.g.,
HTTP/2), this limit applies to the sum of requests made on both the
JMAP API endpoint and the WebSocket connection.

4.3.3.  JMAP Responses

The specification extends the Response object with two additional
arguments when used over a WebSocket:

*  @type: "String"

   This MUST be the string "Response".

*  requestId: "String" (optional; MUST be returned if an identifier
   is included in the request)

   The client-specified identifier in the corresponding request.

4.3.4.  JMAP Request-Level Errors

The specification extends the Problem Details object for request-
level errors (see Section 3.6.1 of [RFC8620]) with two additional
arguments when used over a WebSocket:

*  @type: "String"

   This MUST be the string "RequestError".

*  requestId: "String" (optional; MUST be returned if given in the
   request)

   The client-specified identifier in the corresponding request.

4.3.5.  JMAP Push Notifications

JMAP-over-WebSocket servers that support push notifications on the
WebSocket will advertise a "supportsPush" property with a value of
true in the "urn:ietf:params:jmap:websocket" server capabilities
object.

### 4.3.5.1. Notification Format

All push notifications take the form of a standard StateChange object (see Section 7.1 of [RFC8620]).

The specification extends the StateChange object with one additional argument when used over a WebSocket:

* pushState: "String" (optional)

  A (preferably short) string that encodes the entire server state visible to the user (not just the objects returned in this call).

  The purpose of the "pushState" token is to allow a client to immediately get any changes that occurred while it was disconnected (see Section 4.3.5.2).  If the server does not support "pushState" tokens, the client will have to issue a series of "/changes" requests (see Section 5.2 of [RFC8620]) upon reconnection to update its state to match that of the server.

### 4.3.5.2. Enabling Notifications

A client enables push notifications from the server for the current connection 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" or "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.

### 4.3.5.3. Disabling Notifications

A client disables push notifications from the server for the current connection by sending a WebSocketPushDisable object to the server.  A WebSocketPushDisable object has the following property:

* @type: "String"

  This MUST be the string "WebSocketPushDisable".

### 4.4. 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 WebSocket URL endpoint has been advertised in the JMAP Session object as having a path of "/jmap/ws/" and that TLS negotiation has already succeeded.  Note that folding of header fields is for editorial purposes only.

WebSocket JMAP connection via HTTP/1.1 with push notifications for mail [RFC8621] is enabled.  This example assumes that the client has cached pushState "aaa" from a previous connection.

[[ From Client ]]                    [[ From Server ]]

GET /jmap/ws/ HTTP/1.1
Host: server.example.com

```
Upgrade: websocket
Connection: Upgrade
Authorization: Basic Zm9vOmJhcg==
Sec-WebSocket-Key:
  dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Protocol: jmap
Sec-WebSocket-Version: 13
Origin: https://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": {
                                  "Mailbox": "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."
                                   }
```

[A new email is received]

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

```
WS_CLOSE
```

```
                                   WS_CLOSE
```

[WebSocket connection closed]

WebSocket JMAP connection on an HTTP/2 stream that 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
origin: https://example.com
authorization = Basic Zm9vOmJhcg==
sec-websocket-protocol = jmap
sec-websocket-version = 13
sec-websocket-extensions =
  permessage-deflate
origin = https://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 [RFC8620]) apply to the
WebSocket JMAP subprotocol.  Specific security considerations are
described below.

## 5.1.  Connection Confidentiality and Integrity

To ensure the confidentiality and integrity of data sent and received
via JMAP over WebSocket, the WebSocket connection MUST use TLS 1.2
[RFC5246] or later, following the recommendations in BCP 195
[RFC7525].  Servers SHOULD support TLS 1.3 [RFC8446] or later.

## 5.2.  Non-browser Clients

JMAP over WebSocket can be used by clients both running inside and
outside of a web browser.  As such, the security considerations in
Sections 10.2 and 10.1 of [RFC6455] apply to those respective
environments.

## 6.  IANA Considerations

## 6.1.  Registration of the WebSocket JMAP Subprotocol

Per this specification, IANA has registered the following in the
"WebSocket Subprotocol Name Registry" within the "WebSocket Protocol
Registries".

Subprotocol Identifier:  jmap

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

Subprotocol Definition:  RFC 8887

## 7.  References

## 7.1.  Normative References

[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>.

[RFC2818]   Rescorla, E., "HTTP Over TLS", RFC 2818,
            DOI 10.17487/RFC2818, May 2000,
            <https://www.rfc-editor.org/info/rfc2818>.

[RFC5246]   Dierks, T. and E. Rescorla, "The Transport Layer Security
            (TLS) Protocol Version 1.2", RFC 5246,
            DOI 10.17487/RFC5246, August 2008,
            <https://www.rfc-editor.org/info/rfc5246>.

[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>.

   [RFC7525]  Sheffer, Y., Holz, R., and P. Saint-Andre,
              "Recommendations for Secure Use of Transport Layer
              Security (TLS) and Datagram Transport Layer Security
              (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May
              2015, <https://www.rfc-editor.org/info/rfc7525>.

   [RFC7540]  Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext
              Transfer Protocol Version 2 (HTTP/2)", RFC 7540,
              DOI 10.17487/RFC7540, May 2015,
              <https://www.rfc-editor.org/info/rfc7540>.

   [RFC7692]  Yoshino, T., "Compression Extensions for WebSocket",
              RFC 7692, DOI 10.17487/RFC7692, December 2015,
              <https://www.rfc-editor.org/info/rfc7692>.

   [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>.

   [RFC8446]  Rescorla, E., "The Transport Layer Security (TLS) Protocol
              Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018,
              <https://www.rfc-editor.org/info/rfc8446>.

   [RFC8620]  Jenkins, N. and C. Newman, "The JSON Meta Application
              Protocol (JMAP)", RFC 8620, DOI 10.17487/RFC8620, July
              2019, <https://www.rfc-editor.org/info/rfc8620>.

## 7.2.  Informative References

   [RFC8621]  Jenkins, N. and C. Newman, "The JSON Meta Application
              Protocol (JMAP) for Mail", RFC 8621, DOI 10.17487/RFC8621,
              August 2019, <https://www.rfc-editor.org/info/rfc8621>.

## Acknowledgments

## Author's Address

   Kenneth Murchison
   Fastmail US LLC
   1429 Walnut Street, Suite 1201
   Philadelphia, PA 19102
   United States of America

   Email: murch@fastmailteam.com
   URI:   http://www.fastmail.com/