

JMAP
Internet-Draft
Intended status: Standards Track
Expires: October 31, 2019

N. Jenkins
FastMail
M. Douglass
Spherical Cow Group
April 29, 2019

JMAP for Calendars
draft-ietf-jmap-calendars-00

Abstract

This document specifies a data model for synchronising calendar data with a server using 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 October 31, 2019.

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
1.1. Notational conventions	2
1.2. The Date datatypes	3
1.3. Terminology	3
1.4. Addition to the capabilities object	3
2. Calendars	3
2.1. Calendar/get	5
2.2. Calendar/changes	5
2.3. Calendar/set	5
3. Calendar events	5
3.1. CalendarEvent/get	6
3.2. CalendarEvent/changes	6
3.3. CalendarEvent/set	6
3.4. CalendarEvent/copy	6
3.5. CalendarEvent/query	7
3.5.1. Filtering	7
3.5.2. Sorting	8
3.6. CalendarEvent/queryChanges	8
4. Security considerations	8
5. IANA considerations	9
5.1. JMAP capability registration for "calendars"	9
6. Normative References	9
Authors' Addresses	9

1. Introduction

JMAP ([I-D.ietf-jmap-core] - JSON Meta Application Protocol) is a generic protocol for synchronising data, such as mail, calendars or contacts, between a client and a server. It is optimised for mobile and web environments, and aims to provide a consistent interface to different data types.

This specification defines a data model for synchronising calendar data between a client and a server using JMAP.

1.1. Notational conventions

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.

Type signatures, examples and property descriptions in this document follow the conventions established in section 1.1 of [I-D.ietf-jmap-core].

Object properties may also have a set of attributes defined along with the type signature. These have the following meanings:

- o `*server-set*`: Only the server can set the value for this property. The client **MUST NOT** send this property when creating a new object of this type.
- o `*immutable*`: The value **MUST NOT** change after the object is created.
- o `*default*`: (This is followed by a JSON value). The value that will be used for this property if it is omitted in an argument, or when creating a new object of this type.

Data types defined in the core specification are used in this document.

1.2. The Date datatypes

Where "LocalDate" is given as a type, it means a string in the same format as "Date", but with the `_time-offset_` omitted from the end. The interpretation in absolute time depends upon the time zone for the event, which may not be a fixed offset (for example when daylight saving time occurs). For example, `"2014-10-30T14:12:00"`.

1.3. Terminology

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

1.4. Addition to the capabilities object

The capabilities object is returned as part of the standard JMAP Session object; see the JMAP spec. Servers supporting `_this_` specification **MUST** add a property called `"urn:ietf:params:jmap:calendars"` to the capabilities object.

The value of this property is an empty object in both the JMAP session `_capabilities_` property and an account's `_accountCapabilities_` property.

2. Calendars

A Calendar is a named collection of events. All events are associated with one, and only one, calendar.

A `*Calendar*` object has the following properties:

- o `*id*`: "Id" (immutable; server-set) The id of the calendar.
- o `*name*`: "String" The user-visible name of the calendar. This may be any UTF-8 string of at least 1 character in length and maximum 255 octets in size.
- o `*color*`: "String" Any valid CSS color value. The color to be used when displaying events associated with the calendar. The color SHOULD have sufficient contrast to be used as text on a white background.
- o `*sortOrder*`: "UnsignedInt" (default: 0) Defines the sort order of calendars when presented in the client's UI, so it is consistent between devices. The number MUST be an integer in the range $0 \leq \text{sortOrder} < 2^{31}$. A calendar with a lower order should be displayed before a calendar with a higher order in any list of calendars in the client's UI. Calendars with equal order should be sorted in alphabetical order by name. The sorting should take into locale-specific character order convention.
- o `*isVisible*`: "Boolean" (default: true) Should the calendar's events be displayed to the user at the moment?
- o `*mayReadFreeBusy*`: "Boolean" (server-set) The user may read the free-busy information for this calendar. In JMAP terms, this means the user may use this calendar as part of a filter in a `_CalendarEvent/query_` call, however unless `"mayRead == true"`, the events returned for this calendar will only contain free-busy information, and be stripped of any other data. This property MUST be `"true"` if `_mayRead_` is `"true"`.
- o `*mayReadItems*`: "Boolean" (server-set) The user may fetch the events in this calendar. In JMAP terms, this means the user may use this calendar as part of a filter in a `_CalendarEvent/query_` call
- o `*mayAddItems*`: "Boolean" (server-set) The user may add events to this calendar. In JMAP terms, this means the user may call `_CalendarEvent/set_` to create new events in this calendar or move existing events into this calendar from another calendar. This property MUST be `"false"` if the account to which this calendar belongs has the `_isReadOnly_` property set to `"true"`.
- o `*mayModifyItems*`: "Boolean" (server-set) The user may edit events in this calendar by calling `_CalendarEvent/set_` with the `_update_` argument referencing events in this collection. This property MUST be `"false"` if the account to which this calendar belongs has the `_isReadOnly_` property set to `"true"`.

- o `*mayRemoveItems*`: "Boolean" (server-set) The user may remove events from this calendar by calling `_CalendarEvent/set_` with the `_destroy_` argument referencing events in this collection, or by updating their `_calendarId_` property to a different calendar. This property MUST be "false" if the account to which this calendar belongs has the `_isReadOnly_` property set to "true".
- o `*mayRename*`: "Boolean" (server-set) The user may rename the calendar. This property MUST be "false" if the account to which this calendar belongs has the `_isReadOnly_` property set to "true".
- o `*mayDelete*`: "Boolean" (server-set) The user may delete the calendar itself. This property MUST be "false" if the account to which this calendar belongs has the `_isReadOnly_` property set to "true".

2.1. Calendar/get

Standard `"/get"` method as described in [I-D.ietf-jmap-core] section 5.1. The `_ids_` argument may be "null" to fetch all at once.

2.2. Calendar/changes

Standard `"/changes"` method as described in [I-D.ietf-jmap-core] section 5.2.

2.3. Calendar/set

Standard `"/set"` method as described in [I-D.ietf-jmap-core] section 5.3.

A calendar MAY be deleted that is currently associated with one or more events. In this case, the events belonging to this calendar MUST also be deleted. Conceptually, this MUST happen prior to the calendar itself being deleted, and MUST generate a `*push*` event that modifies the state of the `_CalendarEvent_` type for the account.

3. Calendar events

A `*CalendarEvent*` object contains information about an event, or recurring series of events, that takes place at a particular time. It is a `JSEvent` object, as defined in [I-D.ietf-calext-jscalendar], with the following additional properties:

- o `*id*`: "Id" The id of the event. This property is immutable.
- o `*calendarId*`: "Id" The id of the calendar this event belongs to.

- o `*participantId*`: "Id|null" The id of the participant in the `_participants_` object which corresponds to the account this event is in.

3.1. CalendarEvent/get

Standard `"/get"` method as described in [I-D.ietf-jmap-core] section 5.1.

3.2. CalendarEvent/changes

Standard `"/changes"` method as described in [I-D.ietf-jmap-core] section 5.2

3.3. CalendarEvent/set

Standard `"/set"` method as described in [I-D.ietf-jmap-core] section 5.3.

When an event is created, updated or destroyed, the server MUST also ensure the following:

- o Any alerts are scheduled/cancelled correctly.
- o If there is a `_participantId_`, and the corresponding participant has a `_role_` of "owner":
 - * If an event is created/updated: send a REQUEST iMIP email with the event as an ICS attachment to all participants that are not "you".
 - * When an event is destroyed, if it is in the future, then email all participants other than you the appropriate iMIP email to inform them that the event has been cancelled. If it is in the past, the server SHOULD NOT send a message.
- o If there is a `_participantId_`, and the corresponding participant does not have a `_role_` of "owner", and the `_scheduleStatus_` is updated for this participant, send the appropriate iMIP email to the `_replyTo_` address.

3.4. CalendarEvent/copy

Standard `"/copy"` method as described in [I-D.ietf-jmap-core] section 5.4.

3.5. CalendarEvent/query

Standard `/query` method as described in [I-D.ietf-jmap-core] section 5.5.

3.5.1. Filtering

A `*FilterCondition*` object has the following properties:

- o `*inCalendars*`: `"Id[]|null"` A list of calendar ids. An event must be in ANY of these calendars to match the condition.
- o `*after*`: `"UTCDate|null"` The end of the event, or any recurrence of the event, in UTC time must be after this date to match the condition.
- o `*before*`: `"UTCDate|null"` The start of the event, or any recurrence of the event, in UTC time must be before this date to match the condition.
- o `*text*`: `"String|null"` Looks for the text in the `_title_`, `_description_`, `_locations_` (matching name/description), or `_participants_` (matching name/email) properties of the event or any recurrence of the event.
- o `*title*`: `"String|null"` Looks for the text in the `_title_` property of the event, or the overridden `_title_` property of a recurrence.
- o `*description*`: `"String|null"` Looks for the text in the `_description_` property of the event, or the overridden `_description_` property of a recurrence.
- o `*location*`: `"String|null"` Looks for the text in the `_locations_` property of the event (matching name/description of a location), or the overridden `_locations_` property of a recurrence.
- o `*owner*`: `"String|null"` Looks for the text in the name or email fields of a participant in the `_participants_` property of the event, or the overridden `_participants_` property of a recurrence, where the participant has a role of "owner".
- o `*attendee*`: `"String|null"` Looks for the text in the name or email fields of a participant in the `_participants_` property of the event, or the overridden `_participants_` property of a recurrence, where the participant has a role of "attendee".

If zero properties are specified on the `FilterCondition`, the condition MUST always evaluate to `"true"`. If multiple properties are

specified, ALL must apply for the condition to be "true" (it is equivalent to splitting the object into one-property conditions and making them all the child of an AND filter operator).

The exact semantics for matching "String" fields is **deliberately not defined** to allow for flexibility in indexing implementation, subject to the following:

- o Text SHOULD be matched in a case-insensitive manner.
- o Text contained in either (but matched) single or double quotes SHOULD be treated as a **phrase search**, that is a match is required for that exact sequence of words, excluding the surrounding quotation marks. Use "\"", "\"'" and "\"\" to match a literal "\"", "'" and "\" respectively in a phrase.
- o Outside of a phrase, white-space SHOULD be treated as dividing separate tokens that may be searched for separately in the event, but MUST all be present for the event to match the filter.
- o Tokens MAY be matched on a whole-word basis using stemming (so for example a text search for "bus" would match "buses" but not "business").

3.5.2. Sorting

The following properties MUST be supported for sorting:

- o start
- o uid

3.6. CalendarEvent/queryChanges

Standard "/queryChanges" method as described in [I-D.ietf-jmap-core] section 5.6.

4. Security considerations

All security considerations of JMAP ([I-D.ietf-jmap-core]) apply to this specification. Additional considerations specific to the data types and functionality introduced by this document are described in the following subsections.

TODO

5. IANA considerations

5.1. JMAP capability registration for "calendars"

IANA will register the "calendars" JMAP Capability as follows:

Capability Name: "urn:ietf:params:jmap:calendars"

Specification document: this document

Intended use: common

Change Controller: IETF

Security and privacy considerations: this document, section TODO

6. Normative References

[I-D.ietf-calext-jscalendar]

Jenkins, N. and R. Stepanek, "JSCalendar: A JSON representation of calendar data", draft-ietf-calext-jscalendar-12 (work in progress), March 2019.

[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>>.

[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>>.

Authors' Addresses

Neil Jenkins
FastMail
PO Box 234, Collins St West
Melbourne VIC 8007
Australia

Email: neilj@fastmailteam.com
URI: <https://www.fastmail.com>

Michael Douglass
Spherical Cow Group
226 3rd Street
Troy NY 12180
United States of America

Email: mdouglass@sphericalcowgroup.com
URI: <http://sphericalcowgroup.com>

JMAP
Internet-Draft
Intended status: Standards Track
Expires: September 8, 2019

R. Ouazana, Ed.
Linagora
March 7, 2019

Handling Message Disposition Notification with JMAP
draft-ietf-jmap-mdn-01

Abstract

This document specifies a data model for handling [RFC8098] MDN messages with a server using 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 September 8, 2019.

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
1.1. Notational conventions	3
1.2. Terminology	3
1.3. Addition to the capabilities object	3
2. MDN	3
3. Methods added to the EmailSubmission object	4
3.1. EmailSubmission/sendMDN	4
3.2. EmailSubmission/parseMDN	5
4. Samples	6
4.1. Sending an MDN for a received email	6
4.2. Asking for MDN when sending an email	7
4.3. Parsing a received MDN	8
5. IANA Considerations	8
5.1. JMAP Capability Registration for "mdn"	8
5.2. Registration of JMAP keyword '\$MDNSent'	9
6. Security considerations	9
7. References	10
7.1. Normative References	10
7.2. Informative References	10
Author's Address	10

1. Introduction

JMAP ([I-D.ietf-jmap-core] - JSON Meta Application Protocol) is a generic protocol for synchronising data, such as mail, calendars or contacts, between a client and a server. It is optimised for mobile and web environments, and aims to provide a consistent interface to different data types.

MDN are defined in [RFC8098] and are used as "read receipts", "acknowledgements", or "receipt notifications".

A client can have to deal with MDN in different ways:

1. When receiving an email, an MDN can be sent to the sender. This specification defines an EmailSubmission/sendMDN method to cover this case.
2. When sending an email, an MDN can be requested. This must be done with the help of a header, and is already specified by [RFC8098] and can already be handled by [I-D.ietf-jmap-mail] this way.
3. When receiving an MDN, the MDN could be related to an existing sent mail. This is already covered by [I-D.ietf-jmap-mail] in the EmailSubmission object. Client could want to display

detailed information about a received MDN. This specification defines a `EmailSubmission/parseMDN` method to cover this case.

1.1. Notational conventions

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.

Type signatures, examples and property descriptions in this document follow the conventions established in section 1.1 of [I-D.ietf-jmap-core]. Data types defined in the core specification are also used in this document.

Servers MUST support all properties specified for the new data types defined in this document.

1.2. Terminology

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

1.3. Addition to the capabilities object

The capabilities object is returned as part of the standard JMAP Session object; see the JMAP spec. Servers supporting `_this_` specification MUST add a property called `"urn:ietf:params:jmap:mdn"` to the capabilities object.

2. MDN

An `*MDN*` object has the following properties:

- o `*forEmailId*`: "String" Email Id of the received email this MDN is relative to.
- o `*subject*`: "String|null" Subject used as "Subject" header for this MDN.
- o `*textBody*`: "String|null" Human readable part of the MDN, as plain text.
- o `*reportingUA*`: "String|null" Name of the MUA creating this MDN. It is used to build the MDN Report part of the MDN.

- o `*disposition*`: "Disposition" Object containing the diverse MDN disposition options.
- o `*mdnGateway*`: "String|null" (server-set) Name of the gateway or MTA that translated a foreign (non-Internet) message disposition notification into this MDN.
- o `*originalRecipient*`: "String|null" (server-set) Original recipient address as specified by the sender of the message for which the MDN is being issued.
- o `*finalRecipient*`: "String" (server-set) Recipient for which the MDN is being issued.
- o `*originalMessageID*`: "String|null" (server-set) Message-ID (the [RFC5322] header field, not the JMAP Id) of the message for which the MDN is being issued.
- o `*error*`: "String[]|null" (server-set) Additional information in the form of text messages when the "error" disposition modifier appears.
- o `*extensionFields*`: "String[String]|null" (server-set) Object where keys are extension-field names and values are extension-field values.

A `*Disposition*` object has the following properties:

- o `*actionMode*`: "String" This MUST be one of the following strings: "manual-action" / "automatic-action"
- o `*sendingMode*`: "String" This MUST be one of the following strings: "MDN-sent-manually" / "MDN-sent-automatically"
- o `*type*`: "String" This MUST be one of the following strings: "deleted" / "dispatched" / "displayed" / "processed"

See [RFC8098] for the exact meaning of these different fields.

3. Methods added to the EmailSubmission object

3.1. EmailSubmission/sendMDN

The EmailSubmission/sendMDN method generates and sends an [RFC5322] message from an MDN object.

It takes the following arguments:

- o `*accountId*`: "Id" The id of the account to use.
- o `*mdns*`: "String[MDN]" A map of creation id (client specified) to MDN objects

If the `_forEmailId_`, `_subject_`, `_textBody_`, `_reportingUA_`, `_disposition_` properties are invalid (e.g. missing, wrong type, id not found), the submission creation is rejected with a standard "invalidProperties" SetError and no email is sent. Any other error usually sent by "EmailSubmission/set" for `*create*` can be returned by this method.

The client SHOULD NOT issue a sendMDN request if the message has the "\$MDNSent" keyword set. In this case, the server MUST reject the submission with a standard "forbiddenToSend" SetError.

When sending the MDN, the server is in charge of generating the `_originalRecipient_`, `_finalRecipient_` and `_originalMessageID_` fields accordingly to the [RFC8098] specification.

The response has the following arguments:

- o `*accountId*`: "String" The id of the account used for this call.
- o `*created*`: "String[EmailSubmission]" A map of creation id (client-specified) to an email sent from the referenced properties. The returned EmailSubmission is similar to a call to a standard "EmailSubmission/set" with a `_create_` parameter.
- o `*notCreated*`: "String[SetError]" A map of creation id to a SetError object for each Email that failed to be sent. The possible errors are defined above.

For each "forEmailId" whose EmailSubmission where created, the server MUST add a "\$MDNSent" keyword to the email.

3.2. EmailSubmission/parseMDN

This method allows you to parse blobs as [RFC5322] messages to get MDN objects. This can be used to parse and get detailed information about blobs referenced in the `_mdnBlobIds_` of the EmailSubmission object, or any email the client could expect to be an MDN.

The `_forEmailId_` property can be null or missing if the `_originalMessageID_` property is missing or not referencing an existing email.

The Email/parse method takes the following arguments:

- o `*accountId*`: "String" The id of the account to use.
- o `*blobIds*`: "Id[]" The ids of the blobs to parse.

The response has the following arguments:

- o `*accountId*`: "Id" The id of the account used for the call.
- o `*parsed*`: "Id[MDN]|null" A map of blob id to parsed MDN representation for each successfully parsed blob, or null if none.
- o `*notParsable*`: "Id[]|null" A list of ids given that corresponded to blobs that could not be parsed as MDNs, or null if none.
- o `*notFound*`: "Id[]|null" A list of blob ids given that could not be found, or null if none.

4. Samples

4.1. Sending an MDN for a received email

A client can use the following request to send an MDN back to the sender:

```
[["EmailSubmission/sendMDN", {
  "accountId": "uel50411c",
  "mdns": {
    "k1546": {
      "forEmailId": "Md45b47b4877521042cec0938",
      "subject": "Read receipt for: World domination",
      "textBody": "This receipt shows that the email has been
        displayed on your recipient's computer. There is no
        guaranty it has been read or understood.",
      "reportingUA": "linagora.com; OpenPaaS",
      "disposition": {
        "actionMode": "manual-action",
        "sendingMode": "MDN-sent-manually",
        "type": "displayed"
      }
    }
  }
}], "0" ]]
```

If the email id matches an existing email without the "\$MDNSent" keyword, the server can answer:


```
[["EmailSubmission/sendMDN", {
  "accountId": "ue150411c",
  "oldState": "012421s6-8nrq-4ps4-n0p4-9330r951ns21",
  "newState": "355421f6-8aed-4cf4-a0c4-7377e951af36",
  "created": {
    "k1546": {
      "id": "73191acf-ed7-4008-bde2-57cd9ed3c559"
    }
  }
}], "0" ],
```

4.2. Asking for MDN when sending an email

This is done with the [I-D.ietf-jmap-mail] "Email/set" `_create_` method.

```
[["Email/set", {
  "accountId": "ue150411c",
  "create": {
    "k1546": {
      "mailboxIds": {
        "2ealca41b38e": true
      },
      "keywords": {
        "$seen": true,
        "$draft": true
      },
      "from": [{
        "name": "Joe Bloggs",
        "email": "joe@example.com"
      }],
      "to": [{
        "name": "John",
        "email": "john@example.com"
      }],
      "headers": [{
        "name": "Disposition-Notification-To",
        "value": "joe@example.com"
      }],
      "subject": "World domination",
      ...
    }
  }
}], "0" ]]
```

Note the specified "Disposition-Notification-To" header indicating where to send MDN back (usually the sender of the email).

4.3. Parsing a received MDN

The client issues a parse request:

```
[[ "EmailSubmission/parseMDN", {  
  "accountId": "ue150411c",  
  "blobIds": "0f9f65ab-dc7b-4146-850f-6e4881093965"  
}, "0" ]]
```

The server responds:

```
[[ "EmailSubmission/parseMDN", {  
  "accountId": "ue150411c",  
  "parsed": {  
    "0f9f65ab-dc7b-4146-850f-6e4881093965": {  
      "forEmailId": "Md45b47b4877521042cec0938",  
      "subject": "Read receipt for: World domination",  
      "textBody": "This receipt shows that the email has been  
        displayed on your recipient's computer. There is no  
        guaranty it has been read or understood.",  
      "reportingUA": "linagora.com; OpenPaaS",  
      "disposition": {  
        "actionMode": "manual-action",  
        "sendingMode": "MDN-sent-manually",  
        "type": "displayed"  
      }  
    },  
    "finalRecipient": "rfc822; john@example.com",  
    "originalMessageID": "<1521557867.2614.0.camel@apache.org>"  
  }  
}, "0" ]]
```

5. IANA Considerations

5.1. JMAP Capability Registration for "mdn"

IANA will register the "mdn" JMAP Capability as follows:

Capability Name: "urn:ietf:params:jmap:mdn"

Specification document: this document

Intended use: common

Change Controller: IETF

Security and privacy considerations: this document, section 6.

5.2. Registration of JMAP keyword '\$MDNSent'

This registers the JMAP keyword '\$MDNSent' in the "IMAP and JMAP keywords Registry".

Keyword name: "\$MDNSent"

Scope: IMAP and JMAP

Purpose (description): Specifies that a Message Disposition Notification (MDN) must not be sent for any message annotated with the \$MDNSent IMAP keyword.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword can cause automatic action by the client. See [RFC3503] for more details.

When/by whom the keyword is set/cleared: This keyword is set by an IMAP client when it decides to act on an MDN request, or when uploading a sent or draft message. It can also be set by a delivery agent. Once set, the flag SHOULD NOT be cleared.

Related keywords: None

Related IMAP/JMAP Capabilities: None

Security Considerations: See Section 6 of [RFC3503]

Published specification (recommended): this document

Person & email address to contact for further information: (editor-contact-goes-here)

Intended usage: COMMON

Owner/Change controller: IESG

6. Security considerations

The same considerations regarding MDN (see [RFC8098]) apply to this document.

7. References

7.1. Normative References

- [I-D.ietf-jmap-core]
Jenkins, N. and C. Newman, "JSON Meta Application Protocol", draft-ietf-jmap-core-14 (work in progress), January 2019.
- [I-D.ietf-jmap-mail]
Jenkins, N. and C. Newman, "JMAP for Mail", draft-ietf-jmap-mail-15 (work in progress), February 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>>.
- [RFC3503] Melnikov, A., "Message Disposition Notification (MDN) profile for Internet Message Access Protocol (IMAP)", RFC 3503, DOI 10.17487/RFC3503, March 2003, <<https://www.rfc-editor.org/info/rfc3503>>.
- [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <<https://www.rfc-editor.org/info/rfc5322>>.
- [RFC8098] Hansen, T., Ed. and A. Melnikov, Ed., "Message Disposition Notification", STD 85, RFC 8098, DOI 10.17487/RFC8098, February 2017, <<https://www.rfc-editor.org/info/rfc8098>>.

7.2. Informative References

- [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>>.

Author's Address

Raphael Ouazana (editor)
Linagora
100 Terrasse Boieldieu - Tour Franklin
Paris - La Defense CEDEX 92042
France

Email: rouazana@linagora.com
URI: <https://www.linagora.com>

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2020

A. Melnikov
Isode Ltd
July 8, 2019

Extensions to JMAP for S/MIME signature verification
draft-ietf-jmap-smime-00

Abstract

This document specifies extension to JMAP for returning S/MIME signature verification status.

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 9, 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	2
3. Addition to the capabilities object	2
4. Extension to Email/get for S/MIME signature verification . .	2
5. Open Issues	4
6. IANA Considerations	4
6.1. JMAP capability registration for "smime"	4
7. Security Considerations	5
8. Normative References	5
Author's Address	5

1. Introduction

[I-D.ietf-jmap-mail] is a JSON based application protocol for synchronising email data between a client and a server.

This document describes an extension to JMAP for returning S/MIME [RFC8551] signature verification status, without requiring a JMAP client to download the signature and all signed body parts or to download and decode CMS.

2. Conventions Used in This Document

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].

3. Addition to the capabilities object

The capabilities object is returned as part of the standard JMAP Session object; see the JMAP spec. Servers supporting `_this_` specification MUST add a property called "urn:ietf:params:jmap:smime" to the capabilities object.

The value of this property is an empty object in both the JMAP session `_capabilities_` property and an account's `_accountCapabilities_` property.

4. Extension to Email/get for S/MIME signature verification

[I-D.ietf-jmap-mail] defines Email/get method for retrieving message specific information. This document defines the following pseudo values in the `_properties_` argument:

- o `*smimeStatus*`: If "smimeStatus" is included in the list of requested properties, it MUST be interpreted by the server as a request to return "smimeStatus" property.

The "smimeStatus" response property is defined as follows:

smimeStatus: "String|null". null signifies that the message doesn't contain any signature. Possible string values of the property are listed below. Servers MAY return other values not defined below. Client MUST treat unrecognized values as "unknown":

unknown S/MIME message, but it is neither signed, nor encrypted. This can also be returned for a multipart/signed message which contains unrecognized signing protocol (for example OpenPGP).

signed S/MIME signed message, but the signature was not yet verified. Some servers might not attempt to verify signature until a particular message is requested by the client.

signed/verified S/MIME signed message and the sender's signature was successfully verified, sender matches the From header field and the sender's certificate (and the certificate chain) is trusted for signing.

signed/failed S/MIME signed message, but the signature failed to verify. This might be a policy related decision (message signer doesn't match the From header field), message was modified, the signer's certificate has expired or was revoked, etc.

```
["Email/get", {  
  "ids": [ "f123u987" ],  
  "properties": [ "mailboxIds", "from", "subject", "date", "smimeStatus" ]  
}, "#1"]
```

This will result in the following response:

```
[[["Email/get", {  
  "accountId": "abc",  
  "state": "41234123231",  
  "list": [  
    {  
      id: "f123u457",  
      mailboxIds: { "f123": true },  
      from: [{name: "Joe Bloggs", email: "joe@bloggs.com"}],  
      subject: "Dinner on Thursday?",  
      date: "2013-10-13T14:12:00Z",  
      smimeStatus: "signed/verified"  
    }  
  ]  
}, "#1"]]
```

Example

5. Open Issues

[[This section should be empty before publication]]

6. IANA Considerations

6.1. JMAP capability registration for "smime"

IANA is requested to register the "smime" JMAP Capability as follows:

Capability Name: "urn:ietf:params:jmap:smime"

Specification document: this document

Intended use: common

Change Controller: IETF

Security and privacy considerations: this document, Section 7

7. Security Considerations

Server side S/MIME signature verification requires the client to trust server verification code and configuration to perform S/MIME signature verification. For example, if the server is not configured with some Trust Anchors, some messages will have "signed/failed" status instead of "signed/verified".

TBD.

8. 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.
- [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.
- [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>>.
- [RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <<https://www.rfc-editor.org/info/rfc8551>>.

Author's Address

Alexey Melnikov
Isode Ltd
14 Castle Mews
Hampton, Middlesex TW12 2NP
UK

E-Mail: Alexey.Melnikov@isode.com

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
URI: <http://www.fastmail.com/>