

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>