Stream: Internet Engineering Task Force (IETF)
RFC: 9007
Category: Standards Track
Published: March 2021
ISSN: 2070-1721
Author: R. Ouazana, Ed.
Linagora

RFC 9007
Handling Message Disposition Notification with the JSON Meta Application Protocol (JMAP)

Abstract

This document specifies a data model for handling Message Disposition Notifications (MDNs) (see RFC 8098) in the JSON Meta Application Protocol (JMAP) (see RFCs 8620 and 8621).

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/rfc9007.

Copyright Notice

Table of Contents

1. Introduction
   1.1. Notational Conventions
   1.2. Terminology
   1.3. Addition to the Capabilities Object
2. MDN
   2.1. MDN/send
   2.2. MDN/parse
3. Samples
   3.1. Sending an MDN for a Received Email Message
   3.2. Asking for an MDN When Sending an Email Message
   3.3. Parsing a Received MDN
4. IANA Considerations
   4.1. JMAP Capability Registration for "mdn"
   4.2. JMAP Error Codes Registration for "mdnAlreadySent"
5. Security Considerations
6. Normative References

Author's Address

1. Introduction

JMAP ("The JSON Meta Application Protocol (JMAP)" [RFC8620]) 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 it provides a consistent interface to different data types.

JMAP for Mail ("The JSON Meta Application Protocol (JMAP) for Mail" [RFC8621]) specifies a data model for synchronising email data with a server using JMAP. Clients can use this to efficiently search, access, organise, and send messages.

Message Disposition Notifications (MDNs) are defined in [RFC8098] and are used as "read receipts", "acknowledgements", or "receipt notifications".

A client can come across MDNs in different ways:

1. When receiving an email message, an MDN can be sent to the sender. This specification defines an "MDN/send" method to cover this case.
2. When sending an email message, an MDN can be requested. This must be done with the help of a header field, as already specified by [RFC8098]; the header field can already be handled by guidance in [RFC8621].
3. When receiving an MDN, the MDN could be related to an existing sent message. This is already covered by [RFC8621] in the EmailSubmission object. A client might want to display detailed information about a received MDN. This specification defines an "MDN/parse" 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 [RFC8620]. 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.

Because keywords are case insensitive in IMAP but case sensitive in JMAP, the `$mdnsent` keyword **MUST** always be used in lowercase.

1.3. Addition to the Capabilities Object

Capabilities are announced as part of the standard JMAP Session resource; see [RFC8620], Section 2. This defines a new capability, "urn:ietf:params:jmap:mdn".

The capability "urn:ietf:params:jmap:mdn" being present in the "accountCapabilities" property of an account represents support for the "MDN" data type, parsing MDNs via the "MDN/parse" method, and creating and sending MDN messages via the "MDN/send" method. Servers that include the capability in one or more "accountCapabilities" properties **MUST** also include the property in the "capabilities" property.

The value of this "urn:ietf:params:jmap:mdn" property is an empty object both in the account's "accountCapabilities" property and in the "capabilities" property.

2. MDN

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

- forEmailId: `Id|null`

  The Email id of the received message to which this MDN is related. This property **MUST NOT** be null for "MDN/send" but **MAY** be null in the response from the "MDN/parse" method.
- subject: `String|null`

  The subject used as "Subject" header field for this MDN.
- textBody: `String|null`

  The human-readable part of the MDN, as plain text.
- includeOriginalMessage: `Boolean` (default: false)

  If `true`, the content of the original message will appear in the third component of the multipart/report generated for the MDN. See [RFC8098] for details and security considerations.
- reportingUA: `String|null`

  The name of the Mail User Agent (MUA) creating this MDN. It is used to build the MDN report part of the MDN. Note that a `null` value may have better privacy properties.
- disposition: `Disposition`

  The object containing the diverse MDN disposition options.
- mdnGateway: `String|null` (server-set)

  The name of the gateway or Message Transfer Agent (MTA) that translated a foreign (non-Internet) message disposition notification into this MDN.
- originalRecipient: `String|null` (server-set)

  The original recipient address as specified by the sender of the message for which the MDN is being issued.
- finalRecipient: `String|null`

  The recipient for which the MDN is being issued. If set, it overrides the value that would be calculated by the server from the Identity defined in the "MDN/send" method, unless explicitly set by the client.
- originalMessageId: `String|null` (server-set)

  The "Message-ID" header field [RFC5322] (not the JMAP id) of the message for which the MDN is being issued.
- error: `String[]|null` (server-set)

  Additional information in the form of text messages when the "error" disposition modifier appears.
- extensionFields: `String[String]|null`

The object where keys are extension-field names, and values are extension-field values (see [RFC8098], Section 3.3).

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

- actionMode: `String`

  This **MUST** be one of the following strings: `manual-action` / `automatic-action`
- sendingMode: `String`

  This **MUST** be one of the following strings: `mdn-sent-manually` / `mdn-sent-automatically`
- type: `String`

  This **MUST** be one of the following strings: `deleted` / `dispatched` / `displayed` / `processed`

See [RFC8098] for the exact meaning of these different fields. These fields are defined as case insensitive in [RFC8098] but are case sensitive in this RFC and **MUST** be converted to lowercase by "MDN/parse".

2.1. MDN/send

The "MDN/send" method sends a message in the style of [RFC5322] from an MDN object. When calling this method, the "using" property of the Request object **MUST** contain the capabilities "urn:ietf:params:jmap:mdn" and "urn:ietf:params:jmap:mail"; the latter because of the implicit call to "Email/set" and the use of Identity objects, which is described below. The method takes the following arguments:

- accountId: `Id`

  The id of the account to use.
- identityId: `Id`

  The id of the Identity to associate with these MDNs. The server will use this identity to define the sender of the MDNs and to set the "finalRecipient" field.
- send: `Id[MDN]`

  A map of the creation id (client specified) to MDN objects.
- onSuccessUpdateEmail: `Id[PatchObject]|null`

  A map of the id to an object containing properties to update on the Email object referenced by the "MDN/send" if the sending succeeds. This will always be a backward reference to the creation id (see the example below in Section 3.1).

The response has the following arguments:

- accountId: `Id`

  The id of the account used for the call.
- sent: `Id[MDN]|null`

A map of the creation id to an MDN containing any properties that were not set by the client. This includes any properties that were omitted by the client and thus set to a default by the server. This argument is null if no MDN objects were successfully sent.

- notSent: `Id[SetError]|null`

  A map of the creation id to a SetError object for each record that failed to be sent or null if all successful.

In this context, the existing SetError types defined in [RFC8620] and [RFC8621] are interpreted as follows:

notFound: The reference "forEmailId" cannot be found or has no valid "Disposition-Notification-To" header field.

forbidden: "MDN/send" would violate an Access Control List (ACL) or other permissions policy.

forbiddenFrom: The user is not allowed to use the given "finalRecipient" property.

overQuota: "MDN/send" would exceed a server-defined limit on the number or total size of sent MDNs. It could include limitations on sent messages.

tooLarge: "MDN/send" would result in an MDN that exceeds a server-defined limit for the maximum size of an MDN or more generally, on email message.

rateLimit: Too many MDNs or email messages have been created recently, and a server-defined rate limit has been reached. It may work if tried again later.

invalidProperties: The record given is invalid in some way.

The following is a new SetError:

mdnAlreadySent: The message has the `$mdnsent` keyword already set.

If the "accountId" or "identityId" given cannot be found, the method call is rejected with an `invalidArguments` error.

The client **MUST NOT** issue an "MDN/send" request if the message has the `$mdnsent` keyword set.

When sending the MDN, the server is in charge of generating the "originalRecipient" and "originalMessageId" fields according to [RFC8098]. "finalRecipient" will also generally be generated by the server based on the provided identity, but if specified by the client and allowed (see Section 5), the server will use the client-provided value.

The client is expected to explicitly update each "Email" for which an "MDN/send" has been invoked in order to set the `$mdnsent` keyword on these messages. To ensure that, the server **MUST** reject an "MDN/send" that does not result in setting the keyword `$mdnsent`. Thus, the server **MUST** check that the "onSuccessUpdateEmail" property of the method is correctly set to update this keyword.

2.2. MDN/parse

This method allows a client to parse blobs as messages in the style of [RFC5322] 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 message the client could expect to be an MDN.

The "forEmailId" property can be null or missing if the "originalMessageId" property is missing or does not refer to an existing message or if the server cannot efficiently calculate the related message (for example, if several messages get the same "Message-ID" header field).

The "MDN/parse" method takes the following arguments:

- accountId: `Id`

  The id of the account to use.
- blobIds: `Id[]`

  The ids of the blobs to parse.

The response has the following arguments:

- accountId: `Id`

  The id of the account used for the call.
- parsed: `Id[MDN]|null`

  A map of the blob id to a parsed MDN representation for each successfully parsed blob or null if none.
- notParsable: `Id[]|null`

  A list of ids given that corresponds to blobs that could not be parsed as MDNs or null if none.
- notFound: `Id[]|null`

  A list of blob ids given that could not be found or null if none.

The following additional errors may be returned instead of the "MDN/parse" response:

requestTooLarge: The number of ids requested by the client exceeds the maximum number the server is willing to process in a single method call.

invalidArguments: If the given "accountId" cannot be found, the MDN parsing is rejected with an `invalidArguments` error.

3. Samples

3.1. Sending an MDN for a Received Email Message

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

```
[[ "MDN/send", {
  "accountId": "ue150411c",
  "identityId": "I64588216",
  "send": {
    "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
          guarantee it has been read or understood.",
      "reportingUA": "joes-pc.cs.example.com; Foomail 97.1",
      "disposition": {
        "actionMode": "manual-action",
        "sendingMode": "mdn-sent-manually",
        "type": "displayed"
      },
      "extension": {
        "EXTENSION-EXAMPLE": "example.com"
      }
    }
  },
  "onSuccessUpdateEmail": {
    "#k1546": {
      "keywords/$mdnsent": true
    }
  }
}, "0" ]]
```

If the email id matches an existing email message without the `$mdnsent` keyword, the server can answer:

```
[[ "MDN/send", {
  "accountId": "ue150411c",
  "sent": {
    "k1546": {
      "finalRecipient": "rfc822; john@example.com",
      "originalMessageId": "<199509192301.23456@example.org>"
    }
  }
}, "0" ],
[ "Email/set", {
  "accountId": "ue150411c",
  "oldState": "23",
  "newState": "42",
  "updated": {
    "Md45b47b4877521042cec0938": {}
  }
}, "0" ]]
```

If the `$mdnsent` keyword has already been set, the server can answer an error:

```
 [[ "MDN/send", {
   "accountId": "ue150411c",
   "notSent": {
     "k1546": {
       "type": "mdnAlreadySent",
       "description" : "$mdnsent keyword is already present"
     }
   }
 }, "0" ]]
```

3.2. Asking for an MDN When Sending an Email Message

This is done with the "Email/set" "create" method of [RFC8621].

```
[[ "Email/set", {
  "accountId": "ue150411c",
  "create": {
    "k2657": {
      "mailboxIds": {
        "2ea1ca41b38e": true
      },
      "keywords": {
        "$seen": true,
        "$draft": true
      },
      "from": [{
        "name": "Joe Bloggs",
        "email": "joe@example.com"
      }],
      "to": [{
        "name": "John",
        "email": "john@example.com"
      }],
      "header:Disposition-Notification-To:asText": "joe@example.com",
      "subject": "World domination",
      ...
    }
  }
}, "0" ]]
```

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

3.3. Parsing a Received MDN

The client issues a parse request:

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

The server responds:

```
[[ "MDN/parse", {
  "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
          guarantee it has been read or understood.",
      "reportingUA": "joes-pc.cs.example.com; Foomail 97.1",
      "disposition": {
        "actionMode": "manual-action",
        "sendingMode": "mdn-sent-manually",
        "type": "displayed"
      },
      "finalRecipient": "rfc822; john@example.com",
      "originalMessageId": "<199509192301.23456@example.org>"
    }
  }
}, "0" ]]
```

In the case that a blob id is not found, the server would respond:

```
[[ "MDN/parse", {
  "accountId": "ue150411c",
  "notFound": [ "0f9f65ab-dc7b-4146-850f-6e4881093965" ]
}, "0" ]]
```

If the blob id has been found but is not parsable, the server would respond:

```
[[ "MDN/parse", {
  "accountId": "ue150411c",
  "notParsable": [ "0f9f65ab-dc7b-4146-850f-6e4881093965" ]
}, "0" ]]
```

4. IANA Considerations

4.1. JMAP Capability Registration for "mdn"

This section registers the "mdn" JMAP Capability in the "JMAP Capabilities" registry as follows:

Capability Name: `urn:ietf:params:jmap:mdn`
Intended Use: common
Change Controller: IETF
Security and Privacy Considerations: This document, Section 5.
Reference: This document

4.2. JMAP Error Codes Registration for "mdnAlreadySent"

IANA has registered one new error code in the "JMAP Error Codes" registry, as defined in [RFC8620].

JMAP Error Code: mdnAlreadySent
Intended Use: common
Change Controller: IETF
Description: The message has the `$mdnsent` keyword already set. The client **MUST NOT** try again to send an MDN for this message.
Reference: This document, Section 2.1

5. Security Considerations

The same considerations regarding MDN (see [RFC8098] and [RFC3503]) apply to this document.

In order to reinforce trust regarding the relation between the user sending an email message and the identity of this user, the server **SHOULD** validate in conformance to the provided Identity that the user is permitted to use the "finalRecipient" value and return a `forbiddenFrom` error if not.

6. 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>.

[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>.

[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>.

[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>.

[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>.

Author's Address

Raphaël Ouazana (EDITOR)
Linagora
100 Terrasse Boieldieu - Tour Franklin
92042 Paris - La Défense CEDEX
France
Email: rouazana@linagora.com
URI: https://www.linagora.com