Handling Message Disposition Notification with JMAP
draft-ietf-jmap-mdn-03

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 May 23, 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.
1. Introduction

JMAP ([RFC8620] - 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 MDN/set 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 [RFC8621] this way.

3. When receiving an MDN, the MDN could be related to an existing sent mail. This is already covered by [RFC8621] in the EmailSubmission object. Client could want to display detailed information about a received MDN. This specification defines a 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.

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:

- *forEmailId*: "String" Email Id of the received email this MDN is relative to.

- *subject*: "String|null" Subject used as "Subject" header for this MDN.

- *textBody*: "String|null" Human readable part of the MDN, as plain text.

- *reportingUA*: "String|null" Name of the MUA creating this MDN. It is used to build the MDN Report part of the MDN.

- *disposition*: "Disposition" Object containing the diverse MDN disposition options.
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.

## 2.1. MDN/set

Standard "/set" method as described in [RFC8620] where only the `_create_` parameter is supported.

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

The client SHOULD NOT issue a MDN/set request if the message has the "$MDNSent" keyword set. In this case, the server MUST reject the submission with a standard "alreadyExists" SetError.
When sending the MDN, the server is in charge of generating the _originalRecipient_, _finalRecipient_ and _originalMessageID_ fields accordingly to the [RFC8098] specification.

For each "forEmailId" whose MDN where sent, the server MUST add a "$MDNSent" keyword to the email. See [RFC3503] for more details.

### 2.2. MDN/parse

This method allows a client 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:

- *accountId*: "String" 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 blob id to parsed MDN representation for each successfully parsed blob, or null if none.
- *notParsable*: "Id[]|null" A list of ids given that corresponded 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.

### 3. Samples

#### 3.1. Sending an MDN for a received email

A client can use the following request to send an MDN back to the sender:
If the email id matches an existing email without the "$MDNSent" keyword, the server can answer:

```
[[ "MDN/set", {
    "accountId": "ue150411c",
    "oldState": "012421s6-8nrq-4ps4-9330r95ins21",
    "newState": "355421f6-8aed-4cf4-a0c4-7377e951af36",
    "created": {
        "k1546": {
            "finalRecipient": "rfc822; john@example.com",
            "originalMessageID": "<1521557867.2614.0.camel@apache.org>"
        }
    }
}, "0" ],
```

3.2. Asking for MDN when sending an email

This is done with the [RFC8621] "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"
         }],
         "header:Disposition-Notification-To": "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).

### 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:
4. IANA Considerations

4.1. 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 5.

5. Security considerations

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

6. References
6.1. Normative References


6.2. Informative References


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