Internet Engineering Task Force (IETF)
Request for Comments: 9425
Category: Standards Track
ISSN: 2070-1721

R. Cordier, Ed.
Linagora Vietnam

June 2023

JSON Meta Application Protocol (JMAP) for Quotas

Abstract

This document specifies a data model for handling quotas on accounts with a server using the JSON Meta Application Protocol (JMAP).

Status of This Memo

This is an Internet Standards Track document.

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

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9425.

Table of Contents

1. Introduction

The JSON Meta Application Protocol (JMAP) [RFC8620] is a generic protocol for synchronizing data, such as mails, calendars, or contacts between a client and a server. It is optimized for mobile and web environments and aims to provide a consistent interface to different data types.

This specification defines a data model for handling quotas over JMAP, allowing a user to obtain details about a certain quota.

This specification does not address quota administration, which should be handled by other means.

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.

1.2. Terminology

This document reuses the terminology from the core JMAP specification established in Section 1.6 of [RFC8620].

The term "Quota" (when capitalized) is used to refer to the data type defined in Section 4 and instance of that data type.

2. Addition to the Capabilities Object

The capabilities object is returned as part of the JMAP Session object; see [RFC8620], Section 2.

This document defines one additional capability URI.

2.1. urn:ietf:params:jmap:quota

This represents support for the Quota data type and associated API methods. Servers supporting this specification MUST add a property called "urn:ietf:params:jmap:quota" 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.

3. Sub-types of the Quota Data Type

There are two fields within the Quota data type, which have an enumerated set of possible values. These are:

3.1. Scope

The Scope data type is used to represent the entities the quota applies to. It is defined as a "String" with values from the following set:

* account: The quota information applies to just the client's account.

* domain: The quota information applies to all accounts sharing this domain.

* global: The quota information applies to all accounts belonging to the server.

3.2. ResourceType

The ResourceType data type is used to act as a unit of measure for the quota usage. It is defined as a "String" with values from the following set:

* count: The quota is measured in a number of data type objects. For example, a quota can have a limit of 50 "Mail" objects.

* octets: The quota is measured in size (in octets). For example, a quota can have a limit of 25000 octets.

4. Quota

The Quota is an object that displays the limit set to an account usage. It then shows as well the current usage in regard to that limit.

4.1. Properties of the Quota Object

The Quota object MUST contain the following fields:

* id: Id

  The unique identifier for this object.

* resourceType: String

  The resource type of the quota as defined in Section 3.2.

* used: UnsignedInt

  The current usage of the defined quota, using the "resourceType" defined as unit of measure. Computation of this value is handled by the server.

* hardLimit: UnsignedInt

  The hard limit set by this quota, using the "resourceType" defined as unit of measure. Objects in scope may not be created or updated if this limit is reached.

* scope: String

  The "Scope" of this quota as defined in Section 3.1.

* name: String

  The name of the quota. Useful for managing quotas and using queries for searching.

* types: String[]

A list of all the type names as defined in the "JMAP Types Names" registry (e.g., Email, Calendar, etc.) to which this quota applies. This allows the quotas to be assigned to distinct or shared data types.

The server MUST filter out any types for which the client did not request the associated capability in the "using" section of the request. Further, the server MUST NOT return Quota objects for which there are no types recognized by the client.

The Quota object MAY contain the following fields:

* warnLimit: UnsignedInt|null

  The warn limit set by this quota, using the "resourceType" defined as unit of measure. It can be used to send a warning to an entity about to reach the hard limit soon, but with no action taken yet. If set, it SHOULD be lower than the "softLimit" (if present and different from null) and the "hardLimit".

* softLimit: UnsignedInt|null

  The soft limit set by this quota, using the "resourceType" defined as unit of measure. It can be used to still allow some operations but refuse some others. What is allowed or not is up to the server. For example, it could be used for blocking outgoing events of an entity (sending emails, creating calendar events, etc.) while still receiving incoming events (receiving emails, receiving calendars events, etc.). If set, it SHOULD be higher than the "warnLimit" (if present and different from null) but lower than the "hardLimit".

* description: String|null

  Arbitrary, free, human-readable description of this quota. It might be used to explain where the different limits come from and explain the entities and data types this quota applies to. The description MUST be encoded in UTF-8 [RFC3629] as described in [RFC8620], Section 1.5, and selected based on an Accept-Language header in the request (as defined in [RFC9110], Section 12.5.4) or out-of-band information about the user's language or locale.

The following JMAP methods are supported.

4.2. Quota/get

Standard "/get" method as described in [RFC8620], Section 5.1. The _id_'s argument may be "null" to fetch all quotas of the account at once, as demonstrated in Section 5.1.

4.3. Quota/changes

Standard "/changes" method as described in [RFC8620], Section 5.2, but with one extra argument in the response:

* updatedProperties: String[]|null

  If only the "used" Quota property has changed since the old state, this will be a list containing only that property. If the server is unable to tell if only "used" has changed, it MUST be null.

Since "used" frequently changes, but other properties are generally only changed rarely, the server can help the client optimize data transfer by keeping track of changes to quota usage separate from other state changes. The updatedProperties array may be used

directly via a back-reference in a subsequent Quota/get call in the same request, so only these properties are returned if nothing else has changed.

Servers MAY decide to add other properties to the list that they judge to be changing frequently.

This method’s usage is demonstrated in Section 5.2.

4.4. Quota/query

This is a standard "/query" method as described in [RFC8620], Section 5.5.

A FilterCondition object has the following properties, any of which may be included or omitted:

* name: String

  The Quota _name_ property contains the given string.

* scope: String

  The Quota _scope_ property must match the given value exactly.

* resourceType: String

  The Quota _resourceType_ property must match the given value exactly.

* type: String

  The Quota _types_ property contains the given value.

A Quota object matches the FilterCondition if, and only if, all the given conditions match. If zero properties are specified, it is automatically true for all objects.

The following Quota properties MUST be supported for sorting:

* name

* used

4.5. Quota/queryChanges

This is a standard "/queryChanges" method as described in [RFC8620], Section 5.6.

5. Examples

5.1. Fetching Quotas

Request fetching all quotas related to an account:

```
[[ "Quota/get", {
  "accountId": "u33084183",
  "ids": null
}, "0" ]]
```

With response:

```
[[ "Quota/get", {
  "accountId": "u33084183",
  "state": "78540",
  "list": [{
```

```
      "id": "2a06df0d-9865-4e74-a92f-74dcc814270e",
      "resourceType": "count",
      "used": 1056,
      "warnLimit": 1600,
      "softLimit": 1800,
      "hardLimit": 2000,
      "scope": "account",
      "name": "bob@example.com",
      "description": "Personal account usage. When the soft limit is
                      reached, the user is not allowed to send mails or
                      create contacts and calendar events anymore.",
      "types" : [ "Mail", "Calendar", "Contact" ]
    }, {
      "id": "3b06df0e-3761-4s74-a92f-74dcc963501x",
      "resourceType": "octets",
      ...
    }, ...],
    "notFound": []
  }, "0" ]]
```

5.2. Requesting Latest Quota Changes

Request fetching the changes for a specific quota:

```
[[ "Quota/changes", {
  "accountId": "u33084183",
  "sinceState": "78540",
  "maxChanges": 20
}, "0" ],
[ "Quota/get", {
  "accountId": "u33084183",
  "#ids": {
    "resultOf": "0",
    "name": "Quota/changes",
    "path": "/updated"
  },
  "#properties": {
    "resultOf": "0",
    "name": "Quota/changes",
    "path": "/updatedProperties"
  }
}, "1" ]]
```

With response:

```
[[ "Quota/changes", {
  "accountId": "u33084183",
  "oldState": "78540",
  "newState": "78542",
  "hasMoreChanges": false,
  "updatedProperties": ["used"],
  "created": [],
  "updated": ["2a06df0d-9865-4e74-a92f-74dcc814270e"],
  "destroyed": []
}, "0" ],
[ "Quota/get", {
  "accountId": "u33084183",
  "state": "10826",
  "list": [{
    "id": "2a06df0d-9865-4e74-a92f-74dcc814270e",
    "used": 1246
  }],
  "notFound": []
}, "1" ]]
```

6. Push

Servers MUST support the JMAP push mechanisms, as specified in [RFC8620], Section 7, to allow clients to receive notifications when the state changes for the Quota type defined in this specification.

7. IANA Considerations

7.1. JMAP Capability Registration for "quota"

IANA has registered the "quota" JMAP Capability as follows:

Capability Name: urn:ietf:params:jmap:quota

Reference: RFC 9425

Intended Use: common

Change Controller: IETF

Security and Privacy Considerations: RFC 9425, Section 8

7.2. JMAP Data Type Registration for "Quota"

IANA has registered the "Quota" Data Type as follows:

Type Name: Quota

Can Reference Blobs: No

Can Use for State Change: Yes

Capability: urn:ietf:params:jmap:quota

Reference: RFC 9425

8. Security Considerations

All security considerations of JMAP [RFC8620] apply to this specification.

Implementors should be careful to make sure the implementation of the extension specified in this document does not violate the site's security policy. The resource usage of other users is likely to be considered confidential information and should not be divulged to unauthorized persons.

As for any resource shared across users (for example, a quota with the "domain" or "global" scope), a user that can consume the resource can affect the resources available to the other users. For example, a user could spam themselves with events and make the shared resource hit the limit and unusable for others (implementors could mitigate that with some rate-limiting implementation on the server).

Also, revealing domain and global quota counts to all users may cause privacy leakage of other sensitive data, or at least the existence of other sensitive data. For example, some users are part of a private list belonging to the server, so they shouldn't know how many users are in there. However, by comparing the quota count before and after sending a message to the list, it could reveal the number of people of the list, as the domain or global quota count would go up by the number of people subscribed. In order to limit those attacks, quotas with "domain" or "global" scope SHOULD only be visible to server administrators and not to general users.

9. 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>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[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>.

[RFC9007] Ouazana, R., Ed., "Handling Message Disposition Notification with the JSON Meta Application Protocol (JMAP)", RFC 9007, DOI 10.17487/RFC9007, March 2021, <https://www.rfc-editor.org/info/rfc9007>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

Acknowledgements

Thank you to Michael Bailly, who co-wrote the first draft version of this document, before deciding to turn to other matters.

Thank you to Benoit Tellier for his constant help and support on writing this document.

Thank you to Raphael Ouazana for sharing his own experience on how to write an RFC after finalizing his own document: [RFC9007].

Thank you to Bron Gondwana, Neil Jenkins, Alexey Melnikov, Joris Baum, and the people from the IETF JMAP working group in general, who helped with extensive discussions, reviews, and feedback.

Thank you to the people in the IETF organization, who took the time to read, understand, comment, and give great feedback in the last rounds.

Author’s Address

Ren Cordier (editor)
Linagora Vietnam
5 Dien Bien Phu
Hanoi
10000
Vietnam
Email: rcordier@linagora.com
URI: https://linagora.vn