```
Workgroup: JMAP
Internet-Draft: draft-ietf-jmap-quotas-07
Published: 11 November 2022
Intended Status: Standards Track
Expires: 15 May 2023
Authors: R.C. Cordier, Ed.
         Linagora Vietnam
```

<p align="center">JMAP for Quotas</p>

Abstract

This document specifies a data model for handling quotas on accounts
with a server using JMAP.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

JMAP ([RFC8620] – (U+2013) JSON Meta Application Protocol) is a
generic protocol for synchronising data, such as mails, 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 handling quotas over
JMAP, allowing you to read and explain quota information.

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.

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.

The term Quota (with that specific capitalization) is used to refer to the data type defined in this document and instance of that data type.

1.3. 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.

1.3.1. 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' (U+2019)s accountCapabilities property.

1.4. Data types

In addition to the standard JSON data types, a couple of additional data types are common to the definition of Quota objects and properties.

1.4.1. Scope

The **Scope** is a String from an enumeration defined list of values, handled by the server.

It explains the entities this value applies to. Values for the **Scope** are:

   *account: Applies for this account
   *domain: All accounts of this domain share this part of the quota
   *global: All accounts of this server share this part of the quota

1.4.2. ResourceType

The **ResourceType** is a String from an enumeration defined list of
values, handled by the server.

A resource type is like an unit of measure for the quota usage.
Values for the **ResourceType** are:

  *count: The quota is measured in 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.

1.5. Push

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

2. Quota

The quota is an object that displays the limit set to an account
usage as well as the current usage in regard to that limit.

The quota object MUST contain the following fields:

  ***id**: "Id" The unique identifier for this object. It should respect
   the JMAP ID data type defined in section 1.2 of [RFC8620]

  ***resourceType**: "ResourceType" The resource type of the quota.

  ***used**: "UnsignedInt" The current usage of the defined quota.
   Computation of this value is handled by the server.

  ***limit**: "UnsignedInt" The hard limit set by this quota. Objects in
   scope may not be created or updated if we reach this limit. It
   should be higher than the warnLimit and the softLimit.

  ***scope**: "Scope" The Scope of this quota.

  ***name**: "String" The name of the quota object. Useful for managing
   quotas and using queries for searching.

  ***dataTypes**: "String[]" A list of all the data types values that
   are applying to this quota. This allows to assign quotas to
   separated or shared data types. This MAY include data types the
   client does not recognise. Clients MUST ignore any unknown data
   type in the list.

The quota object MAY contain the following fields:

* **warnLimit**: "UnsignedInt|null" The warn limit set by this quota
  object. 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 and the limit.

* **softLimit**: "UnsignedInt|null" The soft limit set by this quota
  object. 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, ...) while
  still receiving incoming events (receiving emails, receiving
  calendars events, ...). If set, it should be higher than the
  warnLimit but lower than the limit.

* **description**: "String|null" Arbitrary free, human readable,
  description of this quota. Might be used to explain where the
  limit comes from and explain the entities and data types this
  quota applies to.

2.1. Quota/get

Standard " (U+201C)/get" (U+201D) method as described in [RFC8620]
section 5.1. The *ids* argument may be null to fetch all at once.

2.2. Quota/changes

Standard " (U+201C)/changes" (U+201D) method as described in
[RFC8620] section 5.2 but with one extra argument to the response:

* **updatedProperties**: "String[]|null" If only the
  " (U+201C)used" (U+201D) Quota properties has changed since the
  old state, this will be the list of properties that may have
  changed. If the server is unable to tell if only "used" has
  changed, it MUST just be null.

Since "used" frequently changes but other properties are generally
only changed rarely, the server can help the client optimise 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 changing frequently.

2.3. Quota/query

This is a standard " (U+201C)/query" (U+201D) method as described in [RFC8620], Section 5.5.

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

  * **name**: "String" The Quota *name* property contains the given string.
  * **scopes**: "Scope[]" The Quota *scope* property must be in this list to match the condition.
  * **resourceTypes**: "ResourceType[]" The Quota *resourceType* property must be in this list to match the condition.
  * **dataTypes**: "String[]" The Quota *dataTypes* property must contain the elements in this list to match the condition.

A Quota object matches the FilterCondition if and only if all of 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**

2.4. Quota/queryChanges

This is a standard " (U+201C)/queryChanges" (U+201D) method as described in [RFC8620], Section 5.6.

2.5. Examples

2.5.1. 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,
    "limit": 2000,
    "scope": "account",
    "name": "bob@example.com",
    "description": "Personal account usage",
    "dataTypes" : [ "Mail", "Calendar", "Contact" ]
  }, {
    "id": "3b06df0e-3761-4s74-a92f-74dcc963501x",
    "resourceType": "octets",
    ...
  }, ...],
  "notFound": []
}, "0" ]]
```

2.5.2. Requesting latest quota changes

   Request fetching the changes for a specific quota:

```
[[ "Quota/changes", {
  "accountId": "u33084183",
  "sinceState": "10824",
  "maxChanges": 20,
  "updatedProperties": ["used"]
}, "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": "10824",
  "newState": "10826",
  "hasMoreChanges": false,
  "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" ]]
```

3. Security considerations

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

Implementors should be careful to make sure the implementation of
that extension 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 himself 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).

4. IANA Considerations

4.1. JMAP Capability Registration for "quota"

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

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

Specification document: this document

Intended use: common

Change Controller: IETF

Security and privacy considerations: this document, section 4.

5. 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>.

[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>.

Author's Address

René Cordier (editor)
Linagora Vietnam
5 Dien Bien Phu
Hanoi
10000
Vietnam

Email: rcordier@linagora.com
URI: https://linagora.vn