Workgroup: JMAP
Internet-Draft: draft-ietf-jmap-quotas-02
Published: 23 August 2021
Intended Status: Standards Track
Expires: 24 February 2022
Authors: R.C. Cordier, Ed.
Linagora Vietnam

JMAP for Quotas

Abstract

This document specifies a data model for handling quotas on accounts 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 24 February 2022.

Table of Contents

1. Introduction

JMAP ([RFC8620] - 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. The value of this property in the JMAP session capabilities property is an empty object.

The value of this property in an account's accountCapabilities property is an object that MUST contain the following information on server capabilities and permissions for that account:

* **quotaIds:** Id[] (default: []) A list of quota ids bound to that account, or [] if that account has no quota restrictions.

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. Some custom specifications might provide some additional values. If the client does not specify custom scope specifications in the "using" parameter of the request, the server should respond the JSON value

null, instead of answering a scope value that the client does not support. Standard values 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. Some custom specifications might provide some additional values. If the client does not specify custom resource type specifications in the "using" parameter of the request, the server should respond the JSON value null, instead of answering a resource type value that the client does not support. Standard values are:

- *count: The quota is measured in number of data type objects. For example, a quota can have a limit of 50 Mail objects.
- *size: The quota is measured in size (in bytes). For example, a quota can have a limit of 25000 bytes.

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 datatype 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 object. No more outgoing and ingoing objects should be allowed 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 use 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 field:

- ***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 refusing some others. What is allowed or not is up to the server. 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 "/get" method as described in [RFC8620] section 5.1. The ids argument may be null to fetch all at once.

2.2. Quota/changes

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

- ***updatedProperties**: String[]|null If only the "used" 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 "/query" 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 "/queryChanges" 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": "size",
            ...
          }, ...],
          "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
}, "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.

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