

# Development of WebDAV quotas

## Development of WebDAV quotas

\*\*

Auteur : Thomas Bellembois (University of Rennes 1) (<http://http://>)

- [Development of WebDAV quotas](#)
- [Introduction](#)
- [Notational conventions](#)
- [Solution overview](#)
  - [Metadata](#)
  - [Compliance with the WebDAV quota RFC](#)
  - [Coherence](#)
  - [Quota processing](#)
    - [Firstly without the notion of virtual root \(autonomous\)](#)
    - [Now the virtual root...](#)
      - [Why?](#)
      - [How?](#)
- [Configuration - administration](#)
  - [Requirements](#)
  - [Solution 1](#)
    - [Configuration file](#)
    - [<path>](#)
    - [<init-used>](#)
    - [Thread](#)
  - [Solution 2](#)
    - [Web service](#)
    - [Functions](#)
- [Notes](#)

## Introduction

Within the context of a CMS development, we had to develop a Slide-based WebDav server to manage quotas. The purpose of this document is to explain the quota implementation we have chosen. We tried to stick close to the draft ietf-webdav-quota ( <http://tools.ietf.org/id/draft-ietf-webdav-quota-07.txt>) but took the liberty of extending its specifications to meet our requirements. We tried to think about a secure and flexible quota system. We essentially focused on the flexibility and ease of use of quotas. We also wanted to separate the way quotas are configured/updated (how we put/update quota metadata), and the way they are managed by the server. The first part (Solution overview) explains the quota implementation we have chosen and the second part (Configuration - administration) explains the process we thought about to configure/administrate quotas on our WebDAV server. The two parts are independent.

## Notational conventions

The terms " virtual root " and " autonomous collection " are equivalent. The terms " metadata " and " properties " are equivalent. The term " quota " refers to the DAV:quota-available-bytes metadata . The term " used space " refers to the DAV:quota-used-bytes metadata.

## Solution overview

### Metadata

We define three **live protected** properties DAV:quota-available-bytes DAV:quota-used-bytes and ESUP:virtual-root . These properties (called quota metadata in this document) can NOT be modified by a proppatch method (protected). These properties can be retrieved by a propfind method. If one of these properties is not protected, they are ignored by the server. The ESUP:virtual-root metadata is optional and is false if not present. Some collections contain these properties, but not necessary all. A collection either contains the two (DAV😊 or three (DAV: ESUP😊 properties or none of them at all. The DAV:quota-available-bytes and DAV:quota-used-bytes have the same meaning as in the draft ietf-webdav-quota - but we added the "virtual root" notion :

- DAV:quota-available-bytes : value in octets representing the amount of disk space that can be allocated to this resource for further allocations.
- DAV:quota-used-bytes : value in octets representing the amount of disk space used by this resource - including its files and sub-collections **but \*\*NOT its autonomous sub-collections.**
- ESUP:virtual-root defines an autonomous collection - true or false.

### Compliance with the WebDAV quota RFC

We are 98% compliant with the WebDAV quota RFC. We differ on the following points :

- The virtual root notion
- We have implemented a "soft" quota management, it means that quota could be exceeded one time and only "one time" putting a resource on the server. We would like to implement a classic ("hard") quota management in a future version. We have decided to choose a "soft" quota management for the following reason : We have trust in the content-length HTTP header to check if a storage action can be done, except for a copy or a move action (the resource already exists on the server) we get the resource size directly "on the disk". According to the HTTP RFC the content-length header is optional. When the content-length header is not filled in (or equals to -1) we had to calculate the size of the resource handling the HTTP input stream. We could not find an easy way to do that.

## Coherence

Rule 1 > The quota-available-bytes property of a collection CAN NOT be smaller than the sum of the quota-available-bytes properties of its sub-collections.  
Rule 2 > The quota-available-bytes property of a collection SHOULD NOT be smaller than its quota-used-bytes property - sometimes possible because of the "soft" quota management. Anyway, the following quota processing ensures that the server will not run anarchically if these rules are not respected.

## Quota processing

### Firstly without the notion of virtual root (autonomous

collection)...

If a collection does not contain quota metadata, then it inherits them from its first parent containing quota metadata. If no quota metadata can be found (when the root collection is reached), this collection has no quota (unlimited quota). In concrete terms, when a new file "f" is about to be stored in a collection "C":

- We find every parent collection of "C" containing quota metadata - up to the root collection.
- If no collections containing the quota metadata are found, there is no quota for the collection "C" (or unlimited quota).
- If collections with protected quota MD are found, we calculate for each of them if their quota-available-bytes metadata will not be exceeded while adding the new file (or we do nothing if no HTTP header content length - cf. "soft" quota management).
- If at least one of the quota-available-bytes metadata is about to be exceeded, we throw a 507 error.
- If no quota-available-bytes metadata is about to be exceeded, we put the new file and update the quota-used-bytes properties for every collection found in 1).



The calculation at the step 3 is more sophisticated for move or a copy request.

### Now the virtual root...

#### Why?

The rule 1 of the "coherence" section can be a bit rigid to handle quotas. Imagine that you have the following tree : `./.../files/universityOfNorthPole/computerScienceDpt/teacherA ./.../files/universityOfNorthPole/computerScienceDpt/teacherB ./.../files/universityOfNorthPole/computerScienceDpt/teacherC ./.../files/universityOfNorthPole/computerScienceDpt/... ./.../files/universityOfNorthPole/computerScienceDpt/TeacherBigBoss` The computerScienceDpt collection has a quota of 500MB (shared by all the teachers - or we could also have a quota for each teacher (in this case the rule 2 would have to be respected)). Imagine that you want now to give to the TeacherBigBoss collection a quota of 700MB (the boss works a lot 😊):

- We need to increase the quota of the computerScienceDpt collection and all of its parents to be compliant with the rule 1
- Actually the TeacherBigBoss collection will not have 700MB at its disposal because its quota will be limited by the quota and space used by computerScienceDpt.

#### How?

So it could be useful to define some collections as "autonomous" - as if the collection was the root collection (virtual root) - in this case the quota inheritance and coherence are broken:

- An autonomous collection can have a quota bigger than its parents.
- The used space of an autonomous collection is not included in the used space of its parents.  
In concrete terms, when a new file "f" is about to be stored in a collection "C": - =( Final algorithm )=-
- We find every parent collection of "C" containing quota metadata - up to the first virtual root or to the root collection.
- If no collections containing the quota metadata are found, there is no quota for the collection "C" (or unlimited quota).
- If collections with protected quota MD are found, we calculate for each of them if their quota-available-bytes metadata will not be exceeded while adding the new file (or we do nothing if no HTTP header content length - cf. "soft" quota management).
- If at least one of the quota-available-bytes metadata is about to be exceeded, we throw a 507 error.
- If no quota-available-bytes metadata is about to be exceeded, we put the new file and update the quota-used-bytes properties for every collection found in 1).



The calculation at the step 3 is more sophisticated for move or a copy request.

## Configuration - administration

### Requirements

To administrate quotas on our server we had the following requirements: \* Proppatch methods are not allowed to write quota metadata, so we had to think about a server-side system.

- We wanted to be able to specify using only one rule that collections matching the same criteria have the same quota.
- The quota-used-bytes metadata could be initialized automatically.
- The administrator who puts/modifies the quota-available-bytes metadata HAS TO check that they are coherent.
- If the virtual-root metadata of a collection is modified, the quota-used-bytes metadata of the parent collections must be automatically updated.

### Solution 1

#### Configuration file

Quota are configured/modified by the following XML file:

```
<quotas>
  <quota>
    <path="/files/university">
      <size="150MB">
        <init-used="true">
          <virtual-root="true">
        </virtual-root>
      </size>
    </path>
  </quota>
  <quota>
    ...
  </quota>
  ...
</quotas>
```

#### <path>

The path can be a common path ( /files/university ) but also a mask like /files//**computer** . In this case the quota is applied to every collection with a depth of 3 and containing the word "computer". Using this kind of mask is very useful to define in one rule a quota for several collections.

#### <init-used>

Specifies whether the quota-used-bytes metadata has to be calculated automatically or not when it does not exist. If false, the quota-used-bytes metadata is created and set to 0, if true, the quota-used-bytes metadata is created and calculated automatically.

#### Thread

A thread starting with the server reads the file every X minutes. If the file has not been modified it does nothing. If the file was modified, the thread detects each new or modified entry. For each resource to create or modify:

- We create or modify the quota-available-bytes metadata - we do not check the coherence.
- If the quota-used-bytes metadata does not exist, we create it and calculate it if init-used=true.
- If the virtual-root metadata of the collection is **modified** we have to update the quota-used-bytes metadata of its parents up to the first virtual root or root collection.

## Solution 2

### Web service

A Web service exports quota management functions to modify quota metadata.

### Functions

The type of the function parameters can be modified. Get the quota-used-bytes MD of the given resource. `getQuotaUsedBytes(String path);` // not required given that propfind requests are allowed on this MD Get the quota-available-bytes MD of the given resource. `getQuotaAvailableBytes(String path);` // not required given that propfind requests are allowed on this MD Get the virtual-root MD of the given resource. `getVirtualRoot(String path);` // not required given that propfind requests are allowed on this MD Patch the quota-used-bytes MD of the given resource. Private function ? Failed if the MD does not exist. `patchQuotaUsedBytes(String path, int value);` Calculate the quota-used-bytes MD of the given resource - ! resource consuming process. `calculateQuotaUsedBytes(String path);` Patch the quota-available-bytes MD of the given resource. Failed if the MD does not exist. `patchQuotaAvailableBytes(String path, int value);` Set (and overwrite) the 2 DAV quota MD of the given resource. If `calculateQuotaUsedBytes=true` calculate the quota-used-bytes MD - ! resource consuming process. Else `quota-used-bytes=0`. `setQuota(String path, int availableBytes, boolean calculateQuotaUsedBytes);` Set (and overwrite) the virtual-root MD of the given resource. (Re)-calculate the quota-used-bytes MD of the parents resources up to the root node or first virtual root : > Modified from "true" to "false" :  $\text{quota-used-bytes(PR)} = \text{quota-used-bytes(PR)} + \text{quota-used-bytes(CR)}$  > Modified from "false" to "true" :  $\text{quota-used-bytes(PR)} = \text{quota-used-bytes(PR)} - \text{quota-used-bytes(CR)}$  PR = parent resources CR = current resource `setVirtualRoot(String path, boolean value);`

### Notes

- We have separated the way quotas are managed by the server and put by the administrator
- The notions of virtual root and quota metadata inheritance give a flexibility to administrate quotas.
- The thread updating quota metadata could affect performance - it should be run only when the server is not seeked.
- Quotas in the configuration file can be defined in KB, MB or GB but metadata are in octets.
- Due to the fact that some resources have inherited quota metadata, it would be good to return the information "quota inherited" in response to a propfind request.
- Do we have to expose the "virtualRoot" metadata to clients ?
- We do not count storage used for metadata.
- Quota restrictions are on collections and not users.