                        IMAP4 QUOTA extension

Status of this Memo

1.   Abstract

   The QUOTA extension of the Internet Message Access Protocol [IMAP4]
   permits administrative limits on resource usage (quotas) to be
   manipulated through the IMAP protocol.

Table of Contents

2.   Conventions Used in this Document

   In examples, "C:" and "S:" indicate lines sent by the client and
   server respectively.

3.   Introduction and Overview

   The QUOTA extension is present in any IMAP4 implementation which
   returns "QUOTA" as one of the supported capabilities to the
   CAPABILITY command.

   An IMAP4 server which supports the QUOTA capability may support
   limits on any number of resources. Each resource has an atom name
   and an implementation-defined interpretation which evaluates to an
   integer. Examples of such resources are:

      Name   Interpretation

      STORAGE   Sum of messages' RFC822.SIZE, in units of 1024 octets
      MESSAGE   Number of messages

   Each mailbox has zero or more implementation-defined named "quota
   roots". Each quota root has zero or more resource limits. All
   mailboxes that share the same named quota root share the resource
   limits of the quota root.

   Quota root names do not necessarily have to match the names of
   existing mailboxes.

4.   Commands

4.1. SETQUOTA Command

   Arguments: quota root
               list of resource limits

   Data: untagged responses: QUOTA

   Result: OK - setquota completed
               NO - setquota error: can't set that data
               BAD - command unknown or arguments invalid

   The SETQUOTA command takes the name of a mailbox quota root and a
   list of resource limits. The resource limits for the named quota root
   are changed to be the specified limits. Any previous resource limits
   for the named quota root are discarded.

   If the named quota root did not previously exist, an implementation
   may optionally create it and change the quota roots for any number of
   existing mailboxes in an implementation-defined manner.

```
   Example:    C: A001 SETQUOTA "" (STORAGE 512)
               S: * QUOTA "" (STORAGE 10 512)
               S: A001 OK Setquota completed
```

4.2. GETQUOTA Command

   Arguments: quota root

   Data: untagged responses: QUOTA

   Result: OK - getquota completed
               NO - getquota error: no such quota root, permission
               denied
               BAD - command unknown or arguments invalid

   The GETQUOTA command takes the name of a quota root and returns the
   quota root's resource usage and limits in an untagged QUOTA response.

```
   Example:    C: A003 GETQUOTA ""
               S: * QUOTA "" (STORAGE 10 512)
               S: A003 OK Getquota completed
```

4.3. GETQUOTAROOT Command

   Arguments: mailbox name

   Data: untagged responses: QUOTAROOT, QUOTA

   Result: OK - getquota completed
               NO - getquota error: no such mailbox, permission denied
               BAD - command unknown or arguments invalid

   The GETQUOTAROOT command takes the name of a mailbox and returns the
   list of quota roots for the mailbox in an untagged QUOTAROOT
   response. For each listed quota root, it also returns the quota
   root's resource usage and limits in an untagged QUOTA response.

```
   Example:    C: A003 GETQUOTAROOT INBOX
               S: * QUOTAROOT INBOX ""
               S: * QUOTA "" (STORAGE 10 512)
               S: A003 OK Getquota completed
```

5.   Responses

5.1. QUOTA Response

   Data: quota root name
                list of resource names, usages, and limits

      This response occurs as a result of a GETQUOTA or GETQUOTAROOT
      command. The first string is the name of the quota root for which
      this quota applies.

      The name is followed by a S-expression format list of the resource
      usage and limits of the quota root. The list contains zero or
      more triplets. Each triplet conatins a resource name, the current
      usage of the resource, and the resource limit.

      Resources not named in the list are not limited in the quota root.
      Thus, an empty list means there are no administrative resource
      limits in the quota root.

      Example: S: * QUOTA "" (STORAGE 10 512)

5.2. QUOTAROOT Response

   Data: mailbox name
                zero or more quota root names

      This response occurs as a result of a GETQUOTAROOT command. The
      first string is the mailbox and the remaining strings are the
      names of the quota roots for the mailbox.

      Example: S: * QUOTAROOT INBOX ""
                  S: * QUOTAROOT comp.mail.mime

6.   Formal syntax

   The following syntax specification uses the augmented Backus-Naur
   Form (BNF) notation as specified in RFC 822 with one exception; the
   delimiter used with the "#" construct is a single space (SP) and not
   one or more commas.

   Except as noted otherwise, all alphabetic characters are case-
   insensitive. The use of upper or lower case characters to define
   token strings is for editorial clarity only. Implementations MUST
   accept these strings in a case-insensitive fashion.

```
   getquota        ::= "GETQUOTA" SP astring

   getquotaroot    ::= "GETQUOTAROOT" SP astring

   quota_list      ::= "(" #quota_resource ")"

   quota_resource  ::= atom SP number SP number

   quota_response  ::= "QUOTA" SP astring SP quota_list

   quotaroot_response
                   ::= "QUOTAROOT" SP astring *(SP astring)

   setquota        ::= "SETQUOTA" SP astring SP setquota_list

   setquota_list   ::= "(" 0#setquota_resource ")"

   setquota_resource ::= atom SP number
```

7.   References

   [IMAP4] Crispin, M., "Internet Message Access Protocol - Version 4",
   RFC 1730, University of Washington, December 1994.

   [RFC-822] Crocker, D., "Standard for   the Format of ARPA Internet
   Text Messages", STD 11, RFC 822.

8.   Security Considerations

   Implementors should be careful to make sure the implementation of
   these commands 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.

9.   Author's Address

   John G. Myers
   Carnegie-Mellon University
   5000 Forbes Ave.
   Pittsburgh PA, 15213-3890

   EMail: jgm+@cmu.edu