

IMAPEXT Working Group  
Internet Draft  
Document: [draft-ietf-imapext-2086upd-03.txt](#)  
Updates: <<3501?>>  
Obsoletes: [2086](#)  
Expires: August 2005

A. Melnikov  
Editor  
February 2005

## **IMAP4 ACL extension**

### Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, or will be disclosed, and any of which I become aware will be disclosed, in accordance with [RFC 3668](#).

Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts. Internet Drafts are draft documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as ``work in progress''.

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/ietf/1id-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>.

Directories on ds.internic.net, nic.nordu.net, ftp.isi.edu, or munnari.oz.au.

A revised version of this draft document will be submitted to the RFC editor as a Proposed Standard for the Internet Community. Discussion and suggestions for improvement are requested. Distribution of this draft is unlimited.

### Abstract

The ACL (Access Control List) extension ([RFC 2086](#)) of the Internet Message Access Protocol (IMAP4rev1) permits mailbox access control lists to be retrieved and manipulated through the IMAP protocol.

This document is a revision of the [RFC 2086](#). It defines several new access control rights and clarifies which rights are required for different IMAP commands.

## **1. Conventions Used in this Document**

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

In all examples "/" character is used as hierarchy separator.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[KEYWORDS](#)].

## **2. Introduction and Overview**

The ACL (Access Control List) extension of the Internet Message Access Protocol [[IMAP4](#)] permits mailbox access control lists to be retrieved and manipulated through the IMAP protocol.

This document is a revision of the [RFC 2086](#). It tries to clarify different ambiguities in the [RFC 2086](#), in particular use of UTF-8 [[UTF-8](#)] in identifiers, which rights are required for different IMAP4 commands; how READ-WRITE/READ-ONLY response codes are related to ACL.

## **3. Access Control**

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

A server implementation conformant to this document MUST also return rights (see below) not defined in [section 3.2](#) in the "RIGHTS=" capability.

An access control list is a set of <identifier, rights> pairs.  
An ACL applies to a mailbox name.

Identifier is a UTF-8 [[UTF-8](#)] string. The identifier "anyone" is reserved to refer to the universal identity (all authentications, including anonymous). All user name strings accepted by the LOGIN or AUTHENTICATE commands to authenticate to the IMAP server are reserved as identifiers for the corresponding users. Identifiers starting with a dash ("-") are reserved for "negative rights", described below. All other identifier strings are interpreted in an implementation-defined manner.

Rights is a string listing a (possibly empty) set of alphanumeric characters, each character listing a set of operations which is being controlled. Lowercase letters are reserved for "standard" rights, listed in [section 3.1](#). (Note that for compatibility with deployed clients and servers uppercase rights are not allowed). The set of standard rights may only be extended by a standards-track document.

Digits are reserved for implementation or site defined rights.

An implementation may tie rights together or may force rights to always or never be granted to particular identifiers. For example, in an implementation that uses UNIX mode bits, the rights "swite" are tied, the "a" right is always granted to the owner of a mailbox and is never granted to another user. If rights are tied in an implementation, the implementation must be conservative in granting rights in response to SETACL commands--unless all rights in a tied set are specified, none of that set should be included in the ACL entry for that identifier. A client may discover the set of rights which may be granted to a given identifier in the ACL for a given mailbox name by using the LISTRIGHTS command.

It is possible for multiple identifiers in an access control list to apply to a given user (or other authentication identity). For example, an ACL may include rights to be granted to the identifier matching the user, one or more implementation-defined identifiers matching groups which include the user, and/or the identifier "anyone". How these rights are combined to determine the user's access is implementation-defined. An implementation may choose, for example, to use the union of the rights granted to the applicable identifiers. An implementation may instead choose, for example, to only use those rights granted to the most specific identifier present in the ACL. A client may determine the set of rights granted to the logged-in user for a given mailbox name by using the MYRIGHTS command.

When an identifier in an ACL starts with a dash ("-"), that indicates that associated rights are to be removed from the identifier that is prefixed by the dash. This is referred to as a "negative right". This differs from DELETEACL in that a negative right is added to the ACL, and is a part of the calculation of the rights.

Let's assume that an identifier "fred" refers to a user with login "fred". If the identifier "-fred" is granted the "w" right, that indicates that the "w" right is to be removed from users matching the identifier "fred", even though the user "fred" might have the "w" right as a consequence of some other identifier in the ACL. A DELETEACL of "fred" simply deletes the identifier "fred" from the ACL; it does not affect any rights that the user "fred" may get from another entry in the ACL, in particular it doesn't affect rights granted to the identifier "-fred".

Server implementations are not required to support "negative right" identifiers.

### **3.1. Standard rights**

The currently defined standard rights are (note that the list below doesn't list all commands that use a particular right):

- l - lookup (mailbox is visible to LIST/LSUB commands, SUBSCRIBE mailbox)
- r - read (SELECT the mailbox, perform STATUS)
- s - keep seen/unseen information across sessions (set or clear \SEEN flag via STORE, APPEND or COPY)
- w - write (set or clear flags other than \SEEN and \DELETED via STORE, APPEND or COPY)
- i - insert (perform APPEND, COPY into mailbox)
- p - post (send mail to submission address for mailbox, not enforced by IMAP4 itself)
- k - create mailboxes (CREATE new sub-mailboxes in any implementation-defined hierarchy, parent mailbox for the new mailbox name in RENAME)
- x - delete mailbox (DELETE mailbox, old mailbox name in RENAME)
- t - delete messages (set or clear \DELETED flag via STORE, set \DELETED flag during APPEND/COPY)
- e - perform EXPUNGE and expunge as a part of CLOSE.
- a - administer (perform SETACL/DELETEACL/GETACL)

### **3.1.1. Obsolete rights**

Due to ambiguity in [RFC 2086](#) some existing [RFC 2086](#) server implementations use the "c" right to control the DELETE command. Others chose to use the "d" right to control the DELETE command.

For the former group, let's define the "create" right as union of the "k" and "x" rights, and the "delete" right as union of the "e" and "t" rights. For the latter group, let's define the "create" rights as a synonym to the "k" right, and the "delete" right as union of the "e", "t" and "x" rights.

For compatibility with [RFC 2086](#) this section defines two virtual rights "d" and "c".

If a client includes the "d" right in a rights list, then it MUST be treated as if the client had included every member of the "delete" right. (It is not an error for a client to specify both the "d" right and one or more members of the "delete" right, but the effect is no different than if just the "d" right or all members of the "delete" right had been specified).

When any of the "delete" member rights is set in a list of rights, the server MUST also include the "d" right when returning the list in a MYRIGHTS or ACL response. This is so to enable older clients conforming to [RFC 2086](#) to work with newer servers. (\*)

Example:     C: A001 SETACL INBOX.Drafts David lrswida  
              S: A001 OK Setacl complete

The client has specified the "d" right in the SETACL command above and it expands to "et" on the server:

C: A002 GETACL INBOX.Drafts

```
S: * ACL INBOX Fred rwipslxetda David lrswideta
S: A002 OK Getacl complete
```

If the authentication identifier specified in the LISTRIGHTS command can be granted any of the "delete" member rights to access a mailbox, then the server

MUST include the "d" right in the corresponding LISTRIGHTS response. (\*)

If the member rights aren't tied to non-member rights, then the "d" right is returned by itself in the LISTRIGHTS response. If any of the member rights

needs to be tied to one (or more) non-member right, then the "d" right and all

of the member rights need to be tied to the same non-member right(s) (\*\*).

If a client includes the "c" right in a rights list, then it MUST be treated as if the client had included every member of the "create" right. (It is not an error for a client to specify both the "c" right and one or more members of the "create" right, but the effect is no different than if just the "c" right or all members of the "create" right had been specified).

When any of the "create" member rights is set in a list of rights, the server MUST also include the "c" right when returning the list in a MYRIGHTS or ACL response. This is so to enable older clients conforming to [RFC 2086](#) to work with newer servers. (\*)

```
Example:  C: A003 SETACL INBOX.Drafts Byron lrswikda
          S: A001 OK Setacl complete
          C: A002 GETACL INBOX.Drafts
          S: * ACL INBOX Fred rwipslxeta Byron lrswikcdeta
          S: A002 OK Getacl complete
```

The client has specified the "d" right in the SETACL command above and it expands to "et" on the server: As the client has specified the "k" right (which is a member of the "c" right), the server also returns the "c" right.

If the authentication identifier specified in the LISTRIGHTS command can be granted any of the "create" member rights to access a mailbox, then the server

MUST include the "c" right in the corresponding LISTRIGHTS response. (\*)

If the member rights aren't tied to non-member rights, then the "c" right is returned by itself in the LISTRIGHTS response. If any of the member rights

needs to be tied to one (or more) non-member right, then the "c" right and all

of the member rights need to be tied to the same non-member right(s) (\*\*).

Example: The server that ties the rights as follows

lr s w i p k x t

and c=k

will return:

S: \* LISTRIGHTS archive.imap anyone "" lr s w i p k x t c d

Example: The server that ties the rights as follows

lr s w i p k xte

and c=k

will return:

S: \* LISTRIGHTS archive.imap anyone "" lr s w i p k xte c d

Example: The server that ties the rights as follows

lr s w i p k x te

and c=k

will return:

S: \* LISTRIGHTS archive.imap anyone "" lr s w i p k c x te d

Example: The server that ties the rights as follows

lr swte i p k x

and c=kx

will return:

S: \* LISTRIGHTS archive.imap anyone "" lr swted i p k x c

(\*) Clients conforming to this document MUST ignore the virtual "d" and "c" rights in MYRIGHTS, ACL and LISTRIGHTS responses.

(\*\*) The IMAPEXT Working Group has debated this issue in great length and after reviewing existing ACL implementations concluded that this is a reasonable restriction.

### **3.2. Rights defined in [RFC 2086](#).**

For convenience, this section lists the rights which were defined in the [RFC 2086](#):

"l", "r", "s", "w", "i", "p", "a", "c", "d" and digit rights

("0" .. "9").

The listed rights MUST NOT be returned in the "RIGHTS=" capability string.

#### **4. Access control management commands and responses**

Servers, when processing a command that have an authentication identifier as a parameter (i.e. any of SETACL, DELETEACL and LISTRIGHTS commands), SHOULD first prepare the received identifier using "SASLprep" profile [[SASLprep](#)] of the "stringprep" algorithm [StringPrep]. If the preparation of the identifier fails or results in an empty string, the server MUST refuse to perform the command with a BAD response.

##### **4.1. SETACL command**

Arguments: mailbox name  
            authentication identifier  
            access right modification

Data: no specific data for this command

Result: OK - setacl completed  
        NO - setacl failure: can't set acl  
        BAD - arguments invalid

The SETACL command changes the access control list on the specified mailbox so that the specified identifier is granted permissions as specified in the third argument.

The third argument is a string containing an optional plus ("+" ) or minus ("-") prefix, followed by zero or more rights characters. If the string starts with a plus, the following rights are added to any existing rights for the identifier. If the string starts with a minus, the following rights are removed from any existing rights for the identifier. If the string does not start with a plus or minus, the rights replace any existing rights for the identifier.

Note that an unrecognized right MUST cause the command to return the BAD response. In particular, server MUST NOT silently ignore unrecognized rights.

Example: C: A001 SETACL INBOX.Drafts Chris lrswicda  
        S: A001 OK Setacl complete

C: A035 SETACL INBOX.Drafts John lrQswicda  
S: A035 BAD Uppercase rights are not allowed

```
C: A036 SETACL INBOX.Drafts John lrgswicda
S: A036 BAD The q right is not supported
```

#### **4.2. DELETEACL command**

Arguments: mailbox name  
authentication identifier

Data: no specific data for this command

Result: OK - deleteacl completed  
NO - deleteacl failure: can't delete acl  
BAD - arguments invalid

The DELETEACL command removes any <identifier,rights> pair for the specified identifier from the access control list for the specified mailbox.

```
Example: C: B001 GETACL INBOX
S: * ACL INBOX Fred rwipslxeta -Fred wet $team w
S: B001 OK Getacl complete
C: B002 DELETEACL INBOX Fred
S: B002 OK Deleteacl complete
C: B003 GETACL INBOX
S: * ACL INBOX -Fred wet $team w
S: B003 OK Getacl complete
```

#### **4.3. GETACL command**

Arguments: mailbox name

Data: untagged responses: ACL

Result: OK - getacl completed  
NO - getacl failure: can't get acl  
BAD - arguments invalid

The GETACL command returns the access control list for mailbox in an untagged ACL response.

Some implementations may permit multiple forms of an authentication identifier to reference the same IMAP account. Usually, such implementations will have a canonical form that is stored internally. An ACL response caused by an GETACL command may include a canonicalized form of the authentication identifier which might be different from the one used in the corresponding SETACL command.

```
Example: C: A002 GETACL INBOX
S: * ACL INBOX Fred rwipslxeta
```



S: A002 OK Getacl complete

#### **4.4. LISTRIGHTS command**

Arguments: mailbox name  
authentication identifier

Data: untagged responses: LISTRIGHTS

Result: OK - listrights completed  
NO - listrights failure: can't get rights list  
BAD - arguments invalid

The LISTRIGHTS command takes a mailbox name and an identifier and returns information about what rights may be granted to the identifier in the ACL for the mailbox.

Some implementations may permit multiple forms of an authentication identifier to reference the same IMAP account. Usually, such implementations will have a canonical form that is stored internally. A LISTRIGHTS response caused by a LISTRIGHTS command MUST always return the same form of an authentication identifier as specified by the client. This is to allow the client to correlate the response with the command.

Example: C: a001 LISTRIGHTS ~/Mail/saved smith  
S: \* LISTRIGHTS ~/Mail/saved smith la r swicdkxte  
S: a001 OK Listrights completed

Example: C: a005 LISTRIGHTS archive.imap anyone  
S: \* LISTRIGHTS archive.imap anyone "" l r s w i p k x t  
e c d a 0 1 2 3 4 5 6 7 8 9  
S: a005 Listrights successful

#### **4.5. MYRIGHTS command**

Arguments: mailbox name

Data: untagged responses: MYRIGHTS

Result: OK - myrights completed  
NO - myrights failure: can't get rights  
BAD - arguments invalid

The MYRIGHTS command returns the set of rights that the user has to mailbox in an untagged MYRIGHTS reply.

Example: C: A003 MYRIGHTS INBOX  
S: \* MYRIGHTS INBOX rwiptsldaex  
S: A003 OK Myrights complete

#### **4.6. ACL response**

Data:            mailbox name  
                 zero or more identifier rights pairs

The ACL response occurs as a result of a GETACL command. The first string is the mailbox name for which this ACL applies. This is followed by zero or more pairs of strings, each pair contains the identifier for which the entry applies followed by the set of rights that the identifier has.

[Section 3.1.1](#) details additional server requirements related to handling of the virtual "d" and "c" rights.

#### **4.7. LISTRIGHTS response**

Data:            mailbox name  
                 identifier  
                 required rights  
                 list of optional rights

The LISTRIGHTS response occurs as a result of a LISTRIGHTS command. The first two strings are the mailbox name and identifier for which this rights list applies. Following the identifier is a string containing the (possibly empty) set of rights the identifier will always be granted in the mailbox.

Following this are zero or more strings each containing a set of rights the identifier may be granted in the mailbox. Rights mentioned in the same string are tied together--either all must be granted to the identifier in the mailbox or none may be granted. [Section 3.1.1](#) details additional server requirements related to handling of the virtual "d" and "c" rights.

The same right may not be listed more than once in the LISTRIGHTS command.

#### **4.8. MYRIGHTS response**

Data:            mailbox name  
                 rights

The MYRIGHTS response occurs as a result of a MYRIGHTS command. The first string is the mailbox name for which these rights apply. The second string is the set of rights that the client has.

[Section 3.1.1](#) details additional server requirements related to handling of the virtual "d" and "c" rights.

## 5. Rights required to perform different IMAP4rev1 commands

Before executing a command an ACL compliant server must check which rights are required to perform it. This section groups command by functions they perform and list the rights required. It also gives the detailed description of any special processing required.

For the purpose of this section the UID counterpart of a command is considered to be the same command, e.g. both UID COPY and COPY commands require the same set of rights.

The table below summarizes different rights or their combinations that are required in order to perform different IMAP operations. As it is not always possible to express complex right checking and interactions, the description after the table should be used as the primary reference.

Operations\Rights	l	r	s	w	i	k	x	t	e	a	Any	None
LIST	+											
SUBSCRIBE	*											*
UNSUBSCRIBE												+
LSUB	*											*
CREATE (for parent)						+						
DELETE							+	?	?			
RENAME						+	+					
COPY/APPEND			?	?	+			?				
EXPUNGE/CLOSE									+			
SELECT/EXAMINE/STATUS		+										
FETCH			?									
STORE flags			?	?				?				
SETACL/DELETEACL										+		
GETACL/LISTRIGHTS										+		
MYRIGHTS											+	

Legend:

+ - The right is required

\* - Only one of the rights marked with \* is required (see description below)

? - The right is optional (see description below)

"Any" - at least one of the "l", "r", "i", "k", "x", "a" rights is required

"None" - No rights required to perform the command

Listing and subscribing/unsubscribing mailboxes:

LIST - "l" right is required.

Note that if the user has "l" right to a mailbox "A/B", but not to its

parent

mailbox "A", the LIST command should behave as if the mailbox "A" doesn't exist,

for example:

```
C: A777 LIST "" *
S: * LIST (\NoInferiors) "/" "A/B"
S: * LIST () "/" "C"
S: * LIST (\NoInferiors) "/" "C/D"
S: A777 OK LIST completed
```

SUBSCRIBE - "l" right is required only if the server checks for mailbox existence

when performing SUBSCRIBE.

UNSUBSCRIBE - no rights required to perform this operation.

LSUB - "l" right is required only if the server checks for mailbox existence when

performing SUBSCRIBE.

Mailbox management:

CREATE - "k" right on a nearest existing parent mailbox. When a new mailbox is created it SHOULD inherit the ACL from the parent mailbox (if one exists) in the defined hierarchy.

DELETE - "x" right on the mailbox. Note that some servers don't allow to delete a non-empty mailbox. If this is the case, the user would also need "r", "e" and "t" rights, in order to open the mailbox and empty it.

The DELETE command MUST delete the ACL associated with the deleted mailbox.

RENAME - Moving a mailbox from one parent to another requires the "x" right on the mailbox itself and the "k" right for the new parent. For example, if the user wants to rename mailbox named "A/B/C" to "D/E", the user must have the "x" right for the mailbox "A/B/C" and the "k" right for the mailbox "D".

The RENAME command SHOULD NOT change the ACLs on the renamed mailbox and submailboxes.

Copying or appending messages:

Before performing a COPY/APPEND command the server MUST check if the user has "i" right for the target mailbox. If the user doesn't have "i" right, the operation fails. Otherwise for each copied/appended message the server MUST check if the user has

"t" right - when the message has \Deleted flag set

"s" right - when the message has \Seen flag set

"w" right for all other message flags.

Only when the user has a particular right the corresponding flags are

stored for the newly created message. The server MUST NOT fail a COPY/APPEND if the user has no rights to set a particular flag.

Example: C: A003 MYRIGHTS TargetMailbox  
S: \* MYRIGHTS TargetMailbox rwis  
S: A003 OK Myrights complete  
  
C: A004 FETCH 1:3 (FLAGS)  
S: \* 1 FETCH (FLAGS (\Draft \Deleted))  
S: \* 2 FETCH (FLAGS (\Answered))  
S: \* 3 FETCH (FLAGS (\$Forwarded \Seen))  
S: A004 OK Fetch Completed  
  
C: A005 COPY 1:3 TargetMailbox  
S: A005 OK Copy completed  
  
C: A006 SELECT TargetMailbox  
...  
S: A006 Select Completed

Let's assume that the copied messages received message numbers 77:79.

C: A007 FETCH 77:79 (FLAGS)  
S: \* 77 FETCH (FLAGS (\Draft))  
S: \* 78 FETCH (FLAGS (\Answered))  
S: \* 79 FETCH (FLAGS (\$Forwarded \Seen))  
S: A007 OK Fetch Completed

\Deleted flag was lost on COPY, as the user has no "t" right in the target mailbox.

If the MYRIGHTS command with the tag A003 would have returned:  
S: \* MYRIGHTS TargetMailbox rsti

the response from the FETCH with the tag A007 would have been:

C: A007 FETCH 77:79 (FLAGS)  
S: \* 77 FETCH (FLAGS (\Deleted))  
S: \* 78 FETCH (FLAGS ())  
S: \* 79 FETCH (FLAGS (\Seen))  
S: A007 OK Fetch Completed

In the latter case \Answered, \$Forwarded and \Draft flags were lost on COPY, as the user has no "w" right in the target mailbox.

Expunging the selected mailbox:  
EXPUNGE - "e" right on the selected mailbox.

CLOSE - "e" right on the selected mailbox. If the server is unable to expunge the mailbox because the user doesn't have the "e" right, the server MUST ignore expunge request, close the mailbox and return tagged OK response.

Fetch information about a mailbox and its messages:

SELECT/EXAMINE/STATUS - "r" right on the mailbox.

FETCH - A FETCH request that implies setting \Seen flag MUST NOT set it, if the current user doesn't have "s" right.

Changing flags:

STORE - the server MUST check if the user has

"t" right - when the user modifies \Deleted flag

"s" right - when the user modifies \Seen flag

"w" right for all other message flags.

STORE operation SHOULD NOT fail if the user has rights to modify at least one flag specified in the STORE, as the tagged NO response to a STORE command is not handled very well by deployed clients.

Changing ACLs:

SETACL/DELETEACL - "a" right on the mailbox.

Reading ACLs:

GETACL - "a" right on the mailbox.

MYRIGHTS - any of the following rights is required to perform the operation: "l", "r", "i", "k", "x", "a".

LISTRIGHTS - "a" right on the mailbox.

## **6. Other considerations**

### **6.1. Additional requirements and Implementation notes**

#### **6.1.1. Servers**

This document defines an additional capability that is used to announce the list of extra rights (excluding the ones defined in the [RFC 2086](#)) supported by the server. The set of rights MUST include "t", "e", "x" and "k". Note that the extra rights can appear in any order.

Example:     C: 1 capability  
              S: \* CAPABILITY IMAP4REV1 STARTTLS LITERAL+ ACL RIGHTS=texk  
              S: 1 OK completed

Any server implementing an ACL extension MUST accurately reflect the current user's rights in FLAGS and PERMANENTFLAGS responses.

Example:     C: A142 SELECT INBOX  
              S: \* 172 EXISTS  
              S: \* 1 RECENT  
              S: \* OK [UNSEEN 12] Message 12 is first unseen  
              S: \* OK [UIDVALIDITY 3857529045] UIDs valid  
              S: \* OK [UIDNEXT 4392] Predicted next UID

```

S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Seen \Answered \Flagged \*)] Limited
S: A142 OK [READ-WRITE] SELECT completed
C: A143 MYRIGHTS INBOX
S: * MYRIGHTS INBOX lrwis
S: A143 OK completed

```

Note that in order to get better performance the client may pipeline SELECT and MYRIGHTS commands:

```

C: A142 SELECT INBOX
C: A143 MYRIGHTS INBOX
S: * 172 EXISTS
S: * 1 RECENT
S: * OK [UNSEEN 12] Message 12 is first unseen
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: * OK [UIDNEXT 4392] Predicted next UID
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Seen \Answered \Flagged \*)] Limited
S: A142 OK [READ-WRITE] SELECT completed
S: * MYRIGHTS INBOX lrwis
S: A143 OK completed

```

Servers MAY cache the rights a user has on a mailbox when the mailbox is selected, so that if a client's rights on a mailbox are changed with SETACL or DELETEACL, commands specific to the selected state (e.g., STORE, EXPUNGE) might not reflect the changed rights until the mailbox is re-selected. If the server checks the rights on each command, then it SHOULD send FLAGS and PERMANENTFLAGS responses if they have changed.

An ACL server MAY modify one or more ACL for one or more identifier as a side effect of modifying the ACL specified in a SETACL/DELETEACL. If the server does that it MUST send untagged ACL response to notify the client about the changes made.

### **6.1.2. Clients**

A client implementation that allows a user to read and update ACLs MUST preserve unrecognized rights that it doesn't allow the user to change when updating the rights. Otherwise the client may unintentionally remove permissions.

## **6.2. Mapping of ACL rights to READ-WRITE and READ-ONLY response codes**

A particular ACL server implementation may allow "shared multiuser access" to some mailboxes. "Shared multiuser access" to a mailbox means that multiple different users are able to access the same mailbox, if they have proper access rights. "Shared multiuser access" to the mailbox doesn't mean that the ACL for the mailbox is currently set to allow access by multiple users. Let's denote a "shared multiuser

write access" as a "shared multiuser access" when a user may be granted flag modification rights (any of "w", "s" or "t").

[Section 5](#) describes which rights are required for modifying different flags.

If the ACL server implements some flags as shared for a mailbox (i.e., the ACL for the mailbox may be set up so that changes to those flags are visible to another user), let's call the set of rights associated with these flags (as described in [Section 5](#)) for that mailbox collectively as "shared flag rights". Note that "shared flag rights" set MAY be different for different mailboxes.

If the server doesn't support "shared multiuser write access" to a mailbox or doesn't implement shared flags on the mailbox, "shared flag rights" for the mailbox is defined to be the empty set.

Example 1: Mailbox "banan" allows "shared multiuser write access" and implements flags \Deleted, \Answered and \$MDNSent as shared flags. "Shared flag rights" for the mailbox "banan" is a set containing flags "t" (because system flag \Deleted requires "t" right) and "w" (because both \Answered and \$MDNSent require "w" right).

Example 2: Mailbox "apple" allows "shared multiuser write access" and implements \Seen system flag as shared flag. "Shared flag rights" for the mailbox "apple" contains "s" right, because system flag \Seen requires "s" right.

Example 3: Mailbox "pear" allows "shared multiuser write access" and implements flags \Seen, \Draft as shared flags. "Shared flag rights" for the mailbox "apple" is a set containing flags "s" (because system flag \Seen requires "s" right) and "w" (because system flag \Draft requires "w" right).

The server MUST include a READ-ONLY response code in the tagged OK response to

a SELECT command if none of the following rights is granted to the current user:

"i", "e" and "shared flag rights".

The server SHOULD include a READ-WRITE response code in the tagged OK response

if at least one of the "i", "e" or "shared flag rights" is granted to the current user.

\* - Note that a future extension to this document may extend the list of rights that causes the server to return the READ-WRITE response code.

Example 1 (continued): The user that has "lrs" rights for the mailbox "banan". The server returns READ-ONLY response code on SELECT, as none of "iewt" rights is granted to the user.



Example 2 (continued): The user that has "rit" rights for the mailbox "apple". The server returns READ-WRITE response code on SELECT, as the user has "i" right.

Example 3 (continued): The user that has "rset" rights for the mailbox "pear". The server returns READ-WRITE response code on SELECT, as the user has "e" and "s" rights.

## **7. Security Considerations**

An implementation must make sure the ACL commands themselves do not give information about mailboxes with appropriately restricted ACL's. For example, a GETACL command on a mailbox for which the user has insufficient rights should not admit that the mailbox exists, much less return the mailbox's ACL.

LISTRIGHTS command MUST NOT check that a particular identifier exists, however it SHOULD recognize special identifiers like "anyone".

IMAP clients implementing ACL that are able to modify ACLs SHOULD warn a user that wants to give full access (or even just the "a" right) to the special identifier "anyone".

## **8. Formal Syntax**

Formal syntax is defined using ABNF [[ABNF](#)] as modified by [[IMAP4](#)]. Non-terminals referenced but not defined below are as defined by [[IMAP4](#)].

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.

LOWER\_ALPHA = %x61-7A ;; a-z

acl\_data = "ACL" SP mailbox \*(SP identifier SP rights)

deleteacl = "DELETEACL" SP mailbox SP identifier

getacl = "GETACL" SP mailbox

identifier = astring

listrights = "LISTRIGHTS" SP mailbox SP identifier

listrights\_data = "LISTRIGHTS" SP mailbox SP identifier  
SP rights \*(SP rights)

```

mod_rights      = astring
                  ;; +rights to add, -rights to remove
                  ;; rights to replace

myrights        = "MYRIGHTS" SP mailbox

myrights_data    = "MYRIGHTS" SP mailbox SP rights

new_rights       = 1*LOWER_ALPHA
                  ;; MUST include "t", "e", "x" and "k".
                  ;; MUST NOT include standard rights listed
                  ;; in section 3.2

rights          = astring
                  ;; only lowercase ASCII letters and digits
                  ;; are allowed.

rights_capa      = "RIGHTS=" new_rights
                  ;; RIGHTS=... capability

setacl          = "SETACL" SP mailbox SP identifier
                  SP mod_rights

```

## 9. IANA Considerations

IMAP4 capabilities are registered by publishing a standards track or IESG approved experimental RFC. The registry is currently located at:

<http://www.iana.org/assignments/imap4-capabilities>

This document defines the RIGHTS= IMAP capability. IANA is requested to add this capability to the registry.

## 10. References

### 10.1. Normative References

[KEYWORDS] Bradner, "Key words for use in RFCs to Indicate Requirement Levels", [RFC 2119](#), Harvard University, March 1997.

[ABNF] Crocker, Overell, "Augmented BNF for Syntax Specifications: ABNF", [RFC 2234](#), Internet Mail Consortium, Demon Internet Ltd, November 1997.

[IMAP4] Crispin, M., "Internet Message Access Protocol - Version 4rev1", [RFC 3501](#), University of Washington, March 2003.

[UTF-8] Yergeau, F., "UTF-8, a transformation format of ISO 10646",

[RFC 3629](#), Alis Technologies, November 2003.

[Stringprep] Hoffman, P., Blanchet, M., "Preparation of Internationalized Strings ("stringprep")", [RFC 3454](#), December 2002.

[SASLprep] Zeilenga, K., "SASLprep: Stringprep profile for user names and passwords", Work in progress, [draft-ietf-sasl-saslprep-XX.txt](#).

## **[10.2.](#) Informative References**

[RFC2086] Myers, J., "IMAP4 ACL extension", [RFC 2086](#), Carnegie Mellon, January 1997.

## **[11.](#) Editor's Address**

Alexey Melnikov  
email: [alexey.melnikov@isode.com](mailto:alexey.melnikov@isode.com)

Isode Limited

## **[12.](#) IPR Disclosure Acknowledgement**

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with [RFC 3668](#).

## **[13.](#) Intellectual Property Statement**

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in [BCP-11](#). Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

#### **14. Full Copyright Statement**

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in [BCP 78](#) and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

#### Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

#### **Appendix A. Changes since [RFC 2086](#)**

1. Changed the charset of "identifier" from US-ASCII to UTF-8.
2. Specified that mailbox deletion is controlled by the "x" right and EXPUNGE is controlled by the "e" right.
3. Added the "t" right that controls STORE \Deleted. Redefined the "d" right to be a macro for "e", "t" and possibly "x".
4. Added the "k" right that controls CREATE. Redefined the "c" right to be a macro for "k" and possibly "x".
5. Specified that the "a" right also controls DELETEACL.
6. Specified that the "r" right also controls STATUS.
7. Removed the requirement to check the "r" right for CHECK, SEARCH and FETCH, as this is required for SELECT/EXAMINE to be successful.
8. LISTRIGHTS requires the "a" right on the mailbox (same as SETACL).
9. Deleted "PARTIAL", this is a deprecated feature of [RFC 1730](#).
10. Specified that the "w" right controls setting flags other than \Seen and \Deleted on APPEND. Also specified that the "s" right controls the \Seen flag and that the "t" right controls the \Deleted flag.
11. Specified that SUBSCRIBE is NOT allowed with the "r" right.

12. Specified that the "l" right controls SUBSCRIBE.
13. GETACL is NOT allowed with the "r" right, even though there are several implementations that allows that. If a user only has "r" right, GETACL can disclose information about identifiers existing on the mail system.
14. Clarified that RENAME requires the "k" right for the new parent and the "x" right for the old name.
15. Added new section that describes which rights are required and/or checked when performing various IMAP commands.
16. Added mail client security considerations when dealing with special identifier "anyone".
17. Clarified that negative rights are not the same as DELETEACL.
18. Added "Compatibility with [RFC 2086](#)" section.
19. Added section about mapping of ACL rights to READ-WRITE and READ-ONLY response codes.
20. Changed BNF to ABNF.
21. Added "Implementation Notes" section.
22. Updated "References" section.
23. Added more examples.
24. Clarified when the virtual "c" and "d" rights are returned in ACL, MYRIGHTS and LISTRIGHTS responses.

## **[Appendix B](#). Compatibility with [RFC 2086](#)**

This section gives guidelines how an existing [RFC 2086](#) server implementation may be updated to comply with this document.

This document splits the "d" right into several new different rights: "t", "e" and possibly "x" (see [section 3.1.1](#) for more details). The "d" right remains for backwards-compatibility but it is a virtual right. The server should implement one of the following two approaches to handle the "d" right and the new rights that have replaced it.

- a). "t", "e" (and possibly "x") together - almost no changes.
- b). Implement separate "x", "t" and "e". Return the "d" right in a MYRIGHTS response or an ACL response containing ACL information when any of the "t", "e" (and "x") is granted.

In a similar manner this document splits the "c" right into several

new different rights: "k" and possibly "x" (see [section 3.1.1](#) for more details). The "c" right remains for backwards-compatibility but it is a virtual right. The server should implement one of the two approaches described above.

Also check Sections [6.1](#) and [6.2](#), as well as the [appendix A](#) to see other changes required. Server implementors should check which rights are required to invoke different IMAP4 commands as described in [Section 5](#).

## [Appendix C](#). Known deficiencies

This specification has some known deficiencies including:

1. This is inadequate to provide complete read-write access to mailboxes protected by Unix-style rights bits because there is no equivalent to "chown" and "chgrp" commands nor is there a good way to discover such limitations are present.
2. Because this extension leaves the specific semantics of how rights are combined by the server as implementation defined, the ability to build a user-friendly interface is limited.

The work-in-progress "ACL2" extension is intended to redesign this extension to address these deficiencies without the constraint of backwards-compatibility and may eventually supercede this facility. However, [RFC 2086](#) is deployed in multiple implementations so this intermediate step which fixes the straightforward deficiencies in a backwards compatible fashion is considered worthwhile.

## [Appendix D](#). Acknowledgment

This document is a revision of the [RFC 2086](#) written by John G. Myers.

Editor appreciates comments received from Mark Crispin, Chris Newman, Cyrus Daboo, John G. Myers, Dave Cridland, Ken Murchison, Steve Hole, Vladimir Butenko, Larry Greenfield, Robert Siemborski, Harrie Hazewinkel, Philip Guenther, Brian Candler, Curtis King, Lyndon Nerenberg and other participants of the IMAPEXT working group.