

## IMAP4 ACL extension

### Status of this Memo

This document is an Internet Draft and is in full conformance with all provisions of [Section 10 of RFC 2026](#).

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.

### 0. Open issues

This section will be removed when the draft will be published as RFC. It is intended to simplify discussion.

- 1). Require support for special identifier "disabled" for "ACL2=MOST-SPECIFIC" model?
- 2). Obsolete LISTRIGHTS?

Mark Crispin wrote:

- > [...] 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 by using the LISTRIGHTS command.  
 >  
 > This was in [RFC 2086](#) as well.  
 >  
 > Do we really want these semantics?  
 >  
 > The effect of this is that it is mandatory to do a LISTRIGHTS prior to any  
 > SETACL (and multiple LISTRIGHTS prior to any REPLACEACL), since otherwise  
 > there is no way for a client to know how to grant a particular right to an  
 > identifier.  
 >  
 > Presumably, the reason for these semantics is so a client can review the  
 > impact of granting a particular right if it's tied to another. But will  
 > any client actually do this? Or will clients just blindly do the  
 > LISTRIGHTS to get the necessary incantation for SETACL?  
 >  
 > What I see is no savings of work for a client which cares about any side  
 > effects due to tied-rights, but a lot of extra work for clients which do  
 > not care.

Alexey Melnikov replied:

I tend to agree. I would rather have a new response code LISTRIGHTS that is  
 returned in NO response to SETACL command.  
 Imagine that the client wants to grant "cp" rights to a user "smith". The  
 client instead of doing:

```
C: A001 LISTRIGHTS ~/Mail/saved smith
S: * LISTRIGHTS ~/Mail/saved smith la r swi cdext
S: A001 OK Listrights completed
C: A002 SETACL ~/Mail/saved smith cdextp
S: A002 OK Setacl complete
```

will do the following (this assumes that the server doesn't want to grant a  
 right unless all tied rights are specified by the client):

```
C: A001 SETACL ~/Mail/saved smith cp
S: A001 NO [LISTRIGHTS la r swi cdext] No some rights are
tied,
specify both
C: A002 SETACL ~/Mail/saved smith cdextp
S: A002 OK Setacl complete
C: A002 LISTRIGHTS ~/Mail/saved smith
```

and if the server sets all tied rights automatically if at least one of them  
 is specified (not recommended by the document but still possible):

```
C: A001 SETACL ~/Mail/saved smith cp
S: * ACL ~/Mail/saved smith cdextp
```

S: A001 OK Setacl complete

3). Other editorial comments/questions are enclosed in << and >>.

## **1. Abstract**

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

## **2. 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](#)].

## **3. Introduction and Overview**

The ACL extension is present in any IMAP4 implementation which returns a capability starting with "ACL2=" as one of the supported capabilities to the CAPABILITY command.

An access control list is a set of <identifier,rights> pairs.

Identifier is a UTF-8 string. An identifier may have one of the following forms:

- a). "anyone" - special identifier that refers to the universal identity (all authentications, including anonymous).
- b). "authuser" - special identifier that refer to all authenticated users, but not anonymous.
- c). "owner" - special identifier that refers to the owner of a mailbox (if any).
- d). "administrators" - special identifier that refers to all users with administrative rights.
- e). "user=<xxx>" - refers to a user. Here "<xxx>" is a user name string accepted by the LOGIN or AUTHENTICATE commands.
- f). "group=<xxx>" - refers to a group.
- g). "-<identifier>", where <identifier> is one of a).-f). This is reserved for "negative rights", described below.

<<Reserve X- and V- prefixes?>>

Note that a server is not required to implement any special identifier mentioned

above. However if it allows a user to perform ACL operations on any one of them,

server MUST use the semantic as described above.

All other identifier names are reserved for future definition in an extension or revision to this specification (also known as ACL2).

Rights is a string listing a (possibly empty) set of alphanumeric characters, each character listing a set of operations which is being controlled. Letters are reserved for 'standard' rights, listed below. The set of standard rights may only be extended by a standards-track document. Digits are reserved for implementation or site defined rights. The currently defined standard rights are:

- l - lookup (mailbox is visible to LIST/LSUB commands, SUBSCRIBE mailbox)
- r - read (SELECT the mailbox, perform STATUS, CHECK, FETCH, SEARCH, COPY from mailbox)
- 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)
- c - create mailboxes (CREATE new sub-mailboxes in any implementation-defined hierarchy, parent mailbox for the new mailbox name in RENAME).  
When a new mailbox is created it SHOULD inherit rights from the parent mailbox (if one exists) in the defined hierarchy.
- 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
- d - This right is defined for backward compatibility with ACL extension ([RFC 2086](#)). If a client sets "d" right, the server MUST set "x", "e" and "t" rights. When the client clears the "d" right, the server MUST clear "x", "e" and "t" rights. When all three of "x", "e" and "t" are set, the server MUST return "d" right in response to a ACL GET command. If "x", "e" and "t" rights are not tied together, "d" right MUST NOT be returned in a ACL LIST response.
- a - administer (perform ACL STORE, ACL SET and ACL DELETE)

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 "wisd" 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 ACL STORE 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 by using the ACL LIST 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 ACL DELETE in that a negative right is added to the ACL, and is part of the calculation of the rights.

For example, if the identifier "-user=fred" is granted the "w" right, that indicates that the "w" right is to be removed from users matching the identifier "user=fred", even though the user "fred" might have the "w" right as a consequence of some other identifier in the ACL. A ACL DELETE of "user=fred" simply deletes the identifier "user=fred" from the ACL; it does not affect any rights that the user "fred" may get from another ACL.

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

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. The set of rules that describes how access is calculated is defined by a rule identifier (rule-ID).

A client may determine the set of rights granted to the logged-in user for a given mailbox by using the MYRIGHTS command.

If a server implementing ACL2 uses the union of the rights granted to the applicable identifiers minus the union of the negative rights in order to calculate access, it MUST report "ACL2=UNION" in the server's capability list.

An implementation may instead choose to only use those rights granted to the most specific identifier present in the ACL. In this case the server MUST report "ACL2=MOST-SPECIFIC" in the server's capability list.

If the server implements any other policy for rights calculation, it MUST be either registered with IANA using the template provided in 7.1 or start with "X-". The server MUST report one and only one "ACL2=<rule-ID>" capability in its CAPABILITY response.

### **3.1. Rights required to perform different IMAP4rev1 commands**

Before executing a command an ACL2 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.

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.

<<Allow if the user has "r" right?>>

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

DELETE - "x" right on the mailbox.

RENAME - Moving a mailbox from one parent to another requires "x" right on the mailbox itself and "c" 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 "x" right for the mailbox "A/B/C" and "c" right for the mailbox "D".

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.

<<Add an example here?>>

Expunging the selected mailbox:

EXPUNGE - "e" right on the selected mailbox.

CLOSE - "e" right on the selected mailbox. Server MUST NOT fail CLOSE operation if it is unable to expunge the mailbox.

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.

Changing ACLs:

ACL STORE/DELETE/SET - "a" right on the mailbox (\*).

Reading ACLs:

ACL GET - "a" right on the mailbox (\*).

<<Several implementations allow GETACL (ACL GET) with "r" right. If this is allowed,

ACL GET can disclose some identifiers existing on a mail system>>

MYRIGHTS - any of the following rights is required to perform the operation:

"l", "r", "i", "c", "x", "e", "a".

ACL LIST - same as for MYRIGHTS(\*).

(\*) Note, that when one or more mailbox pattern is specified, 'l' right is required for each mailbox matching the mailbox pattern(s).

## 4. Commands

All ACL commands (i.e. ACL STORE/DELETE/SET/GET/LIST, except for MYRIGHTS) accept either a single mailbox name or several mailbox patterns as a parameter. Mailbox pattern is a mailbox with wildcards, wildcards are interpreted as described in [\[IMAP4\]](#) LIST command. In order to distinguish between a mailbox name (that is allowed to have wildcard characters '\*' and '%') and a mailbox pattern, the latter is always represented as a parenthesized list.

For simplicity the behaviour of ACL STORE/DELETE/SET/GET/LIST commands is described for a single mailbox case. When one or more mailbox pattern is specified, the server internally performs LIST command for all specified patterns and then it combines the results. Note, that only mailboxes for which the user has 'l' right are included in the combined result. If the combined result has no mailboxes, an ACL operation completes with success and the tagged OK response is sent. Otherwise the requested operation is performed for each mailbox in the combined result. If a particular mailbox causes the operation to fail (e.g. insufficient permissions), instead of failing the whole command, an untagged ACLFAILED response is sent for this mailbox and the operation continues for the rest of the mailboxes. If the server knows that the operation will fail in the same manner for all matching mailboxes (e.g. user doesn't exist), it SHOULD return tagged NO response instead of sending several untagged ACLFAILED responses.

Example:     C: A002 ACL SET (INBOX Personal/\*) user=Fred rwist  
              S: \* ACLFAILED Personal/ABC  
              S: A002 OK acl set complete

<<Add some responses caused by side effects>>

Example:     C: A002 ACL SET (Fruits/Apples/\*) user=Zak lrs  
              S: A002 NO User Zak doesn't exist

### 4.1. ACL STORE

Arguments: mailbox name or one or more mailbox masks  
            authentication identifier  
            access right modification

Data:       OPTIONAL untagged responses: ACLINFO

Result:     OK - ACL STORE completed  
            NO - ACL STORE failure: can't set acl  
            BAD - command unknown or arguments invalid

The ACL STORE 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 for "ACL2=UNION" access calculation rule <ACL STORE mailbox identifier "> MUST be treated as <ACL DELETE mailbox identifier>. Also note that these two commands don't have the same result for "ACL2=MOST-SPECIFIC".

An ACL2 server MAY modify one or more ACL for one or more identifier as a side effect of modifying the ACL specified in ACL STORE. If the server does that it MUST send untagged ACLINFO response to notify the client about the changes made.

#### **4.2. ACL DELETE**

Arguments: mailbox name or one or more mailbox masks  
authentication identifier

Data: OPTIONAL untagged responses: ACLINFO

Result: OK - ACL DELETE completed  
NO - ACL DELETE failure: can't delete acl  
BAD - command unknown or arguments invalid

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

An ACL2 server MAY modify one or more ACL for one or more identifier as a side effect of modifying the ACL specified in ACL DELETE. If the server does that it MUST send untagged ACLINFO response to notify the client about the changes made.

#### **4.3. ACL SET**

Arguments: mailbox name or one or more mailbox masks  
list of (authentication identifier, access rights) pairs

Data: OPTIONAL untagged responses: ACLINFO

Result: OK - replaceacl completed  
NO - replaceacl failure: can't replace acl

BAD - command unknown or arguments invalid

The ACL SET command replaces the access control list of the specified mailbox with the one provided as the second parameter to ACL SET. This command is semantically equivalent to the following sequence of commands:

- 1). ACL GET <mailbox name>
- 2). For each (authentication identifier AID, access rights RD) pair returned  
in the untagged ACL response that was caused by ACL GET, perform  
ACL DELETE <mailbox name> AID
- 3). For each (authentication identifier AIA, access rights RA) pair from  
the second parameter of ACL SET perform  
ACL STORE <mailbox name> AIA RA

An ACL2 server MAY modify one or more ACL for one or more identifier as a side effect of modifying the ACL specified in ACL SET. If the server does that it MUST send untagged ACLINFO response to notify the client about the changes made.

#### **4.4. ACL GET**

Arguments: mailbox name or one or more mailbox masks

Data: REQUIRED untagged responses: ACLINFO

Result: OK - ACL GET completed  
NO - ACL GET failure: can't get acl  
BAD - command unknown or arguments invalid

The ACL GET command returns the access control list for mailbox in an untagged ACLINFO reply.

Example: C: A002 ACL GET INBOX  
S: \* ACLINFO INBOX user=Fred rwipslextda  
S: A002 OK acl get complete

#### **4.5. ACL LIST**

Arguments: mailbox name or one or more mailbox masks  
authentication identifier

Data: untagged responses: RIGHTS-INFO

Result: OK - ACL LIST completed  
NO - ACL LIST failure: can't get rights list  
BAD - command unknown or arguments invalid

The ACL LIST 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.

```
Example:  C: a001 ACL LIST ~/Mail/saved user=smith
          S: * RIGHTS-INFO ~/Mail/saved user=smith l a r s w i c d e x t
          S: a001 OK ACL LIST completed

          C: a005 ACL LIST archive.imap anyone
          S: * RIGHTS-INFO archive.imap anyone "" l r s w i p c d t e x a
            0 1 2 3 4 5 6 7 8 9
          S: a005 OK ACL LIST completed
```

#### **4.6. MYRIGHTS**

```
Arguments: mailbox name

Data:      untagged responses: MYRIGHTS

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

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

<<There is a proposal to replace MYRIGHT with either ACL FETCH or ACL CHECK:

ACL FETCH is slightly different than MYRIGHTS in that it takes an identifier as an argument and returns the rights for that identifier (?? is this desirable ??).

Or perhaps there could be an ACL CHECK that, given an identifier and a set of right, returns whether or not that identifier has those rights. ACL CHECK is probably more useful.

For now MYRIGHTS is left as is, because neither parameters, nor responses were

affected by the change to identifiers. However, if there is a desire to accept multiple mailboxes or mailbox masks in MYRIGHT, it would be better to change it to ACL <something> as well.>>

The user must have any of the following rights to perform this operation:

"l", "r", "i", "c", "x", "e", "a".

If the user doesn't have any right from the above list, the server MUST behave as if the mailbox doesn't exist.

```
Example:  C: A003 MYRIGHTS INBOX
          S: * MYRIGHTS INBOX rwipsldexta
          S: A003 OK Myrights complete
```

## **5. Responses**

### **5.1. ACLINFO**

Data:            mailbox name  
                 zero or more identifier rights pairs

The ACLINFO response occurs as a result of a ACL GET command. It MAY also occur as a result of ACL STORE/DELETE/SET. 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.

<< There is a proposal to use LISTEXT syntax instead. This requires publishing a new revision of LISTEXT, that in its current form, is not flexible enough. >>

### **5.2. RIGHTS-INFO**

Data:            mailbox name  
                 identifier  
                 required rights  
                 list of optional rights

The RIGHTS-INFO response occurs as a result of a ACL LIST 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.

The same right may not be listed more than once in the RIGHTS-INFO response.

### **5.3. ACLFAILED untagged response**

Contents:        OPTIONAL response code (failure reason)  
                 mailbox name

The ACLFAILED response containing a mailbox name indicates that the ACL operations failed for the specified mailbox. The reason for failure may be described by the response code (if included).

```
Example:  C: A002 ACL SET (INBOX Personal/*) user=Fred rwist
          S: * ACLFAILED Personal/ABC
          S: A002 OK acl set complete
```

<<Add human readable text?>>

<<Currently no existing response code can be used in ACLFAILED>>

#### **5.4. MYRIGHTS untagged 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.

#### **5.5. MYRIGHTS response code**

```
Data:      rights
```

The MYRIGHTS response code is sent in an untagged OK response that results from SELECT/EXAMINE. Also this response code can be sent at any time after opening a mailbox when the server detects that the set of rights allowed for the logged in user has changed.

The MYRIGHT response code is equivalent to MYRIGHTS untagged response for the selected mailbox.

### **6. 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.

```
acl2_command    = myrights | "ACL" SPACE acl2_subcommand
```

```
acl2_subcommand = replaceacl | deleteacl | updateacl | getacl | listrights
```

```
acl_data        = "ACLINFO" SPACE mailbox *(SPACE identifier SPACE rights)
```

```
always_granted  = rights
```

```
deleteacl       = "DELETE" SPACE mbox_or_pat SPACE identifier
```

```

getacl          = "GET" SPACE mbox_or_pat

identifier      = astring
                  ;; UTF-8

<< Specify more detailed syntax for identifiers? >>

listrights      = "LIST" SPACE mbox_or_pat SPACE identifier

listrights_data = "RIGHTS-INFO" SPACE mailbox SPACE identifier
                  SPACE always_granted *(SPACE rights)

mod_rights      = quoted
                  ;; +rights to add, -rights to remove

myrights        = "MYRIGHTS" SPACE mailbox

myrights_data   = "MYRIGHTS" SPACE mailbox SPACE rights

myrights_rspcod = "MYRIGHTS" SPACE rights

replaceacl      = "SET" SPACE mbox_or_pat *(SPACE identifier SPACE rights)

resp-text-code  =/ myrights_data

rights          = quoted
                  ;; MUST NOT contain leading "+" or "-"

updateacl
mod_rights      = "STORE" SPACE mbox_or_pat SPACE identifier SPACE

mbox_or_pat     = mailbox / patterns

patterns        = "(" list-mailbox *(list-mailbox) ")"

partialfail_rsp = "ACLFALIED" [SPACE "[" resp-text-code "]" ] SPACE mailbox
                  ;; May contain optional failure reason

```

## [7.](#) IANA considerations

### [7.1.](#) ACL access calculation rule Registration Template

When an access calculation rule for ACL2 extension is registered, the following information is supplied:

Rule Identification: specify a string that identifies this rule. Unless the rule is registered with the IANA, the rule's identification must start with "X-". The server supporting a particular rule <rule-ID> MUST report "ACL2=<rule-ID>" in the capability list.

Rule Semantics: specify how access rights for a mailbox are calculated.

Negative rights allowed: specify whether "negative right" identifiers are allowed.

Groups allowed: specify whether group identifiers are allowed.

Special Identifiers: describe whether any implementation defined aliases are allowed and define their meaning.

Contact Information: specify the postal and electronic contact information for the author of the feature.

## **8. Initial Registrations**

### **8.1. Registration: UNION access calculation rule**

Rule Identification: UNION

Rule Semantics: the union of the rights granted to the applicable identifiers minus the union of the negative rights.

Negative rights allowed: Yes.

Groups allowed: Yes, but not required.

Special Identifiers: No.

Contact Information: c.f., the "Editor's Address" section of this memo

### **8.2. Registration: MOST-SPECIFIC access calculation rule**

Rule Identification: MOST-SPECIFIC

Rule Semantics: the rights granted to the most specific identifier present in the ACL are used, i.e. if the user identifier is present, its ACL is used. If no user identifier is present, but a group that includes this user as a member is present, the group ACL is used. If neither user, nor group identifier is present, but an ACL for a special group "anyone" is present, the ACL for "anyone" is used.

Negative rights allowed: No.

Groups allowed: Yes, but not required.

Special Identifiers: No.

Contact Information: c.f., the "Editor's Address" section of this memo

## **9. 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 ACL GET command on a mailbox for which the user has insufficient rights should not admit the mailbox exists, much less return the mailbox's ACL.

IMAP clients implementing ACL2 that are able to modify ACLs SHOULD warn a user that wants to give full access to the special identifier "anyone".

## **10. Other considerations**

### **10.1. Compatibility with [RFC 2086](#)**

<< This has to be written >>

### **10.2. ACL2 interaction with QUOTA extension**

Server that implement both ACL2 and QUOTA extensions MUST use the following table to determine if a quote operation should be allowed for the user:

GETQUOTAROOT - "r" right  
<< Replace with/add "a" and "i"? >>

GETQUOTA - no rights required

SETQUOTA - implementation defined, as SETQUOTA operates on a quotaroot, not on a mailbox.

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

A particular ACL2 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 3.1](#) ("Rights required to perform different IMAP4rev1 commands") describes which rights are required for modifying different flags.

If the ACL2 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 3.1) 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).

<<Editor: I know this looks ugly, but this is required for interoperability. Please check whether this makes any sense!>>

The server MUST include a READ-ONLY prefix 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 prefix in the tagged OK response if at least one of the "i", "e" or "shared flag rights" is granted to the current user.

Example 1 (continue): 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 (continue): 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 (continue): 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.

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

Any server implementing an ACL2 extension MUST accurately reflect the current user's rights in FLAGS and PERMANENTFLAGS responses. The server SHOULD issue a MYRIGHTS response code in an untagged OK response as a result of a SELECT or EXAMINE command.

```
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 (\Deleted \Seen \*)] Limited
          S: * OK [MYRIGHTS rwiste] Allowed rights
          S: A142 OK [READ-WRITE] SELECT completed
```

An ACL2 server MAY modify one or more ACL for one or more identifier as a side effect of modifying the ACL specified in ACL STORE/DELETE/SET. If the server does that it MUST send untagged ACL response to notify the client about the changes made.

#### **11. 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 2060](#), University of Washington, December 1996.

[UTF-8] Yergeau, F., "UTF-8, a transformation format of ISO 10646", [RFC 2279](#), Alis Technologies, January 1998.

[QUOTA] Myers, J., "IMAP4 QUOTA extension", [RFC 2087](#), Carnegie Mellon University, January 1997.

#### **12. Acknowledgement**

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, Curtis King, Lyndon Nerenberg, Larry Greenfield, Vladimir Butenko, Dave Cridland and other members of IMAPEXT working group.

### **13. Editor's Address**

Alexey Melnikov  
mailto:mel@messagingdirect.com

ACI WorldWide/MessagingDirect  
59 Clarendon Road, Watford,  
Hertfordshire, United Kingdom, WD17 1FQ

Phone: +44 1923 81 2877

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

Copyright (C) The Internet Society 2002. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS 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.

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

1. Changed the charset of "identifier" from US-ASCII to UTF-8.
2. Changed identifier syntax. Now all users must start with "user=" prefix. Specified that identifiers starting with a "group=" prefix are reserved for groups and implementation defined aliases.
3. Specified that mailbox deletion is controlled by the "x" right and

EXPUNGE is controlled by "e" right.

4. Clarified that RENAME requires "c" right for the new parent and "x" right for the old name.
5. Changed capability name from "ACL" to "ACL2" because changes 2 and 3 are not backward compatible with ACL RFC.
6. Added "t" right that controls STORE \Deleted. Redefined "d" to be a macro for "e", "x" and "t".
7. Added "ACL2=UNION" and "ACL2=MOST-SPECIFIC" capabilities and IANA registration template.
8. Specified that "a" right also controls DELETEACL
9. Specified that "r" right also controls STATUS
10. Specified that "w" controls setting flags other than \Seen and \Deleted on APPEND. Same for "s" and "t" flags.
11. Specified that "l" controls SUBSCRIBE/UNSUBSCRIBE.
12. Added note about compatibility with [RFC 2086](#)
13. Added "Implementation Notes" section.
14. Updated "References" section.
15. Deleted "PARTIAL", this is a deprecated feature of [RFC1730](#).
16. Added MYRIGHT response code as per Cyrus suggestion.
17. Added REPLACEACL (ACL SET) as per Mark and Cyrus request.
18. Added new section that describes which rights are required and/or checked when performing various IMAP commands.
19. Added section about interaction of ACL2 with the QUOTA extension.
20. Added special identifiers "authuser", "owner" and "administrators".
21. Added mail client security considerations when dealing with special identifier "anyone".
22. Clarified that negative rights are not the same as DELETEACL (ACL DELETE).
23. Removed the requirement to check the "r" right for CHECK, SEARCH and FETCH, as this is required for SELECT/EXAMINE to be successful.
24. Added note that a server can modify an ACL for any identifier(s) as a side effect of performing SETACL/DELETEACL/REPLACEACL

(ACL STORE/DELETE/SET). Also specified that the server MUST send untagged ACL response if it does that. Updated command definition to include optional ACL untagged response.

25. Fixed typo in 10.1. (Was "ACL=...", not "ACL2=...")
26. Cleaned up [section 10.2](#) a bit.
27. LISTRIGHTS (ACL LIST) requires same rights as MYRIGHTS.
28. Added section about mapping of ACL rights to READ-WRITE and READ-ONLY response codes.
29. Changed ABNF for rigths/mod\_rights to be quoted instead of astring.
30. Changed syntax of ACL2 commands according to the following table:

SETACL	=> ACL STORE
DELETEACL	=> ACL DELETE
REPLACEACL	=> ACL SET
GETACL	=> ACL GET
LISTRIGHTS	=> ACL LIST

MYRIGHTS was left as is

Changed syntax of ACL2 responses:

ACL	=> ACLINFO
LISTRIGHTS	=> RIGHTS-INFO

MYRIGHTS was left as is

Any better suggestions for names are welcome.
31. Added mailbox patterns and partial failures. Updated ABNF.

#### To Do List:

- 1). Cleanup [appendix A](#) before publication as RFC, as some changes don't apply to [RFC 2086](#).