IMAPEXT Working Group
Internet Draft
Document: draft-ietf-imapext-acl-08.txt

A. Melnikov
Editor

June 2003

IMAP4 ACL extension

Status of this Memo

0. Open issues and ToDo list

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). "ACL2=MOST-SPECIFIC" model: If a user belongs to multiple groups, document how rights are calculated.

3). IANA registry for <vendorname> prefix in identifiers?

4). ACL2 interaction with QUOTA extension should be moved to "QUOTA=" document?

5). Do we need the following functionality: discover the set of rights which may be granted to a given identifier in the ACL for a given mailbox?

6). Cleanup appendix A before publication as RFC, as some changes don't apply to RFC 2086.

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

Table of Contents

1. Abstract

The ACL (Access Control List) extension of the Internet Message Access Protocol [IMAP4] permits mailbox 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 (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.

The ACL extension is present in any IMAP4 implementation which returns a capability starting with "ACL2=" prefix as one of the supported capabilities to the CAPABILITY command. The prefix is followed by "rule identifier" as described in 7.1.

The document contains the following parts: section 3.1 provides the definition of different classes of identifiers and defines standard rights; section 3.2 introduces access calculation model, i.e. it describes how to calculate from an ACL which rigths apply to a particular user; section 3.3 summarizes relationship of different access rights with IMAP commands; section 4 introduces new IMAP commands the client can use to manipulate ACLs; section 5 defines new ACL related responses; and section 9 lists important considerations for compatibility with [IMAP4], RFC 2086 and some IMAP extensions.

3.1. Access Control

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

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 for the server.
e). "user=<username>" - refers to a user. Here "<username>" is a user name string accepted by the LOGIN or AUTHENTICATE commands.
f). "group=<groupname>" - refers to a group. Here "<groupname>" is a group name.
g). "vendor=<vendorname>.<xxx>" - refers to a vendor specific special identifier, not covered by a).-f).
h). "-<identifier>", where <identifier> is one of a).-g). This is reserved for "negative rights", described below.

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.

Also note, that this document doesn't define how groups and administrators are managed.

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. The format of the submission address is not defined by this document)
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 LIST (ACL) command. If "x", "e" and "t" rights are not tied together, "d" right MUST NOT be returned in a RIGHTS-INFO 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. If the server fails an ACL modification command (ACL STORE or ACL SET) because some rights are tied, it MUST return RIGHTS-INFO untagged response (see section 5.2).

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.

3.2. Access calculation model

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). This document doesn't define any commands for manipulating a group membership.

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

This document defines two initial access calculation models: UNION and MOST-SPECIFIC.

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. See also section 7.2.1.

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. See also section 7.2.2.

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.

3.3. 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.

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	c	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			?	?				?				
ACL STORE/DELETE/SET	?									+		
LIST (ACL)	?									+		

```
   |      LIST (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", "c", "x", "e", "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 - "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.

Example: C: A003 LIST (MYRIGHTS) "" TargetMailbox
S: * LIST () "/" TargetMailbox (("MYRIGHTS" "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 LIST (MYRIGHT) command with the tag A003 would have returned:
S: * LIST () "/" TargetMailbox (("MYRIGHTS" "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.

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

Reading ACLs:
 LIST (ACL) - "a" right on the mailbox (*).

 LIST (MYRIGHTS) - any of the following rights is required to perform
  the operation: "l", "r", "i", "c", "x", "e", "a".

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

4. ACL manipulation commands

All ACL commands (i.e. ACL STORE/DELETE/SET) 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 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 than 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 or RIGHTS-INFO 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:

In the example below ACL SET command fails for 2 mailboxes "Personal/Jokes" and "Personal/Deaf and Blind". For the latter a human readable error description is returned. Also, the ACL for the mailbox "Personal/Secret" was updated to include the "r" right for a user "user=Boss" as a side effect of the ACL SET command (see also 4.3).

```
            C: A002 ACL SET (INBOX Personal/*) user=Fred rwist
            S: * ACLFAILED Personal/Jokes
            S: * ACLFAILED "Personal/Deaf and Blind" Temporary Error
            S: * LIST () "/" Personal/Secret (("ACL" (("user=Boss" "r"))))
            S: A002 OK acl set completed
```

```
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: LIST with ACL information (see 5.1)

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 LIST response with ACL information (see section 5.1) to notify the client about the changes made.

If the server is unwilling to perform the command, because some rights for the identifier are tied, it MUST return RIGHTS-INFO untagged response (see section 5.2).

```
Example:    C: A002 ACL STORE ~/Mail/saved user=smith cp
            S: * RIGHTS-INFO ~/Mail/saved user=smith la r swi cdext p
            S: A002 NO Acl store failed, some rights are tied
```

Client decides to grant both rights to the identifier:

```
            C: A003 ACL STORE ~/Mail/saved smith cdextp
            S: A003 OK Setacl complete
```

4.2. ACL DELETE

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

Data:       OPTIONAL untagged responses: LIST with ACL information (see 5.1)

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 LIST response with ACL information (see section 5.1) 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: LIST with ACL information (see 5.1)

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). LIST (ACL) "" <mailbox name>
2). For each (authentication identifier AID, access rights RD) pair returned
in the untagged LIST response that was caused by LIST (ACL), 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 LIST response with ACL information (see section 5.1) to notify the client about the changes made.

If the server is unwilling to perform the command, because some rights for an identifier from the second list parameter are tied, it MUST return RIGHTS-INFO untagged response (see section 5.2).

4.4. LIST with the ACL parameter

This document defines a new option ACL to the LIST command that requests the server to return the corresponding ACLs for all mailboxes that match the LIST mailbox name. The ACL option causes the server to return LIST with the ACL information (see 5.1).

<<List required rights here?>>

Example: C: A002 LIST (ACL) "" INBOX
S: * LIST () "/" INBOX (("ACL" (("user=Fred" "rwipslextda"))))
S: A002 OK List with acl info completed

4.5. LIST with the MYRIGHTS parameter

This document defines a new option MYRIGHTS to the LIST command that requests

the server to return the set of rights that the user has to a mailbox, that matches the LIST mailbox name. The MYRIGHTS option causes the server to return LIST with MYRIGHTS information (see 5.4).

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: A002 LIST (MYRIGHTS) "" INBOX
S: * LIST () "/" INBOX (("MYRIGHTS" "rwipsldexta"))
S: A002 OK List with acl info completed

5. ACL related responses

5.1. Extended LIST response with ACL information

Contents: name attributes
hierarchy delimiter
mailbox name
ACL information (zero or more identifier rights pairs)

This version of LIST response occurs as a result of an LIST (ACL) command.
It MAY also occur as a result of ACL STORE/DELETE/SET.
The proposed syntax conforms to the syntax of an extended LIST response as defined by mbox-list-extended ABNF element of [LISTEXT].

The meaning of "name attributes" and "hierarchy delimiter" is described in
section 7.2.2 of [IMAP4]. Hierarchy delimiter is followed by a mailbox name which this ACL applies to. This is followed by the extension part that
includes "ACL" tag followed by zero or more pairs of strings, each pair contains the identifier followed by the set of rights that the identifier has.

<<Name attributes doesn't have to be precise for the purpose of this extension? >>

Example: S: * LIST () "/" INBOX (("ACL" (("user=Fred" "rwipslextda"))))

The example above shows that a user Fred is granted "rwipslextda" rights to the mailbox "INBOX".

S: * LIST () ":" Drafts (("ACL"
(("user=Fred" "rwipslextda") ("group=Devel" "r"))))

The example above shows that a user Fred is granted "rwipslextda" rights and a member of a group "Devel" is granted "r" right to

the mailbox "Drafts".

```
S: * LIST () NIL Manson (("ACL" ()))
```

The example above shows the mailbox "Manson" has an empty ACL.

5.2. RIGHTS-INFO

```
Data:       mailbox name
            identifier
            required rights
            list of optional rights
```

The RIGHTS-INFO response occurs as a result of a failed ACL STORE or ACL SET 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:   mailbox name
            OPTIONAL response code and human readable text for failure
             reason
```

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). If the command for the mailbox fails because some rights are tied, the server MUST return RIGHTS-INFO response instead of ACLFAILED.

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

5.4. Extended LIST response with MYRIGHTS information

```
Contents:   name attributes
            hierarchy delimiter
            mailbox name
            MYRIGHTS information (list of rights)
```

This version of LIST response occurs as a result of a LIST (MYRIGHTS) command. The proposed syntax conforms to the syntax of an extended LIST response as defined by mbox-list-extended ABNF element of [LISTEXT].

The meaning of "name attributes" and "hierarchy delimiter" is described in section 7.2.2 of [IMAP4]. This is followed by the extension part that includes "MYRIGHTS" tag followed by the set of rights that the user has.

<<Should this be simplified so that we don't repeat general LIST[EXT] stuff?>>

Example: S: * LIST () "/" Sent (("MYRIGHTS" "lrwiste"))

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 MYRIGHTS response code is equivalent to the MYRIGHTS data returned in an untagged LIST 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] or [LISTEXT].

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    = "ACL" SP acl2_subcommand | list

acl2_subcommand = replaceacl | deleteacl | updateacl

acl_list_data   = "(" acl_list_tag SP acl_data ")"
                  ;; acl_list_data conforms to the syntax of
                  ;; mbox-list-extended-item from [LISTEXT]

acl_list_tag    = <"> "ACL" <">

ace             = "(" identifier SP rights ")"
```

```
   acl_data        = "(" [ace *( SP ace )] ")"
                     ;; zero or more parenthesized identifier/rights pairs

<<Currently the empty list doesn't conform to mbox-list-extended-item syntax!>>

   always_granted  = rights

   deleteacl       = "DELETE" SP mbox_or_pat SP identifier

   option          =/ "ACL" | "MYRIGHTS"
                     ;; <option> is defined in [LISTEXT]

   identifier      = astring
                      ;; UTF-8 string with syntax of ident_syntax

   ident_syntax    = ident | "-" ident

   ident           = ident_special | ident_user | ident_group |
                     ident_vendor

   ident_special   = "anyone" | "authuser" | "owner" | "administrators"

   ident_user      = "user=" ident_detail

   ident_group     = "group=" ident_detail

   ident_vendor    = "vendor=" ident_vname "." ident_detail

   ident_detail    = 1*UTF8-CHAR

   ident_vname     = 1*UTF8-CHAR
                      ;; MUST NOT contain "."

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

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

   myrights_list_data
                   = "(" "MYRIGHTS" SP rights ")"
                     ;; myrights_list_data conforms to the syntax of
                     ;; mbox-list-extended-item from [LISTEXT]

   myrights_rspcod = "MYRIGHTS" SP rights

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

   resp-text-code  =/ myrights_rspcod

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

```
updateacl       = "STORE" SP mbox_or_pat SP identifier SP mod_rights

mbox_or_pat     = mailbox / patterns

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

partialfail_rsp = "ACLFAILED" SP mailbox
                  [SP ["[" resp-text-code "]" SP] text]
                   ;; May contain optional failure reason followed by a
                   ;; human readable text

UTF8-CHAR       = CHAR | UTF8-2 | UTF8-3 | UTF8-4 | UTF8-5 | UTF8-6

CHAR            = %x01-7F

UTF8-loworder   = %x80-BF

UTF8-2          = %xC1-DF UTF8-loworder
                   ;; Disallow overlong sequences beginning with 0xC0.

UTF8-3          = (%xE0 %xA0-BF UTF8-loworder) |
                  (%xE1-EC 2UTF8-loworder) |
                  (%xED %x80-9F UTF8-loworder) |
                  (%xEE 2UTF8-loworder) |
                  (%xEF %x80-BE UTF8-loworder) |
                  (%xEF %xBF %x80-BD)
                   ;; Disallow overlong sequences beginning with 0xE0,
                   ;; disallow encoded surrogate characters, and
                   ;; disallow U+FFFE, U+FFFF.

UTF8-4          = (%xF0 %x90-BF 2UTF8-loworder) |
                  (%xF1-F7 3UTF8-loworder)
                   ;; Disallow overlong sequences beginning with 0xF0.

UTF8-5           = %xF8-FB 4UTF8-1

UTF8-6           = %xFC-FD 5UTF8-1
```

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 electronic (and optionally postal) contact information for the author of the feature.

7.2. Initial Registrations

7.2.1. 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

7.2.2. 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

8. 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 LIST (ACL) 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 (or even just "a" right) to the special identifier "anyone".

9. Other considerations

9.1. Compatibility with RFC 2086

This section gives guidelines how an existing RFC 2086 implementation may be modified to support ACL2.

1). Special handling of new identifiers. Two approaches:
 a). prohibit the creation of users on the mail system with special names (ident_special token in the ABNF) and prohibit to use "=" in identifier names. Treat "user=<identif>" the same way as "<identif>" in both ACL and ACL2 version of commands.
 b). implement translation from "user=<identif>" to "<identif>" internally.

2). "d" right mapping to "x", "t" and "e" and back. Two approaches:
 a). Tie "x", "t" and "e" together - almost no changes
 b). Implement separate "x", "t" and "e". Return "d" right in a LIST response containing ACL information when all three of "x", "t" and "e" are granted.

3). "rights" can be only sent as quoted strings, not as literals.

4). Send untagged LIST response with ACL information when server changes other ACLs on ACL STORE/DELETE/SET (servers that don't do that don't have to care).

Additional work required to implement ACL2 when RFC 2086 is already implemented:

1). Recognize new command names

2). Handle multiple mailboxes (a la LIST), return ACLFAILED on a failure.

3). Implement ACL SET

4). Optional: send MYRIGHTS untagged response on SELECT/EXAMINE

5). LISTRIGHTS command is deprecated. LISTRIGHTS response is replaced with RIGHTS-INFO response code.

6). Report "ACL2=UNION" capability.

7). Implement new special identifiers or groups if desired.

9.2. ACL2 interaction with QUOTA extension

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

GETQUOTAROOT - any of the following rights is required to perform the operation: "r", "i", "a".

<<"r" allows to calculate usage, "i" allows to put a mailbox overquota and get mailbox usage with certain implementations, that check message size before receiving the message in APPEND>>

GETQUOTA - no rights required

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

9.3. 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 3.3 ("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 3.3) 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 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.

9.4. 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.

10. 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 IS0 10646", RFC 2279, Alis Technologies, January 1998.

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

11. Informative References

[LISTEXT] Leiba, B., "IMAP4 LIST Command Extensions", work in progress, draft-ietf-imapext-list-extensions-02.txt, IBM T.J. Watson Research Center.

12. Aknowledgement

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, Steve Hole, Curtis King, Lyndon Nerenberg, Larry Greenfield, Vladimir Butenko, Dave Cridland, Harrie Hazewinkel and other participants of IMAPEXT working group.

13. Editor's Address

Alexey Melnikov

mailto: mel@isode.com

Isode Limited

14. Full Copyright Statement

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 controled 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 "Compatibility with RFC 2086" section.

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          => LIST (ACL)
  LISTRIGHTS      => ACL LIST
  MYRIGHTS        => LIST (MYRIGHTS)
```

    Changed syntax of ACL2 responses:

```
  ACL             => Extended LIST response with ACL information
  LISTRIGHTS      => RIGHTS-INFO
  MYRIGHTS        => Extended LIST response with MYRIGHTS information
```

    Any better suggestions for names are welcome.

31. Added mailbox patterns and partial failures. Updated ABNF.

32. SUBSCRIBE is NOT allowed with "r" right.

33. GETACL ("LIST (ACL)") is NOT allowed with "r" right.

34. Added human readable text to ACLFAILED untagged response.

35. GETQUOTAROOT requires any one of "r", "i" or "a".

36. Removed ACL LIST (LISTRIGHTS) command.

37. Added "vendor=" identifier prefix for vendor specific identifiers.

38. Added ABNF for identifier syntax.

39. Text and document structure change based on comments from Harrie Hazewinkel.

40. Replaced "ACLINFO" response with a special variant of LIST. Updated text and ABNF.

41. Replaced SPACE with SP in the ABNF section.

42. Added notes that group and administrators management, as well as the format of an email for posting to a mailbox is out of scope.

43. Added a table that describes all operations and right required to perform them.

44. Fixed several MYRIGHTS examples.

45. Updated ABNF to reflect recent changes ("MYRIGHTS" to "LIST (MYRIGHTS)" and "ACL GET" to "LIST (ACL)")