Network Working Group
Request for Comments: 4314
Obsoletes: 2086
Category: Standards Track

A. Melnikov
Isode Ltd.

December 2005

IMAP4 Access Control List (ACL) Extension

Status of this Memo

Copyright Notice

Abstract

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

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

Table of Contents

1. 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 RFC 2086 [RFC2086]. It tries to clarify different ambiguities in RFC 2086, in particular, the use of UTF-8 [UTF-8] in access identifiers, which rights are required for different IMAP4 commands, and how READ-WRITE/READ-ONLY response codes are related to ACL.

1.1. 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].

The phrase "ACL server" is just a shortcut for saying "IMAP server that supports ACL extension as defined in this document".

2. Access Control

The ACL extension is present in any IMAP4 implementation that 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 2.2 in the "RIGHTS=" capability.

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

Access identifier (or just "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 that is being controlled. Lowercase letters are reserved for "standard" rights, listed in Section 2.1. (Note that for compatibility with deployed clients and servers uppercase rights are not allowed.) The set of standard rights can 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 can discover the set of rights that 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. For example, an ACL may include rights to be granted to the identifier matching the user, one or more implementation-defined identifiers matching groups that 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 use only those rights granted to the most specific identifier present in the ACL. A client can 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 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.

2.1. 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, also set \SEEN during APPEND/COPY/ FETCH BODY[...])
w - write (set or clear flags other than \SEEN and \DELETED via STORE, also set them during APPEND/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/LISTRIGHTS)

2.1.1. 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 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 rwipslxcetda David lrswideta
            S: A002 OK Getacl complete
```

If the identifier specified in the LISTRIGHTS command can be granted any of the "delete" member rights on 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 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 rwipslxcetda 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 identifier specified in the LISTRIGHTS command can be granted any of the "create" member rights on 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.

2.2. Rights Defined in RFC 2086

The "RIGHTS=" capability MUST NOT include any of the rights defined in RFC 2086: "l", "r", "s", "w", "i", "p", "a", "c", "d", and the digits ("0" .. "9").

3. Access control management commands and responses

Servers, when processing a command that has an 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. Note that Section 6 recommends additional identifier's verification steps.

3.1. SETACL Command

```
Arguments:  mailbox name
            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, the server MUST NOT silently ignore unrecognized rights.

```
Example:    C: A001 GETACL INBOX/Drafts
            S: * ACL INBOX/Drafts Fred rwipslxetad Chris lrswi
            S: A001 OK Getacl complete
            C: A002 SETACL INBOX/Drafts Chris +cda
            S: A002 OK Setacl complete
            C: A003 GETACL INBOX/Drafts
            S: * ACL INBOX/Drafts Fred rwipslxetad Chris lrswicdakxet
            S: A003 OK Getacl complete


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


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

3.2. DELETEACL Command

```
Arguments:  mailbox name
            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 rwipslxetad -Fred wetd $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 wetd $team w
S: B003 OK Getacl complete
```

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

```
Example:    C: A002 GETACL INBOX
            S: * ACL INBOX Fred rwipsldexta
            S: A002 OK Getacl complete
```

3.4. LISTRIGHTS Command

```
Arguments:  mailbox name
            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 can be granted to the identifier in the ACL for the mailbox.

Some implementations MAY permit multiple forms of an 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 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
```

3.5. 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
```

3.6. 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 2.1.1 details additional server requirements related to handling of the virtual "d" and "c" rights.

3.7. 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 can be granted in the mailbox. Rights mentioned in the same string are tied together. The server MUST either grant all tied rights to the identifier in the mailbox or grant none. Section 2.1.1 details additional server requirements related to handling of the virtual "d" and "c" rights.

The same right MUST NOT be listed more than once in the LISTRIGHTS command.

3.8. 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 2.1.1 details additional server requirements related to handling of the virtual "d" and "c" rights.

4. 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	Non
commands in authenticated state												
LIST	+											
SUBSCRIBE	*											*
UNSUBSCRIBE												+
LSUB	*											*
CREATE (for parent)						+						
DELETE		?					+	?	?			
RENAME						+	+					
SELECT/EXAMINE		+										
STATUS		+										
SETACL/DELETEACL										+		
GETACL/LISTRIGHTS										+		
MYRIGHTS											+	
APPEND			?	?	+			?				
commands in selected state												
COPY			?	?	+			?				
EXPUNGE									+			
CLOSE									?			
FETCH			?									
STORE flags			?	?				?				

Note: for all commands in the selected state, the "r" is implied, because it is required to SELECT/EXAMINE a mailbox. Servers are not required to check presence of the "r" right once a mailbox is successfully selected.

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
"Non" - No rights required to perform the command

Listing and subscribing/unsubscribing mailboxes:

LIST - "l" right is required. However, unlike other commands (e.g., SELECT) the server MUST NOT return a NO response if it can't list a mailbox.
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. However, unlike other commands (e.g., SELECT) the server MUST NOT return a NO response if it can't list a subscribed mailbox.

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 the 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 are the corresponding flags 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 the expunge request, close the mailbox, and return the 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.

5. Other Considerations

5.1. Additional Requirements and Implementation Notes

5.1.1. Servers

This document defines an additional capability that is used to announce the list of extra rights (excluding the ones defined in 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 \*)] L
            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 \*)] L
            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. If such server detects that the user no longer has read access to the mailbox, it MAY send an untagged BYE response and close connection. It MAY also refuse to execute all commands specific to the selected state until the mailbox is closed; however, server implementors should note that most clients don't handle NO responses very well.

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

An ACL server implementation MUST treat received ACL modification commands as a possible ambiguity with respect to subsequent commands affected by the ACL, as described in Section 5.5 of [IMAP4]. Hence a pipeline SETACL + MYRIGHTS is an ambiguity with respect to the server, meaning that the server must execute the SETACL command to completion before the MYRIGHTS. However, clients are permitted to send such a pipeline.

5.1.2. Clients

The following requirement is put on clients in order to allow for future extensibility. 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. That is, if the client

1) can read ACLs
 and
2) can update ACLs
 but
3) doesn't allow the user to change the rights the client doesn't recognize, then it MUST preserve unrecognized rights.

Otherwise the client could risk unintentionally removing permissions it doesn't understand.

5.2. 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 can be granted flag modification rights (any of "w", "s", or "t").

Section 4 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 4) for that mailbox collectively as "shared flag rights". Note that the "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 can 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.

6. Security Considerations

An implementation MUST make sure the ACL commands themselves do not give information about mailboxes with appropriately restricted ACLs. For example, when a user agent executes a GETACL command on a mailbox that the user has no permission to LIST, the server would respond to that request with the same error that would be used if the mailbox did not exist, thus revealing no existence information, much less the mailbox's ACL.

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".

This document relies on [SASLprep] to describe steps required to perform identifier canonicalization (preparation). The preparation algorithm in SASLprep was specifically designed such that its output is canonical, and it is well-formed. However, due to an anomaly [PR29] in the specification of Unicode normalization, canonical equivalence is not guaranteed for a select few character sequences. Identifiers prepared with SASLprep can be stored and returned by an ACL server. The anomaly affects ACL manipulation and evaluation of identifiers containing the selected character sequences. These

sequences, however, do not appear in well-formed text. In order to address this problem, an ACL server MAY reject identifiers containing sequences described in [PR29] by sending the tagged BAD response. This is in addition to the requirement to reject identifiers that fail SASLprep preparation as described in Section 3.

Other security considerations described in [IMAP4] are relevant to this document. In particular, ACL information is sent in the clear over the network unless confidentiality protection is negotiated.

This can be accomplished either by the use of STARTTLS, negotiated privacy protection in the AUTHENTICATE command, or some other protection mechanism.

7. Formal Syntax

Formal syntax is defined using ABNF [ABNF], extending the ABNF rules in Section 9 of [IMAP4]. Elements not defined here can be found in [ABNF] and [IMAP4].

Except as noted otherwise, all alphabetic characters are case insensitive. The use of uppercase or lowercase 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)

capability      =/ rights-capa
                    ;;capability is defined in [IMAP4]

command-auth    =/ setacl / deleteacl / getacl /
                    listrights / myrights
                    ;;command-auth is defined in [IMAP4]

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)
```

```
   mailbox-data    =/ acl-data / listrights-data / myrights-data
                       ;;mailbox-data is defined in [IMAP4]

   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 2.2

   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
```

8. 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 has added this capability to the registry.

9. Internationalization Considerations

Section 3 states requirements on servers regarding internationalization of identifiers.

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 non-normative section gives guidelines as to 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 2.1.1 for more details). The "d" right remains for backward-compatibility, but it is a virtual right. There are two approaches for RFC 2086 server implementors to handle the "d" right and the new rights that have replaced it:

a. Tie "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 2.1.1 for more details). The "c" right remains for backwards-compatibility but it is a virtual right. Again, RFC 2086 server implementors can choose to tie rights or to implement separate rights, as described above.

Also check Sections 5.1.1 and 5.1.2, as well as Appendix A, to see other changes required. Server implementors should check which rights are required to invoke different IMAP4 commands as described in Section 4.

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.
3. Users, groups, and special identifiers (e.g., anyone) exist in the same namespace.

The work-in-progress "ACL2" extension is intended to redesign this extension to address these deficiencies without the constraint of backward-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 backward-compatible fashion, is considered worthwhile.

Appendix D. Acknowledgements

This document is a revision of 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, Lisa Dusseault, Arnt Gulbrandsen, and other participants of the IMAPEXT working group.

Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[ABNF] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[IMAP4] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

[UTF-8] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

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

[SASLprep] Zeilenga, K., "SASLprep: Stringprep Profile for User Names and Passwords", RFC 4013, February 2005.

Informative References

[RFC2086] Myers, J., "IMAP4 ACL extension", RFC 2086, January 1997.

[PR29] "Public Review Issue #29: Normalization Issue", February 2004, <http://www.unicode.org/review/pr-29.html>.

Author's Address

Alexey Melnikov
Isode Ltd.
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
GB

EMail: alexey.melnikov@isode.com

Full Copyright Statement

Intellectual Property

Acknowledgement

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