IMAPEXT Working Group
Internet Draft
Document: draft-ietf-imapext-acl-03.txt

A. Melnikov
Editor

February 2002

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). Should the document define exact syntax for identifiers starting with '$'? For example, the following was proposed by Mark Crispin:

Definable identifiers. Server implementations are *NOT* required to use any of these, but if they do, these are their semantics. Note that definable identifiers may expand to other definable identifiers.

Gxxx global identifier xxx. Valid everywhere
Uxxx per-user identifier xxx. Valid only for the logged-in user
Mxxx per-mailbox identifer xxx. Valid only for this mailbox

$xxxxx reserved for future definition

Non-definable identifiers. Server implementations SHOULD support $WORLD, and MUST support anyone and user names. However, a server can respond with a NO.

$WORLD all authorized users but not anonymous
$ANONYMOUS anonymous users
$DISABLE disabled rights

2). Do we need a mechanism to encode login names if they start with $, -?

3). Should the annotations (ANNOTATE) be controlled by "w" right?

4). Should the "l" right control UNSUBSCRIBE or should a client be able to unsubscribe a mailbox even if the client can't LIST it?

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.

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. 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 user. Identifiers starting with a dash ("-") are reserved for "negative rights", described below. Identifiers starting with a dollar sign ("$") are reserved for groups and implementation defined aliases. All other identifier strings

are interpreted in an implementation-defined manner.

Rights is a string listing a (possibly empty) set of alphanumeric characters, each character listing a set of operations which is being controlled. 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/UNSUBSCRIBE mailbox)
r - read (SELECT the mailbox, perform STATUS, CHECK, FETCH, PARTIAL, SEARCH, COPY from mailbox)
s - keep seen/unseen information across sessions (set or clear \SEEN flag via STORE or APPEND)
w - write (set or clear flags other than \SEEN and \DELETED via STORE or APPEND)
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 or APPEND)
e - perform EXPUNGE
d - 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 GETACL command. This right is defined for backward compatibility with ACL extension (RFC 2086).
a - administer (perform SETACL and DELETEACL)

Note, that moving (RENAME command) 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" ("/" is hierarchy separator) to "D/E", the user must have "x" right for mailbox "A/B/C" and "c" right for mailbox "D".

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 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.

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". For example, 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". 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 "ACL2=<rule-ID>" capability.

4. Commands

4.1. SETACL

```
Arguments:  mailbox name
            authentication identifier
            access right modification

Data:       no specific data for this command

Result:     OK - setacl completed
            NO - setacl failure: can't set acl
            BAD - command unknown or 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.

4.2. DELETEACL

```
Arguments:  mailbox name
            authentication identifier

Data:       no specific data for this command

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

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

4.3. GETACL

```
Arguments:  mailbox name

Data:       untagged responses: ACL

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

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

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

4.4. LISTRIGHTS

```
Arguments:  mailbox name
```

authentication identifier

Data: untagged responses: LISTRIGHTS

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

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

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

C: a005 LISTRIGHTS archive.imap anyone
S: * LISTRIGHTS archive.imap anyone "" l r s w i p c dtex a
0 1 2 3 4 5 6 7 8 9
S: a005 OK Listrights completed

4.5. 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.

Example: C: A003 MYRIGHTS INBOX
S: * MYRIGHTS INBOX rwipsldexa
S: A003 OK Myrights complete

5. Responses

5.1. ACL

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.

5.2. LISTRIGHTS

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

The LISTRIGHTS response occurs as a result of a LISTRIGHTS command. The first two strings are the mailbox name and identifier for which this rights list applies. Following the identifier is a string containing the (possibly empty) set of rights the identifier will always be granted in the mailbox. Following this are zero or more strings each containing a set of rights the identifier may be granted in the mailbox. Rights mentioned in the same string are tied together--either all must be granted to the identifier in the mailbox or none may be granted.

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

5.3. MYRIGHTS

```
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.

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.

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

deleteacl       = "DELETEACL" SPACE mailbox SPACE identifier
```

```
getacl          = "GETACL" SPACE mailbox

identifier      = astring
                        ;; UTF-8

listrights      = "LISTRIGHTS" SPACE mailbox SPACE identifier

listrights_data = "LISTRIGHTS" SPACE mailbox SPACE identifier
                        SPACE rights *(SPACE rights)

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

myrights        = "MYRIGHTS" SPACE mailbox

myrights_data   = "MYRIGHTS" SPACE mailbox SPACE rights

resp-text-code  =/ myrights_data

rights          = astring

setacl          = "SETACL" SPACE mailbox SPACE identifier SPACE mod_rights
```

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

10. Other considerations

10.1. Compatibility with RFC 2086

Any server that prohibits a user name in LOGIN/AUTHENTICATE that starts
with "$" MUST report "ACL" capability in addition to a "ACL=..." capability.

10.2. 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.

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 IS0 10646", RFC 2279.

12. Aknowledgement

This document is a revision of the RFC 2086 written by John G. Myers.

Editor also appreciate comments received from Mark Crispin, Chris Newman, Cyrus Daboo, Curtis King, Lyndon Nerenberg and other members of IMAPEXT working group.

13. Editor's Address

Alexey Melnikov
mailto: mel@messagingdirect.com

ACI WorldWide/MessagingDirect
#900 10117 - Jasper Ave.
Edmonton, Alberta, T5J 1W8, CANADA

14. Full Copyright Statement

Appendix A. Changes since RFC 2086

1. Changed the charset of "identifier" from US-ASCII to UTF-8.

2. Specified that identifiers starting with a dollar sign ("$") 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 note about compatibility with RFC 2086

13. Added "Implementation Notes" section.

14. Updated "References" section.