IMAP Extensions Working Group R. Gellens
Internet Draft: IMAP ANNOTATE Extension C. Daboo
Document: draft-ietf-imapext-annotate-00.txt July 2000

IMAP ANNOTATE Extension

Status of this Memo

Copyright Notice

Table of Contents

1 Abstract

The ANNOTATE extension to the Internet Message Access Protocol [IMAP4] permits clients and servers to maintain "metadata" for messages stored in an IMAP4 mailbox.

2 Discussion

Public comments can be sent to the IETF IMAP Extensions mailing list, <ietf-imapext@imc.org>. To subscribe, send a message to <ietf-imapext-request@imc.org> with the word SUBSCRIBE as the body. Private comments should be sent to the authors.

3 Conventions Used in This Document

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].

Formal syntax is defined using ABNF [ABNF] as modified by [IMAP4].

In examples, "C:" and "S:" indicate lines sent by the client and server respectively. Line breaks not preceded by a "C:" or "S:" are for editorial clarity only.

4 Document Meta-Data

4.1 Open Issues

At points in this document open issues are discussed, marked by the text "<<<OPEN-ISSUE>>>". These are items which have not been finalized. Discussion and comment is requested. Please use the IETF IMAP Extensions mailing list, as described in section 2.

4.2 Change History

Changes since -00:

1. Clarified text describing attributions, entries, and attributes.
2. Changed 'modifiedsince' to 'modtime'; referenced ACAP spec.
3. Deleted 'queued' flag.
4. Expanded and explained smtp-envelope entry.
5. Restricted including ANNOTATION data in unsolicited responses until the client uses it first. (Open issue as to if needed).
6. Examples now only use valid entries and attributes.
7. Updated Security Considerations.
8. Content-Type now defaults to text/plain.
9. Open Issue: Shared vs. private annotations.
10. Open issue: Annotation Modtime untagged response or VALIDTIME FETCH data.

11. Open issue: Conditional annotation STORE.
12. ANNOTATION criterion available if both "ANNOTATE" and "SORT" in CAPABILITY command response.
13. Prohibition on annotations in lieu of base spec functionality.
14. Specified required ACL rights.
15. ANNOTATION message data item in APPEND.
16. ANNOTATION-MODTIME message data item in STATUS.
17. Replaced ATOM_CHAR with utf8-char.
18. Updated other ABNF entries.

5 Introduction and Overview

The ANNOTATE extension is present in any IMAP4 implementation which returns "ANNOTATE" as one of the supported capabilities in the

CAPABILITY command response.

The ANNOTATE extension adds a new message data item to the FETCH and STORE commands, as well as adding SEARCH and SORT keys and APPEND and STATUS modifiers.

This extension makes the following changes to the IMAP4 protocol:

a) adds a new ANNOTATION message data item for use in the FETCH command
b) adds a new ANNOTATION message data item for use in the STORE command
c) adds a new ANNOTATION search criterion for use in the SEARCH command
d) adds a new ANNOTATION sort key for use in the SORT command extension
e) adds a new ANNOTATION data item for use in the APPEND command
f) adds a new ANNOTATION-MODTIME modifier for use in the STATUS command

The data model used for the storage of annotations is based on that of the Application Configuration Access Protocol [ACAP]. Note that there is no inheritance in annotations.

Clients MUST NOT use annotations in lieu of equivalent IMAP base specification facilities. For example, use of a "seen" flag in the vendor namespace together with ".PEEK" in fetches. Such behavior would significantly reduce IMAP interoperability.

The rest of this document describes the data model and protocol changes more rigorously.

6 Data Model

6.1 Overview

The data model used in ANNOTATE is that of a uniquely named entry with a set of uniquely named attributes, each of which has a value. A single coherent unit of "metadata" for a message is stored as a single entry, made up of several attributes.

For example, a comment added to a message has an entry name of "/message/comment". This entry is composed of several attributes such as "value", "modtime", etc. which contain the properties and data of the entry.

The protocol changes to IMAP described below allow a client to access or change the values of any attributes in any entries in a message annotation, assuming it has sufficient access rights to do so (see section 8.9 for specifics).

6.2 Namespace of Entries and Attributes

Each message annotation is made up of a set of entries. Each entry has a hierarchical name in UTF-8, with each component of the name separated by a slash ("/").

Each entry is made up of a set of attributes. Each attribute has a hierarchical name in UTF-8, with each component of the name separated by a period (".").

The value of an attribute is NIL (has no value), or a string of zero or more octets.

Entry and attribute names are not permitted to contain asterisk ("*") or percent ("%") characters and MUST be valid UTF-8 strings which do not contain NUL. Invalid entry or attribute names result in a BAD response in any IMAP commands where they are used.

Use of non-visible UTF-8 characters in entry and attribute names is strongly discouraged.

This specification defines an initial set of entry and attribute names available for use in message annotations. In addition, an extension mechanism is described to allow additional names to be added for extensibility.

6.2.1 Entry Names

Entry names MUST be specified in a standards track or IESG approved experimental RFC, or fall under the vendor namespace. See section 10.1 for the registration template.

/message
: Defines the top-level of entries associated with an entire message. This entry itself does not contain any attributes.

/message/comment
: Defines a comment or note associated with an entire message.

/message/flags
: Defines the top-level of entries for client-use flags associated with an entire message. All sub-entries are maintained entirely by the client. There is no implicit change to any flag by the server.

/message/flags/redirected
/message/flags/forwarded
: Defines client-use flags for an entire message. The "value" attribute of these entries must be either "1", "0" or NIL. The 'redirected' flag indicates that a message has been handed off to someone else, by resending the message with minimal

alterations, and in such a way that a reply by the new recipient is addressed to the original author, not the user who performed the redirection. The 'forwarded' flag indicates the message was resent to another user, embedded within or attached to a new message.

/message/smtp-envelope
: Defines the top-level of entries which together describe the SMTP envelope used in delivery of the message. There are no attributes at this level. The client SHOULD NOT modify the /message/smtp-envelope entry or any sub-entries or any of their attributes, except in messages which have the DRAFT flag set.

/message/smtp-envelope/from
/message/smtp-envelope/to
/message/smtp-envelope/orcpt
/message/smtp-envelope/envid
: Contains the properties of the SMTP envelope: 'from' is the return-path of the message; 'to' is the recipient of the message. 'orcpt' and 'envid' contain the original recipient and envelope ID as specified in [SMTP-DSN].

/message/subject

Contains text supplied by the message recipient, to be used by the client instead of the original message Subject.

/message/vendor/<vendor-token>
Defines the top-level of entries associated with an entire message as created by a particular product of some vendor. These sub-entries can be used by vendors to provide client-specific attributes. The vendor-token MUST be registered with IANA.

/body/<part-specifier>
Defines the top-level of entries associated with a specific body part of a message. This entry itself does not contain any attributes. The part-specifier uses the same part specifier syntax as the BODY message data item in the FETCH command [IMAP4].

/body/<part-specifier>/comment
Defines a comment or note associated with a specific body part of a message.

/body/<part-specifier>/flags
Defines the top-level of entries associated with flag state for a specific body part of a message. All sub-entries are maintained entirely by the client. There is no implicit change to any flag by the server.

/body/<part-specifier>/flags/seen
/body/<part-specifier>/flags/answered
/body/<part-specifier>/flags/flagged

Defines flags for a specific body part of a message. The "value" attribute of these entries must be either "1", "0" or NIL.

/body/<part-specifier>/vendor/<vendor-token>
Defines the top-level of entries associated with a specific body part of a message as created by a particular product of some vendor. This entry can be used by vendors to provide client specific attributes. The vendor-token MUST be registered with IANA.

6.2.2 Attribute Names

Attribute names MUST be specified in a standards track or IESG approved experimental RFC, or fall under the vendor namespace. See section 10.1 for the registration template.

value
: The data value of the attribute.

size
: The size of the value, in octets. Set automatically by the server, read-only to clients.

modtime
: An opaque value set by the server when this entry is modified. It can be used by the client to request notification of which entries have changed relative to that of a known entry. In addition to its use in disconnected/synchronization operations, it can also be helpful in determining which entries have changed while a client is connected. (The value is intended to be used only for comparisons within a server, not as an accurate timestamp.) It is described more fully in section 3.1.1 of [ACAP].

content-type
: A MIME [MIME] content type and subtype that describes the nature of the content of the "value" attribute. If not present, a value of "text/plain; charset=utf8" is assumed.

vendor.<vendor-token>
: Defines an attribute associated with a particular product of some vendor. This attribute can be used by vendors to provide client specific attributes. The vendor-token MUST be registered with IANA.

7 Private vs. Non-Private

<<<OPEN-ISSUE>>>

Some IMAP mailboxes are private, accessible only to the owning user. Other mailboxes are not, either because the owner has set an ACL [ACL-EXT] which permits access by other users, or because it is a shared mailbox.

This raises the issue of shared versus private annotations.

If all annotations are private, it is impossible to set annotations in a shared or otherwise non-private mailbox that are visible to other users. This eliminates what could be a useful aspect of annotations in a shared environment.

If all annotations are shared, it is impossible to use annotations for private notes on messages in shared mailboxes. Also, modifying an ACL to permit access to a mailbox by other users may unintentionally expose private information.

For example, an administrator may want to set shared annotations on messages in a shared folder, which individual users may wish to supplement with additional notes.

If shared and private annotations are to coexist, we need a clear way to differentiate them. Also, it should be as easy as possible for a client to access both and not overlook either.

One suggestion is to duplicate all attributes, with 'pers.' and 'shared.' prefixes. For example, instead of there being one 'size' attribute in an entry, there would be two: 'pers.size' and 'shared.size'. Both would be returned by default to the client, to make sure the client didn't miss one.

8 IMAP Protocol Changes

8.1 ANNOTATION Message Data Item in FETCH Command

This extension adds an ANNOTATION message data item to the FETCH command. This allows clients to retrieve annotations for a range of messages in the currently selected mailbox.

ANNOTATION <entry-specifier> <attribute-specifier>
The ANNOTATION message data item, when used by the client in the FETCH command, takes an entry specifier and an attribute specifier.

Example:

```
C: a FETCH 1 (ANNOTATION ("/message/comment" "value"))
S: * 1 FETCH (ANNOTATION ("/message/comment" (("value"
                                                "My comment"))))
S: a OK Fetch complete
```


In the above example, the content of the "value" attribute for the "/message/comment" entry is requested by the client and returned by the server.

"*" and "%" wildcard characters can be used in either specifier to match one or more characters at that position, with the exception that "%" does not match the hierarchy delimiter for the specifier it

appears in (that is, "/" for an entry specifier or "." for an attribute specifier). Thus an entry specifier of "/message/%" matches entries such as "/message/comment" and "/message/subject", but not "/message/flags/redirected".

Examples:

```
    C: a FETCH 1 (ANNOTATION ("/message/*" ("value" "modtime")))
    S: * 1 FETCH (ANNOTATION
       (("/message/comment" ("value" "My comment"
                             "modtime" "20000704000001"))
        ("/message/subject" ("value" "Rhinoceroses!"
                             "modtime" "19991231235959"))
        ("/message/vendor/eudora/label" ("value" "label43"
                             "modtime" "20000705101502"))
        ("/message/vendor/eudora/personality"
                            ("value"   "Tallulah Bankhead"
                             "modtime" "20000705101558"))))
    S: a OK Fetch complete
```

In the above example, the contents of the "value" and "modtime" attributes for any entries in the "/message" hierarchy are requested by the client and returned by the server.

```
    C: a FETCH 1 (ANNOTATION ("/message/%" "value"))
    S: * 1 FETCH (ANNOTATION
       (("/message/comment" ("value" "Patch Mangler"))
        ("/message/subject"
         ("value" "Patches?  We don' need no steenkin patches!"))))
    S: a OK Fetch complete
```

In the above example, the contents of the "value" attributes for entries at the top level only of the "/message" hierarchy are requested by the client and returned by the server.

Entry and attribute specifiers can be lists of atomic specifiers, so that multiple items of each type may be returned in a single FETCH command.

Examples:

```
    C: a FETCH 1 (ANNOTATION
       (("/message/comment" "/message/subject") "value"))
    S: * 1 FETCH (ANNOTATION
       (("/message/comment" ("value" "What a chowder-head"))
        ("/message/subject" ("value" "How to crush beer cans"))))
    S: a OK Fetch complete
```


In the above example, the contents of the "value" attributes for the two entries "/message/comment" and "/message/subject" are requested

by the client and returned by the server.

8.2 ANNOTATION Message Data Item in FETCH Response

The ANNOTATION message data item in the FETCH response displays information about annotations in a message.

ANNOTATION parenthesised list
: The response consists of a list of entries, each of which has a list of attribute-value pairs.

Examples:

```
C: a FETCH 1 (ANNOTATION ("/message/comment" "value"))
S: * 1 FETCH (ANNOTATION ("/message/comment" (
                           ("value" "My comment"))))
S: a OK Fetch complete
```

In the above example, a single entry with a single attribute-value pair is returned by the server.

```
C: a FETCH 1 (ANNOTATION
   (("/message/comment" "/message/subject") "value"))
S: * 1 FETCH (ANNOTATION
   (("/message/comment" ("value" "My comment"))
    ("/message/subject" ("value" "My subject"))))
S: a OK Fetch complete
```

In the above example, two entries each with a single attribute-value pair are returned by the server.

```
C: a FETCH 1 (ANNOTATION
                 ("/message/comment" ("value" "modtime")))
S: * 1 FETCH (ANNOTATION
                 (("/message/comment"
                     ("value" "My comment"
                      "modtime" "19990203205432"))))
S: a OK Fetch complete
```

In the above example, a single entry with two attribute-value pairs is returned by the server.

Servers SHOULD send ANNOTATION message data items in unsolicited FETCH responses if the annotation is changed by a third-party, allowing servers to keep clients updated with changes to annotations by other clients. However, servers MUST NOT include ANNOTATION data in unsolicited responses until the client has used ANNOTATION data in a FETCH command. This restriction avoids sending ANNOTATION data to a client until the client has shown it is capable of handling it.

<<<OPEN-ISSUE>>>
Is this prohibition really neccesary?

8.3 ANNOTATION Message Data Item in STORE

ANNOTATION <parenthesised entry-attribute-value list>
Sets the specified list of entries by adding or replacing the specified attributes with the values provided. Clients can use NIL for values of attributes it wants to remove from entries.

The ANNOTATION message data item used with the STORE command has an implicit ".SILENT" behavior. This means the server does not generate an untagged FETCH in response to the STORE command and assumes that the client updates its own cache if the command succeeds.

Examples:

```
C: a STORE 1 ANNOTATION ("/message/comment"
                         ("value" "My new comment"))
S: a OK Store complete
```

In the above example, the entry "/message/comment" is created (if not already present) and the attribute "value" with data set to "My new comment" is created if not already present, or replaced if it exists.

```
C: a STORE 1 ANNOTATION ("/message/comment" ("value" NIL))
S: a OK Store complete
```

In the above example, the "value" attribute of the entry "/message/comment" is removed.

Multiple entries can be set in a single STORE command by listing entry-attribute-value pairs in the list.

Example:

```
C: a STORE 1 ANNOTATION ("/message/comment" ("value"
                                              "Get tix Tuesday")
                         "/message/subject" ("value"
                                              "Wots On"))
S: a OK Store complete
```

In the above example, the entries "/message/comment" and "/message/subject" are created (if not already present) and the attribute "value" is created for each entry if not already present, or replaced if they exist.

Multiple attributes can be set in a single STORE command by listing multiple attribute-value pairs in the entry list.

Example:

```
C: a STORE 1 ANNOTATION ("/message/comment"
                         ("value" "My new comment"
                          "vendor.foobar" "foo's bar"))
S: a OK Store complete
```

In the above example, the entry "/message/comment" is created (if not already present) and the attributes "value" and "vendor.foobar" are created if not already present, or replaced if they exist.

8.3.1 Conditional Annotation STORE

<<<OPEN ISSUE>>>

Should there be a STORE modifier which permits the client to detect and avoid a collision?

MODTIME <modtime>

Instructs the server to abort the STORE and return a NO response if any specified entry has a modtime attribute greater than the supplied value.

Examples:

```
C: a STORE 1 ANNOTATION MODTIME 19720612140000
   ("/message/comment" ("value" "Tune a Fish?"))
S: a NO Entry modified since specified value
```

In the above example, the client attempts a STORE into the "/message/comment" entry which is conditioned on the entry modtime being 19720612140000 or earlier.

```
C: a STORE 1 ANNOTATION MODTIME 19720612140000
   ("/message/comment" ("value" "No Comment")
    "/message/subject" ("value" "The King Is Not a Subject"))
S: a NO Entry modified since specified value
```

In the above example, the client attempts a STORE into the "/message/comment" and "/message/subject" entries which is conditioned on neither of the entry modtimes being greater than 19720612140000.

8.4 ANNOTATION Message Data Item in APPEND

ANNOTATION <parenthesised entry-attribute-value list>
Sets the specified list of entries and attributes in the resulting message.

Example:

```
C: a APPEND drafts ANNOTATION ("/message/comment"
     ("value" "Don't send until we hear from Sally")) {310}
S: + Ready for literal data
```



```
C: MIME-Version: 1.0
...
C:
S: a OK APPEND completed
```

In the above example, a comment is added to a new message appended to the mailbox. The ellipsis represents the bulk of the message.\

8.5 ANNOTATION-MODTIME Message Data Item in STATUS

ANNOTATION-MODTIME
Requests that the STATUS command results include the latest modtime of all annotation entries of all messages in the mailbox.

Example:

```
C: a STATUS cards (ANNOTATION-MODTIME MESSAGES)
S: * STATUS cards (MESSAGES 231
                   ANNOTATION-MODTIME 19991114020700)
S: a OK STATUS completed
```

8.6 ANNOTATION Criterion in SEARCH

The ANNOTATION criterion for the SEARCH command allows a client to search for a specified string in the value of an annotation entry of a message.

ANNOTATION <entry-name> <attribute-name> <value>

Messages that have annotations with entries matching <entry-name> and attributes matching <attribute-name> and the specified string <value> in their values are returned in the SEARCH results. The "*" character can be used in the entry or attribute name fields to match any content in those items. The "%" character can be used in the entry or attribute name fields to match a single level of hierarchy only.

Examples:

```
    C: a SEARCH ANNOTATION "/message/comment" "value" "IMAP4"
    S: * SEARCH 2 3 5 7 11 13 17 19 23
    S: a OK Search complete
```

In the above example, the message numbers of any messages containing the string "IMAP4" in the "value" attribute of the "/message/comment" entry are returned in the search results.

```
    C: a SEARCH ANNOTATION "*" "*" "IMAP4"
    S: * SEARCH 1 2 3 5 8 13 21 34
    S: a OK Search complete
```


In the above example, the message numbers of any messages containing the string "IMAP4" in any attribute of any entry are returned in the search results.

A special case exists when the "modtime" attribute is used as the <attribute-name> parameter in the ANNOTATION search criterion. In this case the server matches messages when the corresponding "modtime" value is greater than the value supplied in the ANNOTATION criterion. This allows a client, for example, to find out which messages contain annotations that have changed since the last time it updated its disconnected cache.

Example:

```
    C: a SEARCH ANNOTATION "*" "modtime" "1999101713283412"
    S: * SEARCH 1 3 6 10 15 21 28 36 45 55
    S: a OK Search complete
```

In the above example, the message numbers of any messages whose "modtime" attribute of any entry exceeds the value "1999101713283412" are returned in the search results.

8.7 ANNOTATION Key in SORT

The ANNOTATION criterion for the SORT command [SORT-EXT] instructs the server to return the message numbers or UIDs of a mailbox, sorted using the values of the specified annotations. The ANNOTATION criterion is available if the server returns both "ANNOTATE" and "SORT" as supported capabilities in the CAPABILITY command response.

```
    ANNOTATION <entry-name> <attribute-name>
```

Messages are sorted using the values of the <attribute-name>

attributes in the <entry-name> entries. (The charset argument determines sort order, as specified in the SORT extension description.)

Examples:

```
C: a SORT (ANNOTATION "/message/subject" "value") UTF-8 ALL
S: * SORT 2 3 4 5 1 11 10 6 7 9 8
S: a OK Sort complete
```

In the above example, the message numbers of all messages are returned, sorted according to the "value" attribute of the "/message/subject" entry.

Note that the ANNOTATION sort key must include a fully specified entry and attribute -- wildcards are not allowed.

8.8 Annotation Modtime Untagged Response

<<<OPEN-ISSUE>>>

Should there be an untagged Annotation Modtime response? This would be issued by the server during a FETCH or SEARCH command to informs the client of the latest modtime of all entries specified in the FETCH or returned in the SEARCH results. It would also be issued following a group of one or more unsolicited FETCH responses to indicate that the client has received all updates to entries which have modtime values less than or equal to the indicated modtime value.

```
* ANNOTATION-MODTIME <modtime>
```

Informs the client of the latest modtime used in immediately prior results.

Example:

```
C: a SEARCH ANNOTATION "*" "modtime" "1999101713283412"
S: * SEARCH 1 3 6 10 15 21 28 36 45 55
S: * ANNOTATION-MODTIME 20000624140000
S: a OK Search complete
```

8.9 ACL Rights

The "r" right, as specified in [ACL-EXT], is required to use annotations in any command other than STORE.

The "w" right is needed to use annotations in the STORE command.

9 Formal Syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) notation as specified in [ABNF].

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.

```
append          = "APPEND" SP mailbox [SP flag-list] [SP date-time]
                  [SP "ANNOTATION" SP att-annotate]
                  SP literal
                   ; modifies original IMAP4 APPEND command

att-annotate    = "(" entry-att *(SP entry-att) ")"

fetch-att       =/ fetch-annotate
                   ; modifies original IMAP4 fetch-att
```



```
fetch-annotate  = "ANNOTATION" SP "(" entries SP attribs ")"
fetch-ann-resp  = "ANNOTATION" SP "(" entry-att *(SP entry-att) ")"

store-att-flags  =/ att-annotate
                    ; modifies original IMAP4 STORE command

search-key       =/ search-annotate
                    ; modifies original IMAP4 search-key

search-annotate  = "ANNOTATION" SP entry-match SP attrib-match
                   SP value

sort-key         =/ sort-annotate
                    ; modifies original
                    ; draft-crispin-imapext-sort-xx.txt sort-key

sort_annotate    = "ANNOTATION" SP entry SP attrib

status           =/ "ANNOTATION-MODTIME"
                    ; modifies original IMAP4 STATUS command

entries          = entry-match /
```

```
                      "(" entry-match *(SP entry-match) ")"
attribs             = attrib-match /
                      "(" attrib-match *(SP attrib-match) ")"
entry-att           = entry SP "(" att-value *(SP att-value) ")
att-value           = attrib SP value

utf8-char           =  %x01-FF
                      ; any character, excluding NUL
atom-slash          = any utf8-char except "/"
atom-dot            = any utf8-char except "."

entry               = DQUOTE 1*atom-slash *("/" 1*atom-slash) DQUOTE
entry-match         = DQUOTE 1*entry-match-atom
                      *("/" 1*entry-match-atom) DQUOTE
entry-match-atom    = 1*(list-wildcards / atom-slash)
                      *(list-wildcards / atom-slash)

attrib              = DQUOTE 1*atom-dot *("/" 1*atom-dot) DQUOTE
attrib-match        = DQUOTE 1*attrib-match-atom
                      *("/" 1*attrib-match-atom) DQUOTE
attrib-match-atom = 1*(list-wildcards / atom-dot)
                      *(list-wildcards / atom-dot)

value               = nstring
```

10 IANA Considerations

Both entry names and attribute names MUST be specified in a standards track or IESG approved experimental RFC, or fall under the vendor namespace. Vendor names MUST be registered.

10.1 Entry and Attribute Registration Template

```
To: iana@iana.org
Subject: IMAP Annotate Registration

Please register the following IMAP Annotate item:

[] Entry        [] Attribute
[] Vendor       [] Open: RFC _______

Name: ______________________________

Description: ________________________
```

_____

_____

Contact person: _____

email: _____

11 Security Considerations

Care must be taken to ensure that annotations whose values are intended to remain private are not stored in mailboxes which are accessible to other users. This includes mailboxes owned by the user by whose ACLs permit access by others as well as any shared mailboxes.

12 References

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

[ACAP] Newman, Myers, "ACAP -- Application Configuration Access Protocol", RFC 2244, Innosoft, Netscape, November 1997.

[ACL-EXT] Myers, "IMAP4 ACL extension", RFC 2086, Carnegie Mellon, January 1997.

[IMAP4] Crispin, "Internet Message Access Protocol - Version 4rev1", RFC 2060, University of Washington, December 1996.

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

[SMTP-DSN] Moore, "SMTP Service Extension for Delivery Status Notifications", RFC 1891, University of Tennessee, January 1996.

[SORT-EXT] Crispin, "Internet Message Access Protocol -- SORT Extension", work in progress. <http://www.ietf.org/internet-drafts/draft-crispin-imapext-sort-xx.txt>

13 Acknowledgments

Many thanks to Chris Newman for his detailed comments on the first draft of this document.

14 Authors' Addresses

Cyrus Daboo
Cyrusoft International, Inc.
Suite 780, 5001 Baum Blvd.
Pittsburgh, PA 15213
U.S.A.

Phone: +1 412 605 0499
Email: daboo@cyrusoft.com

Randall Gellens
QUALCOMM Incorporated
5775 Morehouse Dr.
San Diego, CA 92121-2779
U.S.A.

Phone: +1 858 651 5115
Email: randy@qualcomm.com

15 Full Copyright Statement