Network Working Group
Internet Draft: IMAP ANNOTATEMORE Extension
Document: draft-daboo-imap-annotatemore-01.txt

C. Daboo

November 2002

IMAP ANNOTATEMORE Extension

Status of this Memo

Copyright Notice

Table of Contents

1 Abstract

The ANNOTATEMORE extension to the Internet Message Access Protocol [IMAP4] permits clients and servers to maintain "metadata" on IMAP4 servers. This document describes two "profiles" for mailbox and server metadata.

2 Conventions Used in This Document

The key words "REQUIRED", "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [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.

3 Open Issues:

1 Handling of failures in SETANNOTATION with multiple mailbox entries specified, and at least one fails but others succeed.

2 Do we want explicit access control for the /server hierarchy, beyond simply defining certain attributes as read-only in the spec?

3 Should mailbox name in entry use IMAPURL encoding or should it be pure UTF8?

4 SETANNOTATION does not currently implement conditional store behaviour. Do we want this?

5 Should LIST flags, mailbox referrals, STATUS info be attributes of the /mailbox annotations?

6 Do we want the ability to search for annotations in the /server or /mailbox hierarchies?

4 Change History

Changes from -00 to -01:

1. Multiple entry-att responses are now simply delimited by spaces in line with ANNOTATE spec. Adjusted examples to match.
2. Fixed entry-list formal syntax item to account for unsolicited parenthesised list of entries.
3. Removed setentries formal syntax item for simplicity since its only used once.
4. Removed reference to 'message annotation' in section 5.1.
5. Changed formal syntax to restrict top level entries to /server and /mailbox/{...} only. Re-arranged entry names section to account for this change.
6. Added comment and example for ANNOTATION response to allow servers to return separate responses for each entry if desired.

5 Introduction and Overview

The ANNOTATEMORE extension is present in any IMAP4 implementation which returns "ANNOTATEMORE" as one of the supported capabilities in the CAPABILITY command response.

The goal of ANNOTATEMORE is to provide a means for clients to store and retrieve "metadata" on an IMAP4 server. This draft defines "profiles" for storing metadata associated with specific mailboxes and the server as a whole.

The ANNOTATEMORE extension adds two new commands and one new untagged response to the IMAP4 base protocol.

This extension makes the following changes to the IMAP4 protocol:

a adds a new SETANNOTATION command

b adds a new GETANNOTATION command
c adds a new ANNOTATION untagged response

The data model used in ANNOTATEMORE is exactly the same as that used in the ANNOTATE [ANNOTATE] extension to IMAP4. This is modeled after that of the Application Configuration Access Protocol [ACAP] with the exception of inheritance which is not deemed necessary here.

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 ANNOTATEMORE is one of a uniquely named entry with a set of uniquely named attributes, each of which has a value. An annotation can contain multiple named entries. For example, a general comment being added to a mailbox may have an entry name of "/mailbox/{INBOX}/comment". This entry could include named attributes such as "value", "modifiedsince", "acl" etc to represent properties and data associated with 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 an annotation, assuming it has sufficient access rights to do so.

6.2 Namespace of entries and attributes

Each 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 discouraged.

This specification defines an initial set of entry and attribute names available for use with mailbox and server 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. This specification only allows two top-level entry types for servers and mailboxes.

6.2.1.1 Server Entries

/server
: Defines the top-level of entries associated with the server. This entry itself cannot be accessed directly.

/server/comment
: Defines a comment or note associated with the server.

/server/motd
: Defines a "message of the day" for the server. This entry is always read-only - clients cannot change it.

/server/admin
: Indicates a method for contacting the server administrator. This may be a URL (e.g. a mailto URL) or other contact information, such as a telephone number. This entry is always read-only - clients cannot change it.

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

6.2.1.2 Mailbox Entries

/mailbox
: Defines the top-level of entries associated with mailboxes. This entry itself cannot be accessed directly.

/mailbox/{<mailbox-name>}
: Defines the top-level of entries associated with a specific mailbox. The <mailbox-name> is the name of the mailbox encoded using the mailbox name encoding rules described in the IMAP URL standard [IMAPURL]. The mailbox name appears inside curly-braces to delimit it from the following levels of entry name hierarchy, since it is possible that the mailbox name itself contains the '/' characters used to delimit entry name hierarchical components. This entry itself cannot be accessed directly.

/mailbox/{<mailbox-name>}/comment

Defines a comment or note associated with a mailbox.

/mailbox/{<mailbox-name>}/sort
Defines the default sort criteria [SORT] to use when first displaying the mailbox contents to the user, or NIL if sorting is not required.

/mailbox/{<mailbox-name>}/thread
Defines the default thread criteria [THREAD] to use when first displaying the mailbox contents to the user, or NIL if threading is not required. If both sort and thread are not NIL, then threading should take precedence over sorting.

/mailbox/{<mailbox-name>}/check
Boolean value "true" or "false" that indicates whether this mailbox should be checked at regular intervals by the client. The interval in minutes is specified by the checkperiod attribute.

/mailbox/{<mailbox-name>}/checkperiod
Numeric value indicating a period of minutes to use for checking a mailbox that has the check attribute set to "true".

/mailbox/{<mailbox-name>}/vendor/<vendor-token>
Defines the top-level of entries associated with a specific mailbox 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.

All attribute names implicitly have a ".priv" and a ".shared" suffix which maps to private and shared versions of the entry. Searching or fetching without using either suffix includes both. The client MUST specify either a ".priv" or ".shared" suffix when storing an annotation.

value
The data value of the attribute.

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

modifiedsince
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 since a particular point in time and is useful for disconnected/synchronisation operations.

content-type
A MIME [MIME] content type and subtype that describes the nature of the content of the "value" attribute.

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 versus Shared and Access Control

As discussed in the ANNOTATE extension [ANNOTATE] there is a need to support both private and shared annotations. This document adopts the scheme used in [ANNOTATE] that adds two standard suffixes for all attributes: ".shared" and ".priv". A GETANNOTATION command which specifies neither uses both. SETANNOTATION commands MUST explicitly use .priv or .shared suffixes.

A user can only store and retrieve private annotations on a mailbox which is returned to them via a LIST or LSUB command. A user can only store and retrieve shared annotations on a mailbox that they can SELECT and which opens READ-WRITE. If a client attempts to store or retrieve a shared annotation on a READ-ONLY mailbox, the server MUST respond with a NO response.

8 IMAP Protocol Changes

8.1 GETANNOTATION Command

This extension adds the GETANNOTATION command. This allows clients to retrieve annotations.

This command is only available in authenticated state [IMAP4].

Arguments: entry-specifier
attribute-specifier

Responses: required ANNOTATION response

Result: OK - command completed
NO - command failure: can't access annotations on that mailbox
BAD - command unknown or arguments invalid

Example:

```
C: a GETANNOTATION "/server/comment" "value.priv"
S: * ANNOTATION "/server/comment" ("value.priv" "My comment")
S: a OK GETANNOTATION complete
```

In the above example, the contents of the "value" attribute for the "/server/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 (i.e. "/" for an entry specifier or "." for an attribute specifier). Thus an entry specifier of "/server/%" would match entries such as "/server/comment" and "/server/version", but not "/server/comment/note".

Examples:

```
C: a GETANNOTATION "/server/*" "value.priv"
S: * ANNOTATION "/server/comment" ("value.priv" "My comment")
                 "/server/version" ("value.priv" "1.1")
                 "/server/motd/today" ("value.priv" "Closed at 1 pm")
S: a OK GETANNOTATION complete
```

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

```
C: a GETANNOTATION "/server/%" "value"
S: * ANNOTATION "/server/comment" ("value.priv" "My comment")
                                  ("value.shared" "Your comment")
                "/server/motd" ("value.priv" "Its sunny outside!"
                               ("value.shared" "Today is a Holiday")
S: a OK GETANNOTATION complete
```

In the above example, the contents of the "value" attributes for entries at the top level of the "/server" hierarchy only, are requested by the client and returned by the server. Both the .priv and .shared values are returned, as neither were explicitly requested.

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

Examples:

```
C: a GETANNOTATION ("/server/comment" "/server/motd") "value.priv"
S: * ANNOTATION "/server/comment" ("value.priv" "My comment")
                "/server/motd" ("value.priv" "Its sunny outside!")
S: a OK GETANNOTATION complete
```

In the above example, the contents of the "value" attributes for the two entries "/server/comment" and "/server/motd" are requested by the client and returned by the server.

```
        C: a GETANNOTATION "/server/comment" ("value.priv"
"modifiedsince.priv")
        S: * ANNOTATION "/server/comment"
```

```
                        ("value.priv" "My comment"
                            "modifiedsince.priv" "19990203205432")
        S: a OK GETANNOTATION complete
```

In the above example, the contents of the "value" and "modifiedsince" attributes for the "/server/comment" entry are requested by the client and returned by the server.

8.2 SETANNOTATION command

This extension adds the SETANNOTATION command. This allows clients to set annotations.

This command is only available in authenticated state [IMAP4].

Arguments: entry
attribute-value
list of entry, attribute-values

Responses: no specific responses for this command

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

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 server MAY return an unsolicited ANNOTATION response containing the updated annotation data. Clients MUST NOT assume that an unsolicited ANNOTATION response will be sent, and MUST assume that if the command succeeds then the annotation has been changed.

Examples:

```
    C: a SETANNOTATION "/mailbox/{INBOX}/comment"
                            ("value.priv" "My new comment")
    S: a OK SETANNOTATION complete
```

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

```
    C: a SETANNOTATION "/mailbox/{INBOX}/comment" ("value.priv" NIL)
    S: a OK SETANNOTATION complete
```

In the above example, the private "value" attribute of the entry "/mailbox/{INBOX}/comment" is removed.

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

Example:

```
C: a SETANNOTATION "/mailbox/{INBOX}/comment"
                     ("value.priv" "My new comment")
                   "/mailbox/{INBOX.shared}/comment"
                     ("value.shared" "This is a shared mailbox")
S: a OK SETANNOTATION complete
```

In the above example, the entries "/mailbox/{INBOX}/comment" and "/mailbox/{INBOX.shared}/comment" are created (if not already present) and the private and shared attributes "value" are created respectively for each entry if not already present, or replaced if they previously existed.

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

Example:

```
C: a SETANNOTATION "/mailbox/{INBOX}/comment"
                     ("value.priv" "My new comment"
                      "vendor.foobar.bloop.priv" "foo's bar")
S: a OK SETANNOTATION complete
```

In the above example, the entry "/mailbox/{INBOX}/comment" is created (if not already present) and the private attributes "value" and "vendor.foobar.bloop" are created if not already present, or replaced if they previously existed.

8.3 ANNOTATION Response

The ANNOTATION response displays results of a GETANNOTATION command, or can be returned as a unsolicited response at anytime by the server in response to a change in an annotation.

Servers SHOULD send unsolicited ANNOTATION responses if an annotation is changed by a third-party, allowing servers to keep clients updated with changes to annotations by other clients. In this case, only the entries are returned in the ANNOTATION response, not the attributes and values. If the client wants to update any cached values it must explicitly retrieve those using a GETANNOTATION command.

The ANNOTATION response can contain multiple entries in a single response, but the server is free to return multiple responses for each entry or group of entries if it desires.

This response is only available in authenticated state [IMAP4].

8.3.1 ANNOTATION response with attributes and values

The response consists of a list of entries each of which has a list of attribute-value pairs.

Examples:

```
C: a GETANNOTATION "/server/comment" "value.priv"
S: * ANNOTATION "/server/comment" ("value.priv" "My comment")
S: a OK GETANNOTATION complete
```

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

```
C: a GETANNOTATION ("/server/comment" "/server/motd") "value.priv"
S: * ANNOTATION "/server/comment" ("value.priv" "My comment")
                "/server/motd" ("value.priv" "Its sunny outside!")
S: a OK GETANNOTATION complete
```

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

```
C: a GETANNOTATION ("/server/comment" "/server/motd") "value.priv"
S: * ANNOTATION "/server/comment" ("value.priv" "My comment")
S: * ANNOTATION "/server/motd" ("value.priv" "Its sunny outside!")
S: a OK GETANNOTATION complete
```

In the above example, the server returns two separate response for each of the two entries requested.

```
C: a GETANNOTATION "/server/comment" ("value.priv"
"modifiedsince.priv")
S: * ANNOTATION "/server/comment" ("value.priv" "My comment"
                    "modifiedsince.priv" "19990203205432")
S: a OK GETANNOTATION complete
```

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

8.3.2 Unsolicited ANNOTATION response without attributes and values

The response consists of a parenthesised list of entries, each of which have changed on the server.

Examples:

```
C: a NOOP
S: * ANNOTATION ("/server/comment")
S: a OK NOOP complete
```

In the above example, the server indicates that the "/server/comment" entry has been changed.

```
C: a NOOP
S: * ANNOTATION ("/server/comment" "/server/motd")
S: a OK NOOP complete
```

In the above example, the server indicates a change to two entries.

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.

```
command-auth      /= setannotation / getannotation
                     ; adds to original IMAP4 command

response-data     /= "*" SP annotate-data CRLF
                     ; adds to original IMAP4 data responses

getannotation     = "GETANNOTATION" SP entries SP attribs
                     ; new command

setannotation     = "SETANNOTATION" SP entry-att *(SP entry-att)
                     ; new command

annotate-data     = "ANNOTATION" SP entry-list
                     ; new response

entries           = entry-match / "(" entry-match *(SP entry-match) ")"
                     ; entry specifiers that can include wildcards

attribs           = attrib-match / "(" attrib-match *(SP attrib-match) ")"
                     ; attribute specifiers that can include wildcards

entry-list        = entry-att *(SP entry-att) /
                     "(" entry *(SP entry) ")"
                     ; entry attribute-value pairs list for
                     ; GETANNOTATION response, or
                     ; parenthesised entry list for unsolicited
                     ; notification of annotation changes

entry-att         = entry SP "(" att-value *(SP att-value) ")"
att-value         = attrib SP value

atom-slash        = any ATOM-CHAR except "/"
atom-dot          = any ATOM-CHAR except "."
```

```
        entry             = DQUOTE (entry-server / entry-mbox) DQUOTE
        entry-server      = "/server/" sub-entry
```

```
        entry-mbox        = "/mailbox/{" entry-mname "}/" sub-entry
        entry-mname       = string
    sub-entry         = 1*atom-slash *("/" 1*atom-slash)

    entry-match       = DQUOTE (entry-m-server / entry-m-mbox) DQUOTE
        entry-m-server    = "/server/" sub-m-entry
        entry-m-mbox      = "/mailbox/{" entry-m-mname "}/" sub-m-entry
        entry-m-mname     = list-mailbox
        sub-m-entry       = 1*entry-m-atom *("/" 1*entry-m-atom)
    entry-m-atom      = 1*(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)

    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 More Registration

Please register the following IMAP Annotate More item:

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

Name: ____________________________

Description: _____________________

_________________________________

_________________________________

Contact person: __________________

        email:  __________________
```

11 Security Considerations

There are no known security issues with this extension.

12 References

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

[ACAP] Newman, C., and Myers, J. "ACAP -- Application Configuration Access Protocol", RFC 2244, November 1997.

[ANNOTATE] Daboo, C., Gellens, R., "IMAP ANNOTATE Extension", draft-ietf-imapext-annotate-05.txt, November 2002.

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

[IMAPURL] Newman, C., "IMAP URL Scheme", RFC 2192, Innosoft, September 1997.

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

[MIME] Freed, N., and Borenstein, N. "MIME (Multipurpose Internet Mail Extensions) Part Two: Media Types", RFC 2046, November 1996.

[SORT] Crispin, M., Murchison, K., "Internet Message Access Protocol - Sort Extension", draft-ietf-imapext-sort-10.txt, University of Washington, Oceana Matrix Ltd., October 2002.

[THREAD] Crispin, M., Murchison, K., "Internet Message Access Protocol - Thread Extension", draft-ietf-imapext-thread-12.txt, University of Washington, Oceana Matrix Ltd., October 2001.

13 Acknowledgments

The ideas expressed in this document are based on the message annotation document that was co-authored by Randall Gellens.

14 Author's Address

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

Email: daboo@cyrusoft.com