

IMAP Extensions Working Group
Internet-Draft
Expires: October 18, 2004

R. Gellens
QUALCOMM Incorporated
C. Daboo
Cyrusoft International, Inc.
April 19, 2004

IMAP ANNOTATE Extension
draft-ietf-imapext-annotate-09

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC2026](#).

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 and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate 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>.

This Internet-Draft will expire on October 18, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

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.

Change History (to be removed prior to publication as an RFC)

Changes from -08 to -09:

1. Fix formatting, ID nits etc.
2. Fix subject -> altsubject in examples.
3. Added text to SELECT/EXAMINE optional parameter definition to indicate that the option could trigger a global state change or a

Internet-Draft

IMAP ANNOTATE Extension

April 2004

- mailbox specific change.
4. Changed entry/attribute names to be case-sensitive to avoid case mapping issues with utf8 text.
 5. Clarify COPY interaction to indicate that only the current user's '.priv's are copied, not the '.priv's of other users.

Changes from -07 to -08:

1. ANNOTATESIZE response changed to use "NIL" for a mailbox that does not support any type of annotations, and "0" for a mailbox that only supports read-only annotations.

Changes from -06 to -07:

1. Added text to state entry and attribute names are always case-insensitive.
2. Removed top-level entry namespace.
3. Added server accept minima for annotation size and count.
4. Added [ANNOTATE TOOBIG] & [ANNOTATE TOOMANY] response codes.
5. Added [ANNOTATESIZE <n>] response code.
6. Added comment on suggested CONDSTORE support.
7. Modified append behaviour to account for MULTIAPPEND.
8. Tweaked ABNF.

Changes from -05 to -06:

1. Split references into Normative and Informative.
2. Reworked flags to allow IMAP4 flag prefix to appear in annotation name.
3. Removed smtp-envelope annotation - a future extension can add this.
4. Changed subject to altsubject.
5. Added \$MDNSent flag and reference to document.
6. Cleaned up formal syntax to use IMAP string type for entry and attributes, with requirements on how the string is formatted.
7. Use of ACAP vendor subtree registry for vendor tokens.
8. Fixed STORE syntax.

Changes from -04 to -05:

1. Fixed examples to match formal syntax for FETCH responses where parenthesis do not appear around entry-att items.

Changes from -03 to -04:

1. Fixed attrib/attrib-match grammar to use "." instead of "/".
2. Add text for server to reject unknown <part-specifier>.

3. Do not allow empty part-specifier.
4. Store NIL to value to delete.
5. Comment on COPY interaction with ANNOTATE.
6. Added comment that IMAP flags are mapped one-to-one with their corresponding FLAGS items.

7. Added comment that the recent flag annotation is read-only.

Changes from -02 to -03:

1. Removed reference to status modtime item.
2. Added missing 'notify' and 'ret' dsn annotations for /message/smtp-envelope.
3. Added requirement to store data permanently - no 'session only' annotations.
4. Removed Access Control section. Replaced with comments on read-only/read-write mailboxes and storing private or shared annotations.
5. Removed STORE to default .priv or .shared.
6. Added section on optional select parameters.

Changes from -01 to -02:

1. Now require .priv or .shared on store operations.

Changes from -00 to -01:

1. MODTIME moved to its own draft, which this draft now depends on. Thus, Conditional Annotation STORE and related items deleted from this draft.
2. Private versus Shared Annotations: both are possible (separately addressable using ".priv" and ".shared" suffixes). There is a per-mailbox setting for the default. It is an open issue how this is viewed or changed by the client.
3. In ACLs, the "w" right is needed to updated shared state; the "s" right is needed to update private state.
4. Various clarifications and text modifications.
5. Added 'forwarded' flag for message parts.

Changes from pre-imapext to -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.

Table of Contents

1.	Introduction and Overview	5
2.	Data Model	6
2.1	Overview	6
2.2	Namespace of entries and attributes	6
2.2.1	Entry Names	7
2.2.2	Attribute Names	9
2.3	Private versus Shared and Access Control	10
3.	IMAP Protocol Changes	11
3.1	General Considerations	11
3.2	Optional parameters with the SELECT/EXAMINE commands	11
3.3	ANNOTATION Message Data Item in FETCH Command	13
3.4	ANNOTATION Message Data Item in FETCH Response	15
3.5	ANNOTATION Message Data Item in STORE	16
3.6	ANNOTATION interaction with COPY	18
3.7	ANNOTATION Message Data Item in APPEND	18
3.8	ANNOTATION Criterion in SEARCH	18
3.9	ANNOTATION Key in SORT	19
4.	Formal Syntax	20
5.	IANA Considerations	22
5.1	Entry and Attribute Registration Template	22

6.	Security Considerations	22
	Normative References	22
	Informative References	23
	Authors' Addresses	23
A.	Acknowledgments	24
	Intellectual Property and Copyright Statements	25

[1.](#) Introduction and Overview

The ANNOTATE extension is present in any IMAP4 implementation which returns "ANNOTATE" as one of the supported capabilities in the CAPABILITY 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 an APPEND modifier.

This extension makes the following changes to the IMAP4 protocol:

- a. adds a new ANNOTATION message data item for use in FETCH
- b. adds a new ANNOTATION message data item for use in STORE
- c. adds a new ANNOTATION search criterion for use in SEARCH
- d. adds a new ANNOTATION sort key for use in SORT extension
- e. adds a new ANNOTATION data item for use in APPEND
- f. adds a new requirement on the COPY command
- g. adds a extension mechanism for adding parameters to the SELECT/EXAMINE commands and defines the ANNOTATE parameter
- h. adds two new response codes to indicate store failures of

annotations.

- i. adds a new untagged response codes for the SELECT or EXAMINE commands to indicate the maximum size.

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 behaviour would significantly reduce IMAP interoperability.

If a server supports annotations, then it MUST store all annotation data permanently, i.e. there is no concept of 'session only' annotations that would correspond to the behaviour of 'session' flags as defined in the IMAP base specification. The exception to this is IMAP flags (which are accessible directly through annotations) which may be 'session only' as determined by the FLAGS and PERMANENTFLAGS responses to a SELECT or EXAMINE command.

This extension also introduces a generalised mechanism for adding parameters to the SELECT or EXAMINE commands. It is anticipated that other extensions may want to utilise this, so it is not strictly dependent on the ANNOTATE extension being present.

In order to provide optimum support for a disconnected client (one

that needs to synchronise annotations for use when offline), servers SHOULD also support the Conditional STORE [[CONDSTORE](#)] extension.

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

[2.](#) Data Model

[2.1](#) Overview

The data model used in ANNOTATE is that of a uniquely named entry which contains a set of standard attributes. 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 `"/comment"`. This entry is composed of several attributes such as `"value"`, `"size"`, 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 2.3](#) for specifics).

[2.2](#) 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 is a string of zero or more octets.

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

Entry and attribute names are case-sensitive.

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.

[2.2.1](#) Entry Names

Entry names MUST be specified in a standards track or IESG approved experimental RFC, or fall under the vendor namespace. See [Section 5.1](#) for the registration template.

/

Defines the top-level of entries associated with an entire message. This entry itself does not contain any attributes. All entries that start with a numeric character ("0" - "9") refer to an annotation on a specific body part. All other entries are for annotations on the entire message.

/comment

Defines a comment or note associated with an entire message.

/flags

Defines the top-level of entries for flags associated with an entire message. The "value" attribute of each of the entries described below must be either "1", "0" or NIL. "1" corresponds to the flag being set.

Standard [IMAP4] flags always have a '\' prefix character. Other standard flags have a '\$' prefix. The annotation names used for all flags uses the complete name for that flag, including the prefix character.

The set of standard IMAP flags annotations are:

```
/flags/\answered  
/flags/\flagged  
/flags/\deleted  
/flags/\seen  
/flags/\draft  
/flags/\recent
```

Changes to these annotations are reflected in the standard IMAP flags. The \recent attribute is read only, clients MUST NOT attempt to change it.

Note that entry names are sent as [IMAP4] string elements which requires that '\' characters be escaped if sent as a quoted string as opposed to a literal.

Note that flag and keyword names in [IMAP4] are case-insensitive, however the entry names for the corresponding annotations are case-sensitive. Thus the [IMAP4] flag and keyword names MUST be

mapped to lowercase characters before being used as entry names for annotations.

Additional standard flags are:

```
/flags/$mdnsent  
/flags/$redirected  
/flags/$forwarded
```

The '\$mdnsent' flag is used to indicate message disposition notification processing state [MDNSENT].

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.

/altsubject

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

/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, using the [ACAP] vendor subtree registry.

/<section-part>

Defines the top-level of entries associated with a specific body part of a message. This entry itself does not contain any attributes. The section-part uses the same numeric part specifier syntax as the BODY message data item in the FETCH command [IMAP4]. The server MUST return a BAD response if the client uses an incorrect part specifier (either incorrect syntax or a specifier referring to a non-existent part). The server MUST return a BAD response if the client uses an empty part specifier (which is used in [IMAP4] to represent the entire message).

/<section-part>/comment

Defines a comment or note associated with a specific body part of a message.

`/<section-part>/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.

```
    /<section-part>/flags/seen
    /<section-part>/flags/answered
    /<section-part>/flags/flagged
    /<section-part>/flags/forwarded
```

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

`/<section-part>/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.

[2.2.2](#) Attribute Names

Attribute names MUST be specified in a standards track or IESG approved experimental RFC, or fall under the vendor namespace. See [Section 5.1](#) 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

A UTF8 string representing the data value of the attribute. To delete an annotation, the client can store NIL into the value.

size

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

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, using the [ACAP] vendor subtree registry.

[2.3](#) Private versus Shared and Access Control

Some IMAP mailboxes are private, accessible only to the owning user. Other mailboxes are not, either because the owner has set an ACL [ACL] 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. An example of such use is a shared IMAP folder containing bug reports. Engineers may want to use annotations to add information to existing messages, indicate assignments, status, etc. This use requires shared annotations.

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.

There are also situations in which both shared and private annotations are useful. 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. There is also a danger in allowing a client to store an annotation without knowing if it is shared or private.

This document proposes two standard suffixes for all attributes:

".shared" and ".priv". A search, fetch, or sort which specifies neither uses both. Store operations MUST explicitly use .priv or .shared suffixes.

A user can only store and fetch private annotations on messages in any mailbox which they can SELECT or EXAMINE, including ones which only open READ-ONLY. A user can only store and fetch shared annotations on messages in any mailbox that they can SELECT and which opens READ-WRITE. If a client attempts to store or fetch a shared

annotation on a READ-ONLY mailbox, the server MUST respond with a NO response.

[3. IMAP Protocol Changes](#)

[3.1 General Considerations](#)

The server is allowed to impose limitations on the size of any one annotation or the total number of annotations for a single message. However, the server MUST accept a minimum annotation data size of at least 1024 bytes, and a minimum annotation count per message of at least 10.

The server SHOULD indicate the maximum size for an annotation value by sending an untagged "ANNOTATESIZE" response during a SELECT or EXAMINE command. Clients MUST NOT store annotation values of a size greater than the amount indicated by the server in the "ANNOTATESIZE" response.

In some cases, servers may be able to offer annotations on some mailboxes and not others, or may be able to provide only read-only annotations on some mailboxes. For mailboxes that cannot have annotations associated with them, the server MUST return an "ANNOTATESIZE" response with a value of "NIL" during the SELECT or EXAMINE command for that mailbox. Clients MUST NOT attempt to fetch or store annotations on any messages in a mailbox for which the "ANNOTATESIZE" response was "NIL". For mailboxes that can only have read-only annotations associated with them, the server MUST return an "ANNOTATESIZE" response with a value of "0" (zero) during the SELECT or EXAMINE command for that mailbox. Clients MUST NOT attempt to store annotations on any messages in a mailbox for which the "ANNOTATESIZE" response was zero.

[3.2](#) Optional parameters with the SELECT/EXAMINE commands

This extension adds the ability to include one or more parameters with the IMAP SELECT or EXAMINE commands, to turn on or off certain standard behaviour, or to add new optional behaviours required for a particular extension.

There are two possible modes of operation:

- o A global state change where a single use of the optional parameter will effect the session state from that time on, irrespective of subsequent SELECT/EXAMINE commands.
- o A per-mailbox state change that will effect the session only for the duration of the new selected state. A subsequent SELECT/

Gellens & Daboo

Expires October 18, 2004

[Page 11]

Internet-Draft

IMAP ANNOTATE Extension

April 2004

EXAMINE without the optional parameter will cancel its effect for the newly selected mailbox.

It is anticipated that other extensions may want to use this facility, so a generalised approach is given here. This facility is not dependent on the presence of the ANNOTATE extension - other extensions can use it with a server that does not implement ANNOTATE.

Optional parameters to the SELECT or EXAMINE commands are added as a parenthesised list of atoms or strings, and appear after the mailbox name in the standard SELECT or EXAMINE command. The order of individual parameters is arbitrary. Individual parameters may consist of one or more atoms or strings in a specific order. If a parameter consists of more than one atom or string, it **MUST** appear in its own parenthesised list. Any parameter not defined by extensions that the server supports **MUST** be rejected with a NO response.

Example:

```
C: a SELECT INBOX (ANNOTATE)
S: ...
S: a OK SELECT complete
```

In the above example, a single parameter is used with the SELECT command.

Example:

```
C: a EXAMINE INBOX (ANNOTATE (RESPONSES "UID Responses") CONDSTORE)
S: ...
S: a OK EXAMINE complete
```

In the above example, three parameters are used with the EXAMINE command. The second parameter consists of two items: an atom followed by a quoted string.

Example:

```
C: a SELECT INBOX (BLURDYBLOOP)
S: a NO Unknown parameter in SELECT command
```

In the above example, a parameter not supported by the server is incorrectly used.

The ANNOTATE extension defines a single optional select parameter "ANNOTATE", which is used to turn on unsolicited responses for annotations as described in [Section 3.4](#). This optional parameter results in a per-mailbox state change, i.e. it must be used in each

SELECT/EXAMINE command in order to be effective, irrespective of whether it was used in a previous SELECT/EXAMINE during the same session.

[3.3](#) 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 ("/comment" "value"))
S: * 1 FETCH (ANNOTATION ("/comment"
                        ("value.priv" "My comment"
                         "value.shared" "Group note")))

S: a OK Fetch complete

```

In the above example, the content of the "value" attribute for the "/comment" entry is requested by the client and returned by the server. Since neither ".shared" nor ".priv" was specified, both are returned.

"*" 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 "/" matches entries such as "/comment" and "/altsubject", but not "/flags/\$redirected".

Example:

```

C: a FETCH 1 (ANNOTATION ("/*" ("value.priv" "size.priv")))
S: * 1 FETCH (ANNOTATION
  ("/comment" ("value.priv" "My comment"
                "size.priv" "10")
  "/altsubject" ("value.priv" "Rhinoceroses!"
                  "size.priv" "13")
  "/vendor/foobar/label.priv"
                  ("value.priv" "label43"
                   "size.priv" "7")
  "/vendor/foobar/personality"
                  ("value.priv" "Tallulah Bankhead"

```

```
"size.priv" "17")))
```

S: a OK Fetch complete

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

Example:

```
C: a FETCH 1 (ANNOTATION ("/%" "value.shared"))
S: * 1 FETCH (ANNOTATION
  ("/comment" ("value.shared" "Patch Mangler")
  "/altsubject" ("value.shared" "Patches? We don't
  need no steenkin patches!"))))
S: a OK Fetch complete
```

In the above example, the contents of the shared "value" attributes for entries at the top level only of the "" 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.

Example:

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

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

[3.4](#) 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.

Example:

```
C: a FETCH 1 (ANNOTATION ("/comment" "value"))
S: * 1 FETCH (ANNOTATION ("/comment"
                        ("value.priv" "My comment"
                         "value.shared" NIL)))
S: a OK Fetch complete
```

In the above example, a single entry with a single attribute-value pair is returned by the server. Since the client did not specify a ".shared" or ".priv" suffix, both are returned. Only the private attribute has a value (the shared value is NIL).

Example:

```
C: a FETCH 1 (ANNOTATION
              ("/comment" "/altsubject") "value"))
S: * 1 FETCH (ANNOTATION
              ("/comment" ("value.priv" "My comment"
                           "value.shared" NIL)
                       ("/altsubject" ("value.priv" "My subject"
                                           "value.shared" NIL)))
S: a OK Fetch complete
```

In the above example, two entries each with a single attribute-value pair are returned by the server. Since the client did not specify a ".shared" or ".priv" suffix, both are returned. Only the private attributes have values; the shared attributes are NIL.

Example:

```
C: a FETCH 1 (ANNOTATION
              ("/comment" ("value" "size")))
S: * 1 FETCH (ANNOTATION
              ("/comment"
                ("value.priv" "My comment"
                 "value.shared" NIL
                 "size.priv" "10"
                 "size.shared" "0"))))
S: a OK Fetch complete
```

In the above example, a single entry with two attribute-value pairs is returned by the server. Since the client did not specify a ".shared" or ".priv" suffix, both are returned. Only the private attributes have values; the shared attributes are NIL.

Servers MUST NOT include ANNOTATION data in unsolicited responses unless the client used the ANNOTATE select parameter when it issued the last SELECT or EXAMINE command. This restriction avoids sending ANNOTATION data to a client unless the client explicitly asks for it.

Servers SHOULD send ANNOTATION message data items in unsolicited FETCH responses if an annotation entry is changed by a third-party, and the ANNOTATE select parameter was used. This allows servers to keep clients updated with changes to annotations by other clients.

[3.5](#) 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" behaviour. 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.

If the server is unable to store an annotation because the size of its value is too large, the server MUST return a tagged NO response with a "[ANNOTATE TOOBIG]" response code.

If the server is unable to store a new annotation because the maximum number of allowed annotations has already been reached, the server MUST return a tagged NO response with a "[ANNOTATE TOOMANY]" response code.

Example:

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

In the above example, the entry `"/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 exists.

Example:

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

In the above example, the shared `"value"` attribute of the entry `"/comment"` is removed by storing `NIL` into the attribute.

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 ("/comment"  
                           ("value.priv" "Get tix Tuesday")  
                           "/altsubject"  
                           ("value.priv" "Wots On"))  
S: a OK Store complete
```

In the above example, the entries `"/comment"` and `"/altsubject"` are created (if not already present) and the private 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 ("/comment"
```

```
("value.priv" "My new comment"
"vendor.foobar.priv" "foo's bar"))
```

S: a OK Store complete

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

[3.6](#) ANNOTATION interaction with COPY

The COPY command can be used to move messages from one mailbox to another on the same server. Servers that support the ANNOTATION extension MUST, for each message being copied, copy all `'.priv'` annotation data for the current user only, and all `'.shared'` annotation data along with the message to the new mailbox. The only exceptions to this are if the destination mailbox permissions are such that either the `'.priv'` or `'.shared'` annotations are not allowed, or if the destination mailbox is of a type that does not support annotations or does not support storing of annotations (a mailbox that returns a zero or `"NIL"` value for its `ANNOTATESIZE` response code). Servers MUST NOT copy `'.priv'` annotation data for users other than the current user.

[3.7](#) ANNOTATION Message Data Item in APPEND

ANNOTATION <parenthesised entry-attribute-value list>

Sets the specified list of entries and attributes in the resulting message.

The APPEND command can include annotations for the message being appended via the addition of a new append data item. The new data item can also be used with the multi-append [MULTIAPPEND] extension that allows multiple messages to be appended via a single APPEND command.

Example:

```
C: a APPEND drafts ANNOTATION ("/comment"
    ("value.priv" "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 with a private value is added to a new message appended to the mailbox. The ellipsis represents the bulk of the message.

[3.8](#) ANNOTATION Criterion in SEARCH

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

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.

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.

Example:

```
C: a SEARCH ANNOTATION "/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 shared or private "value" attribute of the "/comment" entry are returned in the search results.

Example:

```
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 (public or private) of any entry are returned in the search results.

[3.9](#) ANNOTATION Key in SORT

ANNOTATION <entry-name> <attribute-name>

The ANNOTATION criterion for the SORT command [[SORT](#)] 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.

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.)

Example:

```
C: a SORT (ANNOTATION "/altsubject" "value.shared") UTF-8 ALL
```

Gellens & Daboo

Expires October 18, 2004

[Page 19]

Internet-Draft

IMAP ANNOTATE Extension

April 2004

```
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 shared "value" attribute of the "/altsubject" entry.

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

[4.](#) 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.

```

annotate-param    = "ANNOTATE"
                    ; defines the select parameter used with
                    ; ANNOTATE extension

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

append-message    = [SP flag-list] [SP date-time]
                    [SP "ANNOTATION" SP att-annotate] SP literal
                    ; modifies [MULTIAPPEND] extension behaviour

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

att-match         = string
                    ; dot-separated attribute name
                    ; MAY contain "*" or "%" for use as wildcards

att-value         = attrib SP value

attrib            = string
                    ; dot-separated attribute name
                    ; MUST NOT contain "*" or "%"

attribs           = att-match /

```

```

                    "(" att-match *(SP att-match) ")"

entries           = entry-match /
                    "(" entry-match *(SP entry-match) ")"

entry             = string
                    ; slash-separated path to entry
                    ; MUST NOT contain "*" or "%"

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

entry-match       = string

```

	; slash-separated path to entry ; MAY contain "*" or "%" for use as wildcards
examine	=/ *(SP "(" select-param *(SP select-param) ")" ; modifies the original IMAP4 examine command to ; accept optional parameters
fetch-ann-resp	= "ANNOTATION" SP "(" entry-att *(SP entry-att) ")"
fetch-att	=/ "ANNOTATION" SP "(" entries SP attribs ")" ; modifies original IMAP4 fetch-att
resp-text-code	=/ "ANNOTATE" SP "TOOBIG" / "ANNOTATE" SP "TOOMANY" / "ANNOTATESIZE" SP (number / "NIL") ; new response codes
search-key	=/ "ANNOTATION" SP entry-match SP att-match SP value ; modifies original IMAP4 search-key
select	=/ *(SP "(" select-param *(SP select-param) ")" ; modifies the original IMAP4 select command to ; accept optional parameters
select-param	= astring / "(" astring SP astring *(SP astring) ")" ; parameters to SELECT may contain one or ; more atoms or strings - multiple items ; are always parenthesised
sort-key	=/ "ANNOTATION" SP entry SP attrib ; modifies original sort-key [SORT]
store-att-flags	=/ att-annotate ; modifies original IMAP4 STORE command

value	= nstring
-------	-----------

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.

[5.1](#) Entry and Attribute Registration Template

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

Please register the following IMAP Annotate item:

☐ Entry ☐ Attribute

Name: _____

Description: _____

Contact person: _____

email: _____

[6.](#) Security Considerations

The ANNOTATE extension does not raise any security considerations that are not present in the base [IMAP4] protocol, and these issues are discussed in [IMAP4].

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.

Normative References

[RFC1891] Moore, K., "SMTP Service Extension for Delivery Status Notifications", [RFC 1891](#), January 1996.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", [RFC 2234](#), November 1997.
- [RFC2244] Newman, C. and J. Myers, "ACAP -- Application Configuration Access Protocol", [RFC 2244](#), November 1997.
- [RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", [RFC 3501](#), March 2003.
- [RFC3502] Crispin, M., "Internet Message Access Protocol (IMAP) - MULTIAPPEND Extension", [RFC 3502](#), March 2003.
- [RFC3503] Melnikov, A., "Message Disposition Notification (MDN) profile for Internet Message Access Protocol (IMAP)", [RFC 3503](#), March 2003.
- [SORT] Crispin, M. and K. Murchison, "Internet Message Access Protocol - Sort and Thread Extension", [draft-ietf-imapext-sort-15.txt](#), April 2004.

Informative References

- [CONDSTORE] Melnikov, A. and S. Hole, "IMAP Extension for Conditional STORE operation", [draft-ietf-imapext-condstore-05.txt](#), April 2004.
- [RFC2086] Myers, J., "IMAP4 ACL extension", [RFC 2086](#), January 1997.

Authors' Addresses

Randall Gellens
QUALCOMM Incorporated
5775 Morehouse Dr.
San Diego, CA 92121-2779
US

EMail: randy@qualcomm.com

Internet-Draft

IMAP ANNOTATE Extension

April 2004

Cyrus Daboo
Cyrusoft International, Inc.
5001 Baum Blvd.
Suite 650
Pittsburgh, PA 15213
US

EMail: daboo@cyrusoft.com

[Appendix A](#). Acknowledgments

Many thanks to Chris Newman for his detailed comments on the first draft of this document, and to the participants at the ACAP working dinner in Pittsburgh.

Internet-Draft

IMAP ANNOTATE Extension

April 2004

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in [BCP-11](#). Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Copyright (C) The Internet Society (2004). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other

Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION

Gellens & Daboo

Expires October 18, 2004

[Page 25]

Internet-Draft

IMAP ANNOTATE Extension

April 2004

HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgment

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

Gellens & Daboo

Expires October 18, 2004

[Page 26]