Network Working Group
Request for Comments: 5257
Category: Experimental

C. Daboo
Apple Inc.
R. Gellens
QUALCOMM Incorporated

June 2008

Internet Message Access Protocol - ANNOTATE Extension

Status of This Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Abstract

The ANNOTATE extension to the Internet Message Access Protocol permits clients and servers to maintain "meta data" for messages, or individual message parts, stored in a mailbox on the server. For example, this can be used to attach comments and other useful information to a message. It is also possible to attach annotations to specific parts of a message, so that, for example, they could be marked as seen, or important, or a comment added.

Note that this document was the product of a WG that had good consensus on how to approach the problem. Nevertheless, the WG felt it did not have enough information on implementation and deployment hurdles to meet all of the requirements of a Proposed Standard. The IETF solicits implementations and implementation reports in order to make further progress.

Implementers should be aware that this specification may change in an incompatible manner when going to Proposed Standard status. However, any incompatible changes will result in a new capability name being used to prevent problems with any deployments of the experimental extension.

Table of Contents

1. Introduction and Overview

The ANNOTATE extension is present in any IMAP [RFC3501] implementation that returns "ANNOTATE-EXPERIMENT-1" as one of the supported capabilities in the CAPABILITY response.

This extension makes the following changes to the IMAP 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 the 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 new ANNOTATE parameter for use with the SELECT/EXAMINE commands.

h. adds two new response codes to indicate store failures of annotations.

i. adds a new untagged response code for the SELECT or EXAMINE commands to indicate the maximum sized annotation that can be stored.

j. adds a new Access Control List (ACL) "bit" for use with the ACL extensions [RFC2086] and [RFC4314].

The data model used for the storage of annotations is based on the Application Configuration Access Protocol [RFC2244]. Note that there is no inheritance in annotations.

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 behavior of "session" flags as defined in the IMAP base specification.

In order to provide optimum support for a disconnected client (one that needs to synchronize annotations for use when offline), servers SHOULD also support the Conditional STORE [RFC4551] extension.

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

2. Conventions Used in This Document

The examples in this document use "C:" and "S:" to indicate lines sent by the client and server, respectively.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Data Model

3.1. Overview

The data model for annotations in ANNOTATE uses a uniquely named entry that contains a set of standard attributes. Thus, a single coherent unit of "meta data" for a message is stored as a single entry, made up of several attributes.

For example, a comment annotation added to a message has an entry name of "/comment". This entry is composed of several attributes such as "value", "size", etc., that 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 attribute in any entry in a message annotation, assuming it has sufficient access rights to do so (see Section 3.4 for specifics).

3.2. Namespace of Entries and Attributes

A message may contain zero or more annotations, each of which is a single uniquely named entry. Each entry has a hierarchical name, with each component of the name separated by a slash ("/"). An entry name MUST NOT contain two consecutive "/" characters and MUST NOT end with a "/" character.

Each entry is made up of a set of attributes. Each attribute has a hierarchical name, with each component of the name separated by a period ("."). An attribute name MUST NOT contain two consecutive "." characters and MUST NOT end with a "." character.

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 NOT contain non-ASCII characters or the NULL octet. Invalid entry or attribute names result in a BAD response in any IMAP commands where they are used.

Attribute names MUST NOT contain any hierarchical components with the names "priv" or "shared", as those have special meaning (see Section 3.3).

Entry and attribute names are case-sensitive.

Use of control or punctuation 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 as needed.

3.2.1. Entry Names

Entry names MUST be specified in a standards track or IESG approved experimental RFC, or fall under the vendor namespace. See Section 6.1 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
: This entry hierarchy is reserved for future use.

/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 annotations. The vendor-token MUST be registered with IANA, using the [RFC2244] 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 is a numeric part specifier. Its

syntax is the same as the section-part ABNF element defined in [RFC3501]. 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 IMAP 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 the 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
This is similar to the IMAP \Seen flag, except it applies to only the body part referenced by the entry.

/<section-part>/flags/answered
This is similar to the IMAP \Answered flag, except it applies to only the body part referenced by the entry.

/<section-part>/flags/flagged
This is similar to the IMAP \Flagged flag, except it applies to only the body part referenced by the entry.

/<section-part>/flags/forwarded
This is similar to the IMAP $Forwarded keyword, except it applies to only the body part referenced by the entry.

Defines flags for a specific body part of a message. The "value" attribute of each of the entries described above must be either "1", "0", or "NIL". "1" corresponds to the flag being set.

/<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 annotations. The vendor-token MUST be registered with IANA.

3.2.2. Attribute Names

Attribute names MUST be specified in a standards track or IESG approved experimental RFC. See Section 6.1 for the registration template.

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

value
: A string or binary data representing the value of the annotation. To delete an annotation, the client can store "NIL" into the value. If the client requests the value attribute for a non-existent entry, then the server MUST return "NIL" for the value. The content represented by the string is determined by the content-type used to register the entry (see Section 6.1 for entry registration templates). Where applicable, the registered content-type MUST include a charset parameter. Text values SHOULD use the utf-8 [RFC3629] character set. Note that binary data (data which may contain the NULL octet) is allowed (e.g., for storing images), and this extension uses the "literal8" syntax element [RFC4466] to allow such data to be written to or read from the server.

size
: The size of the value, in octets. Set automatically by the server, read-only to clients. If the client requests the size attribute for a non-existent entry, then the server MUST return "0" (zero) for the size.

3.3. Private Versus Shared

Some IMAP mailboxes are private, accessible only to the owning user. Other mailboxes are not, either because the owner has set an ACL [RFC4314] that 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 then impossible to have annotations in a shared or otherwise non-private mailbox be 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 or FETCH command that specifies neither, uses both. STORE, APPEND, and SORT commands MUST explicitly use ".priv" or ".shared" suffixes.

If the ANNOTATE extension is present, support for shared annotations in servers is REQUIRED, while support for private annotations in servers is OPTIONAL. This recognizes the fact that support for private annotations may introduce a significant increase in complexity to a server in terms of tracking ownership of the annotations, how quota is determined for users based on their own annotations, etc. Clients that support the ANNOTATE extension MUST handle both shared and private annotations.

3.4. Access Control

A user needs to have appropriate rights in order to read or write ".priv" or ".shared" annotation values. How those rights are calculated depends on whether or not the ACL [RFC2086] extension or its update [RFC4314] is present. If a client attempts to store or fetch an annotation to which they do not have the appropriate rights, the server MUST respond with a NO response.

When the ACL extension is not present, access to annotation values is governed by the nature of the selected state, in particular whether the mailbox was SELECTED or EXAMINED in READ-ONLY or READ-WRITE mode.

When the ACL extension is present, the server MUST recognize the new ACL "n" right, in addition to the ones defined by the ACL extension itself.

For ".priv" annotation values, the "r" right controls both read and write access. When it is on, access to ".priv" annotations is allowed; when it is off, access to ".priv" annotations is disallowed.

For ".shared" annotation values, the "r" right controls read access. When it is on, ".shared" annotations can be read; when it is off, ".shared" annotations cannot be read.

For ".shared" annotation values, the "n" right controls write access. When it is on, ".shared" annotations can be changed or created through either a STORE or APPEND command; when it is off, ".shared" annotations cannot be changed or created. The "n" right constitutes a "shared flag right" as defined in Section 6.2 of [RFC4314].

A summary of all the access control restrictions is tabulated below

Server Type	Action on annotation	Type of mailbox
Server without ACL Extension	read .priv values	Any mailbox that can be SELECTED or EXAMINED.
	write .priv values	Any SELECTED [READ-WRITE] mailbox. SELECTED [READ-ONLY] mailboxes MAY also permit writes.
	read .shared values	Any mailbox that can be SELECTED or EXAMINED.
	write .shared values	Any mailbox that can be SELECTED or EXAMINED and is [READ-WRITE].
Server with ACL Extension	read .priv values	Any mailbox with the "r" ACL right.
	write .priv values	Any mailbox with the "r" ACL right.
	read .shared values	Any mailbox with the "r" ACL right.
	write .shared values	Any mailbox with the "n" ACL right.

3.5. Access to Standard IMAP Flags and Keywords

Due to the ambiguity of how private and shared values would map to the base IMAP flag and keyword values, the ANNOTATE extension does not expose IMAP flags or keywords as entries. However, the /flags annotation entry is reserved for future use and MUST NOT be used by clients or servers supporting this extension.

Clients that need to implement shared and private "flags" can create their own annotation entries for those, completely bypassing the base IMAP flag/keyword behavior.

4. IMAP Protocol Changes

4.1. General Considerations

Servers may be able to offer only a limited level of support for annotations in mailboxes, and it is useful for clients to be able to know what level of support is available. Servers MUST return an ANNOTATIONS response code during the SELECT or EXAMINE command for a mailbox to indicate the level of support. Possible data items used with the ANNOTATIONS response code are:

"NONE" - this indicates that the mailbox being selected does not support annotations at all. Clients MUST NOT attempt to use annotation extensions in commands for this mailbox.

"READ-ONLY" - this indicates that the annotations supported by the mailbox cannot be changed by the client. Clients MUST NOT attempt to store annotations on any messages in a mailbox with this response code.

"NOPRIVATE" - this indicates that the server does not support private annotations on the mailbox. Only shared annotations are supported. Clients SHOULD only attempt to read or store annotations attributes with the ".shared" suffix. If a client uses an attribute with the ".priv" suffix in a FETCH command, then servers should return the attribute value in the FETCH response as "NIL". If a client uses an attribute with the ".priv" suffix in a STORE command (or an APPEND command targeted at the mailbox), then the server MUST return a NO response.

numeric values - if servers support writable annotations, then the server MUST indicate the maximum size in octets for an annotation value by providing the maximum size value in the response code. Clients MUST NOT store annotation values of a size greater than the amount indicated by the server. Servers MUST accept a minimum

annotation data size of at least 1024 octets if annotations can be written.

In addition, the server MAY limit the total number of annotations for a single message. However, the server MUST provide a minimum annotation count per message of at least 10.

4.2. ANNOTATE Parameter with the SELECT/EXAMINE Commands

The ANNOTATE extension defines a single optional SELECT parameter [RFC4466] "ANNOTATE", which is used to turn on unsolicited responses for annotations as described in Section 4.4. 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.

Example:

```
C: a SELECT INBOX (ANNOTATE)
S: * FLAGS (\Answered \Flagged \Draft \Deleted \Seen)
S: * OK [PERMANENTFLAGS (\Answered \Flagged \Draft
                                    \Deleted \Seen \*)]
S: * 10268 EXISTS
S: * 1 RECENT
S: * OK [UNSEEN 10268]
S: * OK [UIDVALIDITY 890061587]
S: * OK [UIDNEXT 34643]
S: * OK [ANNOTATIONS 20480 NOPRIVATE]
S: a OK [READ-WRITE] Completed
```

In the above example, a SELECT command with the ANNOTATE parameter is issued. The response from the server includes the required ANNOTATIONS response that indicates that the server supports annotations up to a maximum size of 20480 octets, and does not support private annotations (only shared).

4.3. 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 "%" wild card characters can be used in entry specifiers to match one or more characters at that position, with the exception that "%" does not match the "/" hierarchy delimiter. Thus, an entry specifier of "/%" matches entries such as "/comment" and "/altsubject", but not "/1/comment".

Example:

```
C: a UID FETCH 1123 (UID ANNOTATION
                     (/* (value.priv size.priv)))
S: * 12 FETCH (UID 1123 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.

4.4. ANNOTATION Message Data Item in FETCH Response

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

ANNOTATION parenthesized list

The response consists of a list of entries, each of which have 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 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.

Unsolicited ANNOTATION responses MUST NOT include ANNOTATION data values -- only the entry name of the ANNOTATION that has changed. This restriction avoids sending ANNOTATION data values (which may be large) to a client unless the client explicitly asks for the value.

Example:

```
C: a STORE 1 +FLAGS (\Seen)
S: * 1 FETCH (FLAGS (\Seen))
              ANNOTATION (/comment))
S: a OK STORE complete
```

In the above example, an unsolicited ANNOTATION response is returned during a STORE command. The unsolicited response contains only the entry name of the annotation that changed, and not its value.

4.5. ANNOTATION Message Data Item in STORE

ANNOTATION <parenthesized 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. Though note, that if the Conditional STORE extension [RFC4551] is present, then an untagged FETCH response with a MODSEQ data item will be returned by the server as required by [RFC4551].

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). Its private attribute "value" is created if not already present, or replaced if it exists. "value.priv" is set to "My new comment".

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 or replaced for each entry.

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"
                          value.shared "foo's bar"))
S: a OK Store complete
```

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

4.6. 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 "NONE" or "READ-ONLY" response code in its ANNOTATIONS response), or if the destination mailbox cannot support the size of an annotation because it exceeds the ANNOTATIONS value. Servers MUST NOT copy ".priv" annotation data for users other than the current user.

4.7. ANNOTATION Message Data Item in APPEND

ANNOTATION <parenthesized 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 [RFC4466]. The new data item can also be used with the multi-append [RFC3502] 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 I say so")) {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.

4.8. 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>, 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 name field to match any content in those items. The "%" character can be used in the entry name field to match a single level of hierarchy only.

Only the "value", "value.priv", and "value.shared" attributes can be searched. Clients MUST NOT specify an attribute other than either "value", "value.priv", or "value.shared". Servers MUST return a BAD response if the client tries to search any other attribute.

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 * value.priv "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 the private "value" attribute of any entry are returned in the search results.

4.9. ANNOTATION Key in SORT

ANNOTATION <entry-name> <attribute-name>

The ANNOTATION criterion for the SORT command [RFC5256] instructs the server to return the sequence numbers or Unique Identifiers (UIDs) of messages in a mailbox, sorted using the values of the specified annotations. The ANNOTATION criterion is available if the server returns both ANNOTATE-EXPERIMENT-1 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.

Clients MUST provide either the ".priv" or ".shared" suffix to the attribute name to ensure that the server knows which specific value to sort on.

Only the "value.priv" and "value.shared" attributes can be used for sorting. Clients MUST NOT specify an attribute other than either "value.priv" or "value.shared". Servers MUST return a BAD response if the client tries to sort on any other attribute.

When either "value.priv" or "value.shared" is being sorted, the server MUST use the character set value specified in the SORT command to determine the appropriate sort order.

Example:

```
C: a SORT (ANNOTATION /altsubject value.shared) 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 shared "value" attribute of the "/altsubject" entry.

Note that the ANNOTATION sort key must include a fully specified entry -- wild cards are not allowed.

4.10. New ACL Rights

As discussed in Section 3.4, this extension adds a new "n" right to the list of rights provided by the ACL extensions [RFC2086] and [RFC4314].

5. Formal Syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) notation as specified in [RFC5234].

Non-terminals referenced but not defined below are as defined by [RFC3501] with the new definitions in [RFC4466] superseding those in [RFC3501].

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.

```
   ann-size          = "NONE" /
                       (("READ-ONLY" / nz-number)
                        [SP "NOPRIVATE"])
                       ; response codes indicating the level of
                       ; support for annotations in a mailbox

   append-ext        =/ att-annotate
                       ; modifies [RFC3501] extension behaviour

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

   att-search        = "value" / "value.priv" / "value.shared"
                       ; the only attributes that can be searched

   att-sort          = "value.priv" / "value.shared"
                       ; the only attributes that can be sorted

   att-value         = attrib SP value

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

```
      attribs           = attrib / "(" attrib *(SP attrib) ")"
                          ; one or more attribute specifiers

      capability        =/ "ANNOTATE-EXPERIMENT-1"
                          ; defines the capability for this extension

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

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

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

      entry-match       = list-mailbox
                          ; slash-separated path to entry
                          ; MAY contain "*" or "%" for use as wild cards

      fetch-att         =/ "ANNOTATION" SP "(" entries SP attribs ")"
                          ; modifies original IMAP fetch-att

      msg-att-dynamic   =/ "ANNOTATION" SP
                             ( "(" entry-att *(SP entry-att) ")" /
                               "(" entry *(SP entry) ")" )
                          ; extends FETCH response with annotation data

      resp-text-code    =/ "ANNOTATE" SP "TOOBIG" /
                           "ANNOTATE" SP "TOOMANY" /
                           "ANNOTATIONS" SP ann-size
                          ; new response codes

      search-key        =/ "ANNOTATION" SP entry-match SP att-search
                           SP value
                          ; modifies original IMAP search-key

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

      sort-key          =/ "ANNOTATION" SP entry SP att-sort
                          ; modifies original sort-key

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

      value             = nstring / literal8
```

6. IANA Considerations

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

Attribute names MUST be specified in a standards track or IESG approved experimental RFC.

Each entry registration MUST include a content-type that is used to indicate the nature of the annotation value. Where applicable, a charset parameter MUST be included with the content-type.

6.1. Entry and Attribute Registration Template

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

Please register the following IMAP Annotate item:

[] Entry [] Attribute

Name: _____

Description: _____

_____

_____

Content-Type:_____

Contact person: _____

email: _____

6.2. Entry Registrations

The following templates specify the IANA registrations of annotation entries specified in this document.

6.2.1. /comment

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

Please register the following IMAP Annotate item:

[X] Entry [] Attribute

Name: /comment

Description: Defined in IMAP ANNOTATE extension document.

Content-Type: text/plain; charset=utf-8

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.2.2. /flags

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

Please register the following IMAP Annotate item:

[X] Entry [] Attribute

Name: /flags

Description: Reserved entry hierarchy.

Content-Type: -

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.2.3. /altsubject

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

Please register the following IMAP Annotate item:

[X] Entry [] Attribute

Name: /altsubject

Description: Defined in IMAP ANNOTATE extension document.

Content-Type: text/plain; charset=utf-8

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.2.4. /<section-part>/comment

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

Please register the following IMAP Annotate item:

[X] Entry [] Attribute

Name: /<section-part>/comment

Description: Defined in IMAP ANNOTATE extension document.

Content-Type: text/plain; charset=utf-8

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.2.5. /<section-part>/flags/seen

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

Please register the following IMAP Annotate item:

[X] Entry [] Attribute

Name: /<section-part>/flags/seen

Description: Defined in IMAP ANNOTATE extension document.

Content-Type: text/plain; charset=utf-8

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.2.6. /<section-part>/flags/answered

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

Please register the following IMAP Annotate item:

[X] Entry [] Attribute

Name: /<section-part>/flags/answered

Description: Defined in IMAP ANNOTATE extension document.

Content-Type: text/plain; charset=utf-8

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.2.7. /<section-part>/flags/flagged

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

Please register the following IMAP Annotate item:

[X] Entry [] Attribute

Name: /<section-part>/flags/flagged

Description: Defined in IMAP ANNOTATE extension document.

Content-Type: text/plain; charset=utf-8

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.2.8. /<section-part>/flags/forwarded

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

Please register the following IMAP Annotate item:

[X] Entry [] Attribute

Name: /<section-part>/flags/forwarded

Description: Defined in IMAP ANNOTATE extension document.

Content-Type: text/plain; charset=utf-8

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.3. Attribute Registrations

The following templates specify the IANA registrations of annotation attributes specified in this document.

6.3.1. value

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

Please register the following IMAP Annotate item:

[] Entry [X] Attribute

Name: value

Description: Defined in IMAP ANNOTATE extension document.

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.3.2. size

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

Please register the following IMAP Annotate item:

[] Entry [X] Attribute

Name: size

Description: Defined in IMAP ANNOTATE extension document.

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.4. Capability Registration

This document registers "ANNOTATE-EXPERIMENT-1" as an IMAPEXT capability.

7. Internationalization Considerations

Annotations may contain values that include text strings, and both searching and sorting are possible with annotations. Servers MUST follow standard IMAP text normalization, character set conversion, and collation rules when such operations are carried out, as would be done for other textual fields being searched or sorted on.

8. Security Considerations

Annotations whose values are intended to remain private MUST be stored in ".priv" values instead of ".shared" values, which may be accessible to other users.

Excluding the above issues, the ANNOTATE extension does not raise any security considerations that are not present in the base IMAP protocol; these issues are discussed in [RFC3501].

9. References

9.1. Normative References

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 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.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4314] Melnikov, A., "IMAP4 Access Control List (ACL) Extension", RFC 4314, December 2005.

[RFC4466] Melnikov, A. and C. Daboo, "Collected Extensions to IMAP4 ABNF", RFC 4466, April 2006.

[RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5256] Crispin, M. and K. Murchison, "Internet Message Access Protocol - SORT and THREAD Extensions", RFC 5256, June 2008.

9.2. Informative References

[RFC4551] Melnikov, A. and S. Hole, "IMAP Extension for Conditional STORE Operation or Quick Flag Changes Resynchronization", RFC 4551, June 2006.

10. 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. The participants of the IMAPext working group made significant contributions to this work.

Authors' Addresses

Cyrus Daboo
Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
USA

EMail: cyrus@daboo.name
URI: http://www.apple.com/

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

EMail: randy@qualcomm.com

Full Copyright Statement

Intellectual Property