Network Working Group
Internet-Draft
Expires: September 30, 2006

C. Daboo

March 29, 2006

IMAP METADATA Extension
draft-daboo-imap-annotatemore-09

Status of this Memo

Copyright Notice

Abstract

The METADATA extension to the Internet Message Access Protocol permits clients and servers to maintain "annotations" or "meta data" on IMAP servers. It is possible to have annotations on a per-mailbox basis or on the server as a whole. For example, this would allow comments about the purpose of a particular mailbox to be "attached" to that mailbox, or a "message of the day" containing server status information to be made available to anyone logging in to the server.

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

Changes from -08 to -09:
1. Remove content-language attribute and reference.
2. Changed capability and command names.
3. Revised abstract.

Changes from -07 to -08:
1. Changed 'string' formal syntax to 'list-mailbox' and 'astring' for entry/attribute names.
2. Updated examples to match new astring syntax.
3. Changed CAPABILITY name due to incompatible syntax change.
4. Removed content-type attribute.
5. Added Content-type to IANA registration for entries.
6. Removed vendor attributes.
7. Fixed examples in section 3.3 for multi-mailbox and multi-entry cases.
8. Removed wildcards for attributes.
9. Entry/attributes can now only be ASCII.
10. Tied up text wrt storing/fetching.
11. Added Conventions section
12. Entry/attributes MUST NOT contain consecutive or trailing '/' or '.'.
13. Changed to use IMAP ABNF extensions document for some formal syntax items.

Changes from -06 to -07:
1. Reworded /checkperiod item.
2. Clarified unsolicited response behaviour.

Changes from -05 to -06:
1. Removed 'modifiedsince' attribute as there is currently no use for it.
2. Added content-language attribute.
3. Changed access to allow .priv and .shared on any mailbox returned by LIST/LSUB.
4. Added IANA registrations for items defined in this document.
5. Added latest IPR statement.
6. Updated references.

Changes from -04 to -05:
1. Fix for valid IMAP state of commands.
2. Fix formatting, ID nits etc.

Changes from -03 to -04:
1. Allow retrieval of shared annotations for READ-ONLY mailbox.
2. Clarification of annotation loss on implicit removal of \Noselect mailboxes.

3. Now requires roll-back of all changes to matching mailboxes if there is a partial failure in SETANNOTATION.

Changes from -02 to -03:

1. Reworked entry naming scheme to split out mailbox name and use empty string for server items.

Changes from -01 to -02:

1. SETANNOTATION lists use (..).
2. Explicitly state behaviour of unsolicited responses.
3. Adding SHOULD behaviour for rename/delete of mailboxes.
4. Added statement about supporting annotations on \Noselect mailboxes.
5. Cleaned up formal syntax to use IMAP string type for entry and attributes, with requirements on how the string is formatted.
6. Use of ACAP vendor subtree registry for vendor tokens.

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.

Table of Contents

1. Introduction and Overview

The METADATA extension is present in any IMAP [RFC3501] implementation which returns "METADATA" as one of the supported capabilities in the CAPABILITY command response. The "LISTEXT" [I-D.ietf-imapext-list-extensions] extension MUST also be present.

The goal of METADATA is to provide a means for clients to set and retrieve "annotations" or "meta data" on an IMAP server. The annotations can be associated with specific mailboxes or the server as a whole.

The METADATA extension adds two new commands and one new untagged response to the IMAP base protocol. It adds one new LISTEXT "selection" [I-D.ietf-imapext-list-extensions] option type and one new LISTEXT "return" [I-D.ietf-imapext-list-extensions] option type.

This extension makes the following changes to the IMAP protocol:

- adds a new SETMETADATA command
- adds a new GETMETADATA command
- adds a new METADATA untagged response
- adds a new METADATA response code
- adds a new METADATA LISTEXT selection option type
- adds a new METADATA LISTEXT return option typer

The data model used in METADATA is exactly the same as that used in the ANNOTATE [I-D.ietf-imapext-annotate] extension to IMAP. This is modeled after that of the Application Configuration Access Protocol [RFC2244] 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.

2. Conventions Used in This Document

In examples, "C:" and "S:" indicate lines sent by the client and server respectively.

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

3. Data Model

3.1. Overview

The data model used in METADATA is one of a uniquely named entry with a set of uniquely named attributes, each of which has a value. Annotations can be added to mailboxes when a mailbox name is provided as the first argument to the new commands, or to the server as a whole when the empty string is provided as the first argument to the new commands.

An annotation can contain multiple named entries. For example, a general comment being added to a mailbox may have an entry name of "/comment". This entry could include named attributes such as "value" and "size" 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.

3.2. Namespace of entries and attributes

Each annotation is made up of a set of entries. 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 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 with mailbox and server annotations. In addition an extension mechanism is described to allow additional names to be added for extensibility.

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.

3.2.1.1. Server Entries

These entries are used when the mailbox name argument to the new SETMATADATA command is the empty string or with the new GETMETADATA command.

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

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

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

/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 annotations. The vendor-token MUST be registered with IANA, using the ACAP [RFC2244] vendor subtree registry.

3.2.1.2. Mailbox Entries

These entries are used when the mailbox name argument to the new SETMETADATA command is not the empty string, or with the new LISTEXT selection option type.

/comment
: Defines a comment or note associated with a mailbox.

/sort

Defines the default sort criteria [I-D.ietf-imapext-sort] to use when first displaying the mailbox contents to the user, or NIL if sorting is not required.

/thread
Defines the default thread criteria [I-D.ietf-imapext-sort] 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.

/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 entry.

/checkperiod
Numeric value indicating a period of minutes that the client uses to determine the interval of regular 'new mail' checks for the corresponding mailbox.

/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 annotations. The vendor-token MUST be registered with IANA, using the ACAP [RFC2244] vendor subtree registry.

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 which maps to private and shared versions of the entry. Retrieving an annotation without using either suffix includes both. The client MUST specify either a ".priv" or ".shared" suffix when setting an annotation.

value
A string or binary data representing the value of the annotation. To delete an annotation, the client can store "NIL" into 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 etc), and this extension uses the "literal8" syntax element [I-D.melnikov-imap-ext-abnf] 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.

3.3. Private versus Shared and Access Control

As discussed in the ANNOTATE [I-D.ietf-imapext-annotate] extension there is a need to support both private and shared annotations. This document adopts the scheme used in [I-D.ietf-imapext-annotate] that adds two standard suffixes for all attributes: ".shared" and ".priv". A GETMETADATA or extended LIST command which specifies neither uses both. SETMETADATA commands MUST explicitly use .priv or .shared suffixes.

A user can only set and retrieve private or shared annotations on a mailbox which is returned to them via a LIST or LSUB command, irrespective of whether they have read or write access to the actual message content of the mailbox. If the client attempts to set or retrieve annotations on other mailboxes, the server MUST respond with a NO response.

If the METADATA extension is present, support for shared annotations is REQUIRED, whilst support for private annotations is OPTIONAL. This recognises the fact that support for private annotations may introduce significantly more 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 METADATA extension MUST handle both shared and private annotations.

4. IMAP Protocol Changes

4.1. General Considerations

The new SETMETADATA command and the METADATA response each have a mailbox name argument, indicating that the annotations being referred to are attached to the specified mailbox. An empty string can be used for the mailbox name to signify server annotations.

Both "*" and "%" list wildcard characters MAY be used in the mailbox name argument in the SETMETADATA command to match all possible occurrences of a mailbox name pattern. However, "*" or "%" by themselves MUST NOT match the empty string (server) entries. Server

entries can only be accessed by explicitly using the empty string as the mailbox name.

Servers SHOULD ensure that mailbox annotations are automatically moved when the mailbox they refer to is renamed, i.e. the annotations follow the mailbox. Servers SHOULD delete annotations for a mailbox when the mailbox is deleted, so that a mailbox created with the same name as a previously existing mailbox does not inherit the old mailbox annotations. Servers SHOULD allow annotations on all 'types' of mailbox, including ones reporting \Noselect for their LIST response. Servers can implicitly remove \Noselect mailboxes when all child mailboxes are removed, and as such any annotations associated with the \Noselect mailbox SHOULD be removed.

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

4.2. GETMETADATA Command

This extension adds the GETMETADATA command. This allows clients to retrieve server annotations. Mailbox annotations are retrieved via the extended LIST command described in the next section.

This command is only available in authenticated or selected state [RFC3501].

```
Arguments:  entry-specifier
            attribute-specifier

Responses:  required METADATA response

Result:     OK - command completed
            NO - command failure: can't access annotations on
                 the server
            BAD - command unknown or arguments invalid
```

Example:

```
C: a GETMETADATA /comment value.priv
S: * METADATA /comment (value.priv "My comment")
S: a OK GETMETADATA complete
```

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

"*" and "%" wildcard characters can be used in the entry specifier to match one or more characters at that position, with the exception that "%" does not match the "/" hierarchy delimiter. Thus an entry specifier of "/%" would match entries such as "/comment" and "/version", but not "/comment/note".

Example:

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

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

Example:

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

In the above example, the contents of the "value" attributes for server entries at the top level of the entry 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 GETMETADATA command.

Example:

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

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

Example:

```
C: a GETMETADATA /comment (value.priv
                           size.priv)
S: * METADATA /comment (value.priv "My comment"
                        size.priv "1234")
S: a OK GETMETADATA complete
```

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

4.3. Extended LIST Command Extensions

This extension adds the METADATA LIST command selection and return option types, extending the LISTEXT extension. This allows clients to retrieve mailbox annotations.

The LISTEXT "METADATA" selection option type is used to request that the extended LIST command match mailboxes which have an annotation with a specific entry and value. It can also be used to test for the existence of a particular annotation entry on a mailbox. This selection option takes one or more entry/attribute/value triples to use as tests. A mailbox matches this criteria if it has an entry, attribute and value matching the ones specified. In the case of multiple criteria being specified, mailbox MUST only match when all criteria match ("and" operation). To test for the existence of an entry, a test for the attribute "value" with the empty string "" as the value argument can be used. To test for the absence of an entry, a test for the attribute "value" with NIL as the value argument can be used. If the COMPARATOR [I-D.ietf-imapext-i18n] extension is present, then the active comparator MUST be used when doing text comparisons of the value. Clients SHOULD only use entries defined with a "text" Content-Type in the selection option arguments.

The LISTEXT "METADATA" return option type is used to request that the extended LIST command return specific annotation entries for each mailbox returned in the LIST responses. A list of entries and attributes to return can be specified in a manner similar to the GETMETADATA command.

Example:

```
C: a LIST "" "INBOX" RETURN (METADATA /comment value.priv)
S: * LIST (\metadata /comment (value.priv "My comment"))
          "/" "INBOX"
S: a OK LIST complete
```

In the above example, the contents of the "value.priv" attribute for the "/comment" entry for the mailbox INBOX is requested by the client and returned by the server.

"*" and "%" wildcard characters can be used in the entry specifier to match one or more characters at that position, with the exception that "%" does not match the "/" hierarchy delimiter. Thus an entry specifier of "/%" would match entries such as "/comment" and "/version", but not "/comment/note".

Example:

```
C: a LIST "" "INBOX" RETURN (METADATA /* value.priv)
S: * LIST (\metadata /comment (value.priv "My comment")
             /version (value.priv "1.1")
             /motd/today (value.priv "Closed at 1 pm"))
          "/" "INBOX"
S: a OK LIST complete
```

In the above example, the contents of the "value.priv" attributes for any entries for the mailbox INBOX are requested by the client and returned by the server.

Example:

```
C: a LIST "" "INBOX" RETURN (METADATA /% value)
S: * LIST (\metadata /comment (value.priv "My comment")
                        (value.shared "Your comment")
             /motd (value.priv "Its sunny outside!"
                   (value.shared "Today is a Holiday"))
          "/" "INBOX"
S: a OK LIST complete
```

In the above example, the contents of the "value" attributes for entries for the mailbox INBOX at the top level of the entry 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 LIST command.

Example:

```
C: a LIST "" "INBOX" RETURN (METADATA (/comment /motd)
                                value.priv)
S: * LIST (\metadata /comment (value.priv "My comment")
             /motd (value.priv "Its sunny outside!"))
          "/" "INBOX"
S: a OK LIST complete
```

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

Example:

```
C: a LIST "" "INBOX" RETURN (METADATA /comment
                                (value.priv size.priv))
S: * LIST (\metadata /comment (value.priv "My comment"
                           size.priv "1234"))
          "/" "INBOX"
S: a OK LIST complete
```

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

Example:

```
C: a LIST "" ("INBOX" "FOOBOX" "BARBOX") RETURN
          (METADATA /comment value.priv)
S: * LIST (\metadata /comment (value.priv "My comment"))
          "/" "INBOX"
S: * LIST (\metadata /comment (value.priv "Another comment"))
          "/" "FOOBOX"
S: * LIST (\metadata /comment (value.priv NIL))
          "/" "BARBOX"
S: a OK LIST complete
```

In the above example, the contents of the "value.priv" attribute for the "/comment" entry for the mailboxes INBOX, FOOBOX and BARBOX are requested by the client and returned by the server. Note that the mailbox BARBOX does not contain the entry, hence NIL is returned for the attribute value.

Example:

```
C: a LIST (METADATA ("/comment" "value" "comm")
          "" "*"
S: * LIST () "/" "INBOX"
S: * LIST (\NoInferiors) "/" "FOOBOX"
```

```
S: a OK LIST complete
```

In the above example, the client requests that any mailbox in the entire hierarchy containing the text "comm" in any "value" attribute of the "/comment" entry be returned. Two mailboxes are returned in separate responses. Note that the client did not ask for annotations to be returned in the responses.

Example:

```
C: a LIST (METADATA ("/comment" "value" "")
          "" "*"
S: * LIST () "/" "INBOX"
S: * LIST (\NoInferiors) "/" "FOOBOX"
S: a OK LIST complete
```

In the above example, the client requests that any mailbox in the entire hierarchy containing the "/comment" entry be returned. Two mailboxes are returned in separate responses. Note that the client did not ask for annotations to be returned in the responses.

Example:

```
C: a LIST (METADATA ("/comment" "value" NIL)
          "" "*"
S: * LIST (\NoInferiors) "/" "BARBOX"
S: a OK LIST complete
```

In the above example, the client requests that any mailbox in the entire hierarchy that does not have a "/comment" entry be returned. One mailbox is returned in a response. Note that the client did not ask for annotations to be returned in the responses.

Example:

```
C: a LIST (METADATA ("/comment" "value" "comm")
          "" "*" RETURN (METADATA /comment size.priv)
S: * LIST (\metadata /comment (size.priv "10"))
          "/" "INBOX"
S: * LIST (\metadata /comment (size.priv "15"))
          "/" "FOOBOX"
S: a OK LIST complete
```

In the above example, the client requests that any mailbox in the entire hierarchy containing the text "comm" in any "value" attribute of the "/comment" entry be returned. Two mailboxes are

returned in separate responses. The client also asked for annotations to be returned in the responses.

4.3.1. Unsolicited LIST repsonse

Servers SHOULD send unsolicited LIST responses if mailbox annotations are changed by a third-party, allowing servers to keep clients updated with changes. Unsolicited mailbox annotations MUST only be returned for the currently selected mailbox.

Unsolicited LIST responses MUST only contain entry names, not the attributes and values. If the client wants to update any cached values it must explicitly retrieve those using a LIST command.

The LIST 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.

Example:

```
C: a NOOP
S: * LIST (\metadata /comment) "/" "INBOX"
S: a OK NOOP complete
```

In the above example, the server sends an unsolicited LIST response indicating that the "/comment" entry of the mailbox "INBOX" has been changed.

4.4. SETMETADATA Command

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

This command is only available in authenticated or selected state [RFC3501].

```
Arguments:  mailbox-name
            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,
                 or annotation too big or too many
            BAD - command unknown or arguments invalid
```

This command 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 a METADATA response containing the updated annotation data. Clients MUST NOT assume that a METADATA response will be sent, and MUST assume that if the command succeeds then the annotation has been changed.

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

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

If the server is unable to set a new annotation because it does not support private annotations on one of the specified mailboxes, the server MUST return a tagged NO response with a "[METADATA NOPRIVATE]" response code.

If the server is unable to set the annotations for one or more mailboxes matching the mailbox-name pattern, then the SETMETADATA command MUST fail and there MUST NOT be any changes to any of the matching mailboxes, even those for which annotations could have been changed successfully.

Example:

```
C: a SETMETADATA INBOX /comment
                          (value.priv "My new comment")
S: a OK SETMETADATA complete
```

In the above example, the entry "/comment" for the mailbox "INBOX" 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.

Example:

```
C: a SETMETADATA INBOX /comment (value.priv NIL)
S: a OK SETMETADATA complete
```

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

Annotations on multiple mailboxes can be set in a single SETMETADATA

command by using a wildcard specification for the mailbox name.

Example:

```
        C: a SETMETADATA INBOX.% /comment
                                  (value.priv "My new comment")
        S: a OK SETMETADATA complete
```

In the above example, the entry "/comment" for all mailboxes at the top-level of the "INBOX" hierarchy are created (if not already present) and the private attribute "value" are created respectively for each entry if not already present, or replaced if they previously existed.

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

Example:

```
        C: a SETMETADATA INBOX (/comment
                                (value.priv "My new comment")
                                /thread
                                (value.priv "REFERENCES ALL"))
        S: a OK SETMETADATA complete
```

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

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

Example:

```
        C: a SETMETADATA INBOX /comment
                              (value.priv "My new comment"
                               value.shared "foo's bar")
        S: a OK SETMETADATA complete
```

In the above example, the entry "/comment" for the mailbox "INBOX" is created (if not already present) and the attributes "value.priv" and "value.shared" are created if not already present, or replaced if they previously existed.

Example:

```
C: a SETMETADATA INBOX /comment
                          (value.priv "My new comment")
S: a NO [METADATA TOOMANY] SETMETADATA failed
```

In the above example, the server is unable to set the requested (new) annotation as it has reached the limit on the number of annotations it can support on the specified mailbox.

4.5. METADATA Response

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

Servers SHOULD send unsolicited METADATA responses if server annotations are changed by a third-party, allowing servers to keep clients updated with changes.

Unsolicited METADATA responses MUST only contain entry names, not the attributes and values. If the client wants to update any cached values it must explicitly retrieve those using a GETMETADATA command.

The METADATA 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 or selected state [RFC3501].

4.5.1. METADATA response with attributes and values

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

Example:

```
C: a GETMETADATA /comment value.priv
S: * METADATA /comment (value.priv "My comment")
S: a OK GETMETADATA complete
```

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

Example:

```
C: a GETMETADATA (/comment /motd) value.priv
S: * METADATA /comment (value.priv "My comment")
              /motd (value.priv "Its sunny outside!")
```

```
S: a OK GETMETADATA complete
```

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

Example:

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

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

Example:

```
C: a GETMETADATA /comment (value.priv size.priv)
S: * METADATA /comment (value.priv "My comment"
                                   size.priv "10")
S: a OK GETMETADATA complete
```

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

4.5.2. Unsolicited METADATA response without attributes and values

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

Example:

```
C: a NOOP
S: * METADATA (/comment)
S: a OK NOOP complete
```

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

Example:

```
C: a NOOP
S: * METADATA (/comment /motd)
S: a OK NOOP complete
```

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

5. Formal Syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) notation as specified in [RFC2234].

Non-terminals referenced but not defined below are as defined by [RFC3501] with the new definitions in [I-D.melnikov-imap-ext-abnf] 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.

```
    annotate-data     = "METADATA" SP entry-list

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

    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        =/ "METADATA"
                        ; defines the capability for this extension

    command-auth      =/ setmetadata / getmetadata
                        ; adds to original IMAP command

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

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

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

    entry-list        = entry-att *(SP entry-att) /
                        "(" entry *(SP entry) ")"
                        ; entry attribute-value pairs list for
                        ; GETMETADATA response, or
```

```
                        ; parenthesised entry list for unsolicited
                        ; notification of annotation changes

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

      getmetadata       = "GETMETADATA" SP entries SP attribs

      list-select-base-opt =/ "METADATA" SP
                              "(" list-select-metadata
                                   *(SP list-select-metadata) ")"
                        ; adds a new selection option type to
                        ; LISTEXT. When evaluating multiple entry,
                        ; attribute, value combinations, a match to
                        ; a mailbox must occur when all items match.


      list-select-metadata = entry-match SP att-select SP value
                        ; value set to the empty string means match
                        ; if that entry and attribute exist.
                        ; value set to NIL means match if that entry
                        ; and attribute do not exist.

      response-payload  =/ annotate-data
                        ; adds to original IMAP data responses

      resp-text-code    =/ "METADATA" SP ("TOOBIG" / "TOOMANY" /
                                          "NOPRIVATE")
                        ; new response codes for SETMETADATA
                        ; failures

      setmetadata       = "SETMETADATA" SP list-mailbox
                                        SP setentryatt
                        ; empty string for list-mailbox implies
                        ; server annotation.

      setentryatt       = entry-att / "(" entry-att *(SP entry-att) ")"

      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. 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 METADATA Registration

Please register the following IMAP METADATA item:

[ ] Entry [ ] Attribute

[ ] Mailbox [ ] Server

Name: _____

Description: _____

_____

_____

Content-type: _____

Contact person: _____

email: _____

6.2. Server 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 METADATA Registration

Please register the following IMAP METADATA item:

[x] Entry [ ] Attribute

[ ] Mailbox [x] Server

Name: /comment

Description: Defined in IMAP METADATA extension document.

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

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.2.2. /motd

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

Please register the following IMAP METADATA item:

[x] Entry [ ] Attribute

[ ] Mailbox [x] Server

Name: /motd

Description: Defined in IMAP METADATA extension document.

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

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.2.3. /admin

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

Please register the following IMAP METADATA item:

[x] Entry [ ] Attribute

[ ] Mailbox [x] Server

Name: /admin

Description: Defined in IMAP METADATA extension document.

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

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.3. Mailbox Entry Registrations

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

6.3.1. /comment

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

Please register the following IMAP METADATA item:

[x] Entry [ ] Attribute

[x] Mailbox [ ] Server

Name: /comment

Description: Defined in IMAP METADATA extension document.

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

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.3.2. /sort

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

Please register the following IMAP METADATA item:

[x] Entry [ ] Attribute

[x] Mailbox [ ] Server

Name: /sort

Description: Defined in IMAP METADATA extension document.

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

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.3.3. /thread

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

Please register the following IMAP METADATA item:

[x] Entry [ ] Attribute

[x] Mailbox [ ] Server

Name: /thread

Description: Defined in IMAP METADATA extension document.

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

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.3.4. /check

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

Please register the following IMAP METADATA item:

[x] Entry [ ] Attribute

[x] Mailbox [ ] Server

Name: /check

Description: Defined in IMAP METADATA extension document.

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

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.3.5. /checkperiod

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

Please register the following IMAP METADATA item:

[x] Entry [ ] Attribute

[x] Mailbox [ ] Server

Name: /checkperiod

Description: Defined in IMAP METADATA extension document.

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

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.4. Attribute Registrations

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

6.4.1. value

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

Please register the following IMAP METADATA item:

[ ] Entry [x] Attribute

[ ] Mailbox [ ] Server

Name: value

Description: Defined in IMAP METADATA extension document.

Content-type: -

Contact person: Cyrus Daboo

email: cyrus@daboo.name

6.4.2. size

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

Please register the following IMAP METADATA item:

[ ] Entry [x] Attribute

[ ] Mailbox [ ] Server

Name: size

Description: Defined in IMAP METADATA extension document.

Content-type: -

Contact person: Cyrus Daboo

email: cyrus@daboo.name

7. Security Considerations

Annotations whose values are intended to remain private MUST use .priv values, and not .shared values which may be accessible to other users.

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

8. References

8.1. Normative References

[I-D.ietf-imapext-i18n]
Newman, C., "Internet Message Access Protocol Internationalization", draft-ietf-imapext-i18n-06 (work in progress), March 2006.

[I-D.ietf-imapext-list-extensions]
Leiba, B. and A. Melnikov, "IMAP4 LIST Command Extensions", draft-ietf-imapext-list-extensions-16 (work in progress), March 2006.

[I-D.ietf-imapext-sort]
Crispin, M. and K. Murchison, "INTERNET MESSAGE ACCESS PROTOCOL - SORT AND THREAD EXTENSION", draft-ietf-imapext-sort-17 (work in progress), May 2004.

[I-D.melnikov-imap-ext-abnf]
Daboo, C. and A. Melnikov, "Collected extensions to IMAP4 ABNF", draft-melnikov-imap-ext-abnf-08 (work in progress), January 2006.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D., Ed. 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.

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

8.2. Informative References

[I-D.ietf-imapext-annotate]
Gellens, R. and C. Daboo, "IMAP ANNOTATE Extension",

draft-ietf-imapext-annotate-15 (work in progress), March 2006.

Appendix A. Acknowledgments

The ideas expressed in this document are based on the message annotation document that was co-authored by Randall Gellens. The participants of the IMAPext working group made significant contributions to this work.

Author's Address

Cyrus Daboo

Email: cyrus@daboo.name

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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