Network Working Group
Request for Comments: 5465
Updates: 5267
Category: Standards Track

A. Gulbrandsen
Oryx Mail Systems GmbH
C. King
A. Melnikov
Isode Ltd.

February 2009

The IMAP NOTIFY Extension

Status of This Memo

Copyright Notice

Abstract

This document defines an IMAP extension that allows a client to request specific kinds of unsolicited notifications for specified mailboxes, such as messages being added to or deleted from such mailboxes.

Table of Contents

1. Overview and Rationale

The IDLE command (defined in [RFC2177]) provides a way for the client to go into a mode where the IMAP server pushes it notifications about IMAP mailstore events for the selected mailbox. However, the IDLE extension doesn't restrict or control which server events can be sent, or what information the server sends in response to each event. Also, IDLE only applies to the selected mailbox, thus requiring an additional TCP connection per mailbox.

This document defines an IMAP extension that allows clients to express their preferences about unsolicited events generated by the server. The extension allows clients to only receive events that they are interested in, while servers know that they don't need to go to the effort of generating certain types of untagged responses.

Without the NOTIFY command defined in this document, an IMAP server will only send information about mailstore changes to the client in the following cases:

- as the result of a client command (e.g., FETCH responses to a FETCH or STORE command),
- as unsolicited responses sent just before the end of a command (e.g., EXISTS or EXPUNGE) as the result of changes in other sessions, and
- during an IDLE command.

The NOTIFY command extends what information may be returned in those last two cases, and also permits and requires the server to send information about updates between commands. The NOTIFY command also allows for the client to extend what information is sent unsolicited about the selected mailbox and to request some update information to be sent regarding other mailboxes.

The interaction between IDLE and NOTIFY commands is described in Section 4.

For the new messages delivered to or appended to the selected mailbox, the NOTIFY command can be used to request that a set of attributes be sent to the client in an unsolicited FETCH response. This allows a client to be a passive recipient of events and new mail and to be able to maintain full synchronisation without having to issue any subsequent commands except to modify the state of the mailbox on the server.

Some mobile clients, however, may want mail "pushed" only for mail that matches a SEARCH pattern. To meet that need, [RFC5267] is augmented by this document to extend the UPDATE return option to specify a list of fetch-atts to be returned when a new message is delivered or appended in another session.

2. Conventions Used in This Document

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

The acronym MSN stands for Message Sequence Numbers (see Section 2.3.1.2 of [RFC3501]).

Example lines prefaced by "C:" are sent by the client and ones prefaced by "S:", by the server. "[...]" means elision.

3. The NOTIFY Extension

IMAP servers that support this extension advertise the NOTIFY capability. This extension adds the NOTIFY command as defined in Section 5.1.

A server implementing this extension is not required to implement LIST-EXTENDED [RFC5258], even though a NOTIFY-compliant server must be able to return extended LIST responses, defined in [RFC5258].

3.1. The NOTIFY Command

```
Arguments:  "SET"
            Optional STATUS indicator
            Mailboxes to be watched
            Events about which to notify the client

Or
Arguments:  "NONE"

Responses:  Possibly untagged STATUS responses (for SET)

Result:     OK - The server will notify the client as requested.
            NO -  Unsupported NOTIFY event, NOTIFY too complex or
                  expensive, etc.
            BAD - Command unknown, invalid, unsupported, or has
                  unknown arguments.
```

The NOTIFY command informs the server that the client listens for event notifications all the time (even when no command is in progress), and requests the server to notify it about the specified set of events. The NOTIFY command has two forms. NOTIFY NONE specifies that the client is not interested in any kind of event happening on the server. NOTIFY SET replaces the current list of interesting events with a new list of events.

Until the NOTIFY command is used for the first time, the server only sends notifications while a command is being processed, and notifies the client about these events on the selected mailbox (see Section 5 for definitions): MessageNew, MessageExpunge, or FlagChange. It does not notify the client about any events on other mailboxes.

The effect of a successful NOTIFY command lasts until the next NOTIFY command or until the IMAP connection is closed.

A successful NOTIFY SET command MUST cause the server to immediately return any accumulated changes to the currently selected mailbox (if any), such as flag changes and new or expunged messages. Thus, a successful NOTIFY SET command implies an implicit NOOP command.

The NOTIFY SET command can request notifications of message-related changes to the selected mailbox, whatever that may be at the time the message notifications are being generated. This is done by specifying either the SELECTED or the SELECTED-DELAYED mailbox selector (see Section 6.1) in the NOTIFY SET command. If the SELECTED/SELECTED-DELAYED mailbox selector is not specified in the NOTIFY SET command, this means that the client doesn't want to receive any <message-event>s for the currently selected mailbox. This is the same as specifying SELECTED NONE.

The client can also request notifications on other mailboxes by name or by a limited mailbox pattern match. Message-related notifications returned for the currently selected mailbox will be those specified by the SELECTED/SELECTED-DELAYED mailbox specifier, even if the selected mailbox also appears by name (or matches a pattern) in the command. Non-message-related notifications are controlled by mailbox specifiers other than SELECTED/SELECTED-DELAYED.

If the NOTIFY command enables MessageNew, MessageExpunge, AnnotationChange, or FlagChange notifications for a mailbox other than the currently selected mailbox, and the client has specified the STATUS indicator parameter, then the server MUST send a STATUS response for that mailbox before NOTIFY's tagged OK. If MessageNew is enabled, the STATUS response MUST contain MESSAGES, UIDNEXT, and UIDVALIDITY. If MessageExpunge is enabled, the STATUS response MUST contain MESSAGES. If either AnnotationChange or FlagChange are

included and the server also supports the CONDSTORE [RFC4551] and/or QRESYNC [RFC5162] extensions, the STATUS response MUST contain UIDVALIDITY and HIGHESTMODSEQ. Absence of the STATUS indicator parameter allows the client to avoid the additional STATUS responses. This might be useful if the client already retrieved this information before issuing the NOTIFY command.

Clients are advised to limit the number of mailboxes used with NOTIFY. Particularly, if a client asks for events for all accessible mailboxes, the server may swamp the client with updates about shared mailboxes. This may reduce the client's battery life. Also, this wastes both server and network resources.

For each mailbox specified, the server verifies that the client has access using the following test:

- If the name does not refer to an existing mailbox, the server MUST ignore it.

- If the name refers to a mailbox that the client can't LIST, the server MUST ignore it. For a server that implements [RFC4314], this means that if the client doesn't have the 'l' (lookup) right for the name, then the server MUST ignore the mailbox. This behavior prevents disclosure of potentially confidential information to clients who don't have rights to know it.

- If the name refers to a mailbox that the client can LIST (e.g., it has the 'l' right from [RFC4314]), but the client doesn't have another right required for processing of the specified event(s), then the server MUST respond with an untagged extended LIST response containing the \NoAccess name attribute.

The server SHOULD return the tagged OK response if the client has access to at least one of the mailboxes specified in the current list of interesting events. The server MAY return the tagged NO response if the client has no access to any of the specified mailboxes and no access can ever be granted in the future (e.g., the client specified an event for 'Subtree Bar/Foo', 'Bar/Foo' doesn't exist, and LIST returns \Noinferiors for the parent 'Bar').

If the notification would be prohibitively expensive for the server (e.g., "notify me of all flag changes in all mailboxes"), the server MAY refuse the command with a tagged NO [NOTIFICATIONOVERFLOW] response.

If the client requests information for events of an unsupported type, the server MUST refuse the command with a tagged NO response (not a BAD). This response SHOULD contain the BADEVENT response code, which MUST list names of all events supported by the server.

Here's an example:

```
      S: * OK [CAPABILITY IMAP4REV1 NOTIFY]
      C: a login bob alice
      S: a OK Password matched
      C: b notify set status (selected MessageNew (uid
         body.peek[header.fields (from to subject)]) MessageExpunge)
         (subtree Lists MessageNew)
      S: * STATUS Lists/Lemonade (UIDVALIDITY 4 UIDNEXT 9999 MESSAGES
         500)
      S: [...]
      S: * STATUS Lists/Im2000 (UIDVALIDITY 901 UIDNEXT 1 MESSAGES 0)
      S: b OK done
      C: c select inbox
      S: [...] (the usual 7-8 responses to SELECT)
      S: c OK INBOX selected
            (Time passes.  A new message is delivered to mailbox
            Lists/Lemonade.)
      S: * STATUS Lists/Lemonade (UIDVALIDITY 4 UIDNEXT 10000
         MESSAGES 501)
            (Time passes.  A new message is delivered to inbox.)
      S: * 127 FETCH (UID 127001 BODY[HEADER.FIELDS (From To
         Subject)] {75}
      S: Subject: Re: good morning
      S: From: alice@example.org
      S: To: bob@example.org
      S:
      S: )
            (Time passes.  The client decides it wants to know about
            one more mailbox.  As the client already knows necessary
            STATUS information for all mailboxes below the Lists
            mailbox, and because "notify set status" would cause
            STATUS responses for *all* mailboxes specified in the
            NOTIFY command, including the ones for which the client
            already knows STATUS information, the client issues an
            explicit STATUS request for the mailbox to be added to
            the watch list, followed by the NOTIFY SET without the
            STATUS parameter.)
      C: d STATUS misc (UIDVALIDITY UIDNEXT MESSAGES)
      S: * STATUS misc (UIDVALIDITY 1 UIDNEXT 999)
      S: d STATUS completed
```

```
C: e notify set (selected MessageNew (uid
   body.peek[header.fields (from to subject)]) MessageExpunge)
   (subtree Lists MessageNew) (mailboxes misc MessageNew)
S: e OK done
```

4. Interaction with the IDLE Command

If IDLE [RFC2177] (as well as this extension) is supported, then while processing any IDLE command, the server MUST send exactly the same events as instructed by the client using the NOTIFY command.

NOTIFY makes IDLE unnecessary for some clients. If a client does not use MSNs and '*' in commands, it can request MessageExpunge and MessageNew for the selected mailbox by using the NOTIFY command instead of entering the IDLE mode.

A client that uses MSNs and '*' in commands can still use the NOTIFY command if it specifies the SELECTED-DELAYED mailbox specifier in the NOTIFY command.

5. Event Types

Only some of the events in [RFC5423] can be expressed in IMAP, and for some of them there are several possible ways to express the event.

This section specifies the events of which an IMAP server can notify an IMAP client, and how.

The server SHOULD omit notifying the client if the event is caused by this client. For example, if the client issues CREATE and has requested a MailboxName event that would cover the newly created mailbox, the server SHOULD NOT notify the client of the MailboxName change.

All event types described in this document require the 'l' and 'r' rights (see [RFC4314]) on all observed mailboxes. Servers that don't implement [RFC4314] should map the above rights to their access-control model.

If the FlagChange and/or AnnotationChange events are specified, MessageNew and MessageExpunge MUST also be specified by the client. Otherwise, the server MUST respond with the tagged BAD response.

If one of MessageNew or MessageExpunge is specified, then both events MUST be specified. Otherwise, the server MUST respond with the tagged BAD response.

The client can instruct the server not to send an event by omitting the necessary event from the list of events specified in NOTIFY SET, by using the NONE event specifier in the NOTIFY SET, or by using NOTIFY NONE. In particular, NOTIFY SET ... NONE can be used as a snapshot facility by clients.

5.1. FlagChange and AnnotationChange

If the flag and/or message annotation change happens in the selected mailbox, the server MUST notify the client by sending an unsolicited FETCH response, which MUST include UID and FLAGS/ANNOTATION FETCH data items. It MAY also send new FLAGS and/or OK [PERMANENTFLAGS ...] responses.

If a search context is in effect as specified in [RFC5267], an ESEARCH ADDTO or ESEARCH REMOVEFROM will also be generated, if appropriate. In this case, the FETCH response MUST precede the ESEARCH response.

If the change happens in another mailbox, then the server responds with a STATUS response. The exact content of the STATUS response depends on various factors. If CONDSTORE [RFC4551] and/or QRESYNC [RFC5162] are enabled by the client, then the server sends a STATUS response that includes at least HIGHESTMODSEQ and UIDVALIDITY status data items. If the number of messages with the \Seen flag changes, the server MAY also include the UNSEEN data item in the STATUS response. If CONDSTORE/QRESYNC is not enabled by the client and the server chooses not to include the UNSEEN data item, the server does not notify the client. When this event is requested, the server MUST notify the client about mailbox UIDVALIDITY changes. This is done by sending a STATUS response that includes UIDVALIDITY.

FlagChange covers the MessageRead, MessageTrash, FlagsSet, and FlagsClear events in [RFC5423].

Example in the selected mailbox:

```
S: * 99 FETCH (UID 9999 FLAGS ($Junk))
```

And in another mailbox, with CONDSTORE in use:

```
S: * STATUS Lists/Lemonade (HIGHESTMODSEQ 65666665 UIDVALIDITY
101)
```

5.2. MessageNew

This covers both MessageNew and MessageAppend in [RFC5423].

If the new/appended message is in the selected mailbox, the server notifies the client by sending an unsolicited EXISTS response, followed by an unsolicited FETCH response containing the information requested by the client. A FETCH response SHOULD NOT be generated for a new message created by the client on this particular connection, for instance, as the result of an APPEND or COPY command to the selected mailbox performed by the client itself. The server MAY also send a RECENT response, if the server marks the message as \Recent.

Note that a single EXISTS response can be returned for multiple MessageAppend/MessageNew events.

If a search context is in effect as specified in [RFC5267], an ESEARCH ADDTO will also be generated, if appropriate. In this case, the EXISTS response MUST precede the ESEARCH response. Both the NOTIFY command and the SEARCH and SORT commands (see Section 7) can specify attributes to be returned for new messages. These attributes SHOULD be combined into a single FETCH response. The server SHOULD avoid sending duplicate data. The FETCH response(s) MUST follow any ESEARCH ADDTO responses.

If the new/appended message is in another mailbox, the server sends an unsolicited STATUS (UIDNEXT MESSAGES) response for the relevant mailbox. If the CONDSTORE extension [RFC4551] and/or the QRESYNC extension [RFC5162] is enabled, the HIGHESTMODSEQ status data item MUST be included in the STATUS response.

The client SHOULD NOT use FETCH attributes that implicitly set the \seen flag, or that presuppose the existence of a given bodypart. UID, MODSEQ, FLAGS, ENVELOPE, BODY.PEEK[HEADER.FIELDS... and BODY/BODYSTRUCTURE may be the most useful attributes.

Note that if a client asks to be notified of MessageNew events with the SELECTED mailbox specifier, the number of messages can increase at any time, and therefore the client cannot refer to a specific message using the MSN/UID '*'.

Example in the selected mailbox:

```
S: * 444 EXISTS
S: * 444 FETCH (UID 9999)
```

And in another mailbox, without CONDSTORE enabled:

```
S: * STATUS Lists/Lemonade (UIDNEXT 10002 MESSAGES 503)
```

5.3. MessageExpunge

If the expunged message or messages are in the selected mailbox, the server notifies the client using EXPUNGE (or VANISHED, if [RFC5162] is supported by the server and enabled by the client).

If a search context is in effect, as specified in [RFC5267], an ESEARCH REMOVEFROM will also be generated, if appropriate.

If the expunged message or messages are in another mailbox, the server sends an unsolicited STATUS (UIDNEXT MESSAGES) response for the relevant mailbox. If the QRESYNC [RFC5162] extension is enabled, the HIGHESTMODSEQ data item MUST be included in the STATUS response as well.

Note that if a client requests MessageExpunge with the SELECTED mailbox specifier, the meaning of an MSN can change at any time, so the client cannot use MSNs in commands anymore. For example, such a client cannot use FETCH, but has to use UID FETCH. The meaning of '*' can also change when messages are added or expunged. A client wishing to keep using MSNs can either use the SELECTED-DELAYED mailbox specifier or can avoid using the MessageExpunge event entirely.

The MessageExpunge notification covers both MessageExpunge and MessageExpire events from [RFC5423].

Example in the selected mailbox, without QRESYNC:

```
S: * 444 EXPUNGE
```

The same example in the selected mailbox, with QRESYNC:

```
S: * VANISHED 5444
```

And in another mailbox, when QRESYNC is not enabled:

```
S: * STATUS misc (UIDNEXT 999 MESSAGES 554)
```

5.4. MailboxName

These notifications are sent if an affected mailbox name was created (with CREATE), deleted (with DELETE), or renamed (with RENAME). For a server that implements [RFC4314], granting or revocation of the 'l' right to the current user on the affected mailbox MUST be considered mailbox creation or deletion, respectively. If a mailbox is created or deleted, the mailbox itself and its direct parent (whether it is an existing mailbox or not) are considered to be affected.

The server notifies the client by sending an unsolicited LIST response for each affected mailbox name. If, after the event, the mailbox name does not refer to a mailbox accessible to the client, the \Nonexistent flag MUST be included.

For each LISTable mailbox renamed, the server sends an extended LIST response [RFC5258] for the new mailbox name, containing the OLDNAME extended data item with the old mailbox name. When a mailbox is renamed, its children are renamed too. No additional MailboxName events are sent for children in this case. When INBOX is renamed, a new INBOX is assumed to be created. No MailboxName event is sent for INBOX in this case.

If the server automatically subscribes a mailbox when it is created or renamed, then the unsolicited LIST response for each affected subscribed mailbox name MUST include the \Subscribed attribute (see [RFC5258]). The server SHOULD also include \HasChildren or \HasNoChildren attributes [RFC5258] as appropriate.

Example of a newly created mailbox (or granting of the 'l' right on the mailbox):

```
S: * LIST () "/" "NewMailbox"
```

And a deleted mailbox (or revocation of the 'l' right on the mailbox):

```
S: * LIST (\NonExistent) "." "INBOX.DeletedMailbox"
```

Example of a renamed mailbox:

```
S: * LIST () "/" "NewMailbox" ("OLDNAME" ("OldMailbox"))
```

5.5. SubscriptionChange

The server notifies the client by sending an unsolicited LIST response for each affected mailbox name. If and only if the mailbox is subscribed after the event, the \Subscribed attribute (see [RFC5258]) is included. Note that in the LIST response, all mailbox attributes MUST be accurately computed (this differs from the behavior of the LSUB command).

Example:

```
S: * LIST (\Subscribed) "/" "SubscribedMailbox"
```

5.6. MailboxMetadataChange

Support for this event type is OPTIONAL unless the METADATA extension [RFC5464] is also supported by the server, in which case support for this event type is REQUIRED.

A client willing to receive unsolicited METADATA responses as a result of using the MailboxMetadataChange event in the NOTIFY command doesn't have to issue ENABLE METADATA.

The server sends an unsolicited METADATA response (as per Section 4.4.2 of [RFC5464]). If possible, only the changed metadata SHOULD be included, but if the server can't detect a change to a single metadata item, it MAY include all metadata items set on the mailbox. If a metadata item is deleted (set to NIL), it MUST always be included in the METADATA response.

Example:

```
S: * METADATA "INBOX" /shared/comment
```

5.7. ServerMetadataChange

Support for this event type is OPTIONAL unless the METADATA or the METADATA-SERVER extension [RFC5464] is also supported by the server, in which case support for this event type is REQUIRED.

A client willing to receive unsolicited METADATA responses as a result of using the ServerMetadataChange event in the NOTIFY command doesn't have to issue ENABLE METADATA or ENABLE METADATA-SERVER.

The server sends an unsolicited METADATA response (as per Section 4.4.2 of [RFC5464]). Only the names of changed metadata entries SHOULD be returned in such METADATA responses. If a metadata item is deleted (set to NIL), it MUST always be included in the METADATA response.

Example:

```
S: * METADATA "" /shared/comment
```

5.8. Notification Overflow

If the server is unable or unwilling to deliver as many notifications as it is being asked to, it may disable notifications for some or all clients. It MUST notify these clients by sending an untagged "OK [NOTIFICATIONOVERFLOW]" response and behave as if a NOTIFY NONE command had just been received.

Example:

```
S: * OK [NOTIFICATIONOVERFLOW] ...A comment can go here...
```

5.9. ACL (Access Control List) Changes

Even if NOTIFY succeeds, it is still possible to lose access to the mailboxes being monitored at a later time. If this happens, the server MUST stop monitoring these mailboxes. If access is later granted, the server MUST restart event monitoring.

The server SHOULD return the LIST response with the \NoAccess name attribute if and when the mailbox loses the 'l' right. Similarly, the server SHOULD return the LIST response with no \NoAccess name attribute if the mailbox was previously reported as having \NoAccess and the 'l' right is later granted.

6. Mailbox Specification

Mailboxes to be monitored can be specified in several different ways.

Only 'SELECTED' and 'SELECTED-DELAYED' (Section 6.1) match the currently selected mailbox. All other mailbox specifications affect other (non-selected) mailboxes.

Note that multiple <event-group>s can apply to the same mailbox. The following example demonstrates this. In this example, MessageNew and MessageExpunge events are reported for INBOX, due to the first <event-group>. A SubscriptionChange event will also be reported for INBOX, due to the second <event-group>.

```
C: a notify set (mailboxes INBOX (Messagenew messageExpunge))
   (personal (SubscriptionChange))
```

A typical client that supports the NOTIFY extension would ask for events on the selected mailbox and some named mailboxes.

In the next example, the client asks for FlagChange events for all personal mailboxes except the currently selected mailbox. This is different from the previous example because SELECTED overrides all other message event definitions for the currently selected mailbox (see Section 3.1).

```
C: a notify set (selected (Messagenew (uid flags) messageExpunge))
   (personal (MessageNew FlagChange MessageExpunge))
```

6.1. Mailbox Specifiers Affecting the Currently Selected Mailbox

Only one of the mailbox specifiers affecting the currently selected mailbox can be specified in any NOTIFY command. The two such mailbox specifiers (SELECTED and SELECTED-DELAYED) are described below.

Both refer to the mailbox that was selected using either SELECT or EXAMINE (see [RFC3501], Sections 6.3.1 and 6.3.2). When the IMAP connection is not in the selected state, such mailbox specifiers don't refer to any mailbox.

The mailbox specifiers only apply to <message-event>s. It is an error to specify other types of events with either the SELECTED or the SELECTED-DELAYED selector.

6.1.1. Selected

The SELECTED mailbox specifier requires the server to send immediate notifications for the currently selected mailbox about all specified <message-event>s.

6.1.2. Selected-Delayed

The SELECTED-DELAYED mailbox specifier requires the server to delay a MessageExpunge event until the client issues a command that allows returning information about expunged messages (see Section 7.4.1 of [RFC3501] for more details), for example, till a NOOP or an IDLE command has been issued. When SELECTED-DELAYED is specified, the server MAY also delay returning other <message-event>s until the client issues one of the commands specified above, or it MAY return them immediately.

6.2. Personal

Personal refers to all selectable mailboxes in the user's personal namespace(s), as defined in [RFC2342].

6.3. Inboxes

Inboxes refers to all selectable mailboxes in the user's personal namespace(s) to which messages may be delivered by a Message Delivery Agent (MDA) (see [EMAIL-ARCH], particularly Section 4.3.3).

If the IMAP server cannot easily compute this set, it MUST treat "inboxes" as equivalent to "personal".

6.4. Subscribed

Subscribed refers to all mailboxes subscribed to by the user.

If the subscription list changes, the server MUST reevaluate the list.

6.5. Subtree

Subtree is followed by a mailbox name or list of mailbox names. A subtree refers to all selectable mailboxes that are subordinate to the specified mailbox plus the specified mailbox itself.

6.6. Mailboxes

Mailboxes is followed by a mailbox name or a list of mailbox names. The server MUST NOT do a wildcard expansion. This means there is no special treatment for the LIST wildcard characters ('*' and '%') if they are present in mailbox names.

7. Extension to SEARCH and SORT Commands

If the server that supports the NOTIFY extension also supports CONTEXT=SEARCH and/or CONTEXT=SORT as defined in [RFC5267], the UPDATE return option is extended so that a client can request that FETCH attributes be returned when a new message is added to the context result set.

For example:

```
C: a00 SEARCH RETURN (COUNT UPDATE (UID BODY[HEADER.FIELDS (TO
   FROM SUBJECT)])) FROM "boss"
S: * ESEARCH (TAG "a00") (COUNT 17)
S: a00 OK
   [...a new message is delivered...]
S: * EXISTS 93
S: * 93 FETCH (UID 127001 BODY[HEADER.FIELDS (FROM TO SUBJECT)]
   {76}
S: Subject: Re: good morning
S: From: myboss@example.org
S: To: bob@example.org
S:
S: )
S: * ESEARCH (TAG "a00") ADDTO (0 93)
```

Note that the EXISTS response MUST precede any FETCH responses, and together they MUST precede the ESEARCH response.

No untagged FETCH response SHOULD be returned if a message becomes a member of UPDATE SEARCH due to flag or annotation changes.

8. Formal Syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) notation as specified in [RFC5234]. [RFC3501] defines the non-terminals "capability", "command-auth", "mailbox", "mailbox-data", "resp-text-code", and "search-key". The "modifier-update" non-terminal is defined in [RFC5267]. "mbx-list-oflag" is defined in [RFC3501] and updated by [RFC5258].

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. For example, the <filter-mailboxes-selected> non-terminal value "SELECTED" must be treated in the same way as "Selected" or "selected".

```
   capability      =/ "NOTIFY"

   command-auth    =/ notify

   notify          = "NOTIFY" SP
                   (notify-set / notify-none)

   notify-set      = "SET" [status-indicator] SP event-groups
                   ; Replace registered notification events
                   ; with the specified list of events

   notify-none     = "NONE"
                   ; Cancel all registered notification
                   ; events.  The client is not interested
                   ; in receiving any events.

   status-indicator = SP "STATUS"

   one-or-more-mailbox = mailbox / many-mailboxes

   many-mailboxes  = "(" mailbox *(SP mailbox) ")"

   event-groups    = event-group *(SP event-group)

   event-group     = "(" filter-mailboxes SP events ")"
                   ;; Only <message-event>s are allowed in <events>
                   ;; when <filter-mailboxes-selected> is used.

   filter-mailboxes = filter-mailboxes-selected /
                   filter-mailboxes-other
```

```
filter-mailboxes-other = "inboxes" / "personal" / "subscribed" /
                ( "subtree" SP one-or-more-mailbox ) /
                ( "mailboxes" SP one-or-more-mailbox )

filter-mailboxes-selected = "selected" / "selected-delayed"
                ;; Apply to the currently selected mailbox only.
                ;; Only one of them can be specified in a NOTIFY
                ;; command.

events          = ( "(" event *(SP event) ")" ) / "NONE"
                ;; As in [MSGEVENT].
                ;; "NONE" means that the client does not wish
                ;; to receive any events for the specified
                ;; mailboxes.

event           = message-event /
                mailbox-event / user-event / event-ext

message-event   = ( "MessageNew" [SP
                    "(" fetch-att *(SP fetch-att) ")" ] )
                / "MessageExpunge"
                / "FlagChange"
                / "AnnotationChange"
                ;; "MessageNew" includes "MessageAppend" from
                ;; [MSGEVENT]. "FlagChange" is any of
                ;; "MessageRead", "MessageTrash", "FlagsSet",
                ;; "FlagsClear" [MSGEVENT]. "MessageExpunge"
                ;; includes "MessageExpire" [MSGEVENT].
                ;; MessageNew and MessageExpunge MUST always
                ;; be specified together.  If FlagChange is
                ;; specified, then MessageNew and MessageExpunge
                ;; MUST be specified as well.
                ;; The fett-att list may only be present for the
                ;; SELECTED/SELECTED-DELAYED mailbox filter
                ;; (<filter-mailboxes>).

mailbox-event   = "MailboxName" /
                "SubscriptionChange" / "MailboxMetadataChange"
                ; "SubscriptionChange" includes
                ; MailboxSubscribe and MailboxUnSubscribe.
                ; "MailboxName" includes MailboxCreate,
                ; "MailboxDelete" and "MailboxRename".

user-event      = "ServerMetadataChange"

event-ext       = atom
                ;; For future extensions
```

```
oldname-extended-item =  "OLDNAME" SP "(" mailbox ")"
                ;; Extended data item (mbox-list-extended-item)
                ;; returned in a LIST response when a mailbox is
                ;; renamed.
                ;; Note 1: the OLDNAME tag can be returned
                ;; with or without surrounding quotes, as per
                ;; mbox-list-extended-item-tag production.

resp-text-code  =/ "NOTIFICATIONOVERFLOW" /
                unsupported-events-code

message-event-name   = "MessageNew" /
                "MessageExpunge" / "FlagChange" /
                "AnnotationChange"

event-name = message-event-name / mailbox-event /
                user-event

unsupported-events-code = "BADEVENT"
                SP "(" event-name *(SP event-name) ")"

modifier-update = "UPDATE"
                [ "(" fetch-att *(SP fetch-att) ")" ]

mbx-list-oflag =/ "\NoAccess"
```

9. Security Considerations

It is very easy for a client to deny itself service using NOTIFY. Asking for all events on all mailboxes may work on a small server, but with a big server, can swamp the client's network connection or processing capability. In the worst case, the server's processing could also degrade the service it offers to other clients.

Server authors should be aware that if a client issues requests and does not listen to the resulting responses, the TCP window can easily fill up, and a careless server might block. This problem also exists in plain IMAP; however, this extension magnifies the problem.

This extension makes it possible to retrieve messages immediately when they are added to the mailbox. This makes it wholly impractical to delete sensitive messages using programs like imapfilter. Using SIEVE [RFC5228] or similar is much better.

10. IANA Considerations

The IANA has added NOTIFY to the list of IMAP extensions.

10.1. Initial LIST-EXTENDED Extended Data Item Registrations

The following entry has been added to the LIST-EXTENDED response registry [RFC5258]:

To: iana@iana.org
Subject: Registration of OLDNAME LIST-EXTENDED extended data item

LIST-EXTENDED extended data item tag: OLDNAME

LIST-EXTENDED extended data item description: The OLDNAME extended data item describes the old mailbox name for the mailbox identified by the LIST response.

Which LIST-EXTENDED option(s) (and their types) causes this extended data item to be returned (if any): none

Published specification : RFC 5465, Section 5.4.

Security considerations: none

Intended usage: COMMON

Person and email address to contact for further information: Alexey Melnikov <Alexey.Melnikov@isode.com>

Owner/Change controller: iesg@ietf.org

11. Acknowledgments

The authors gratefully acknowledge the help of Peter Coates, Dave Cridland, Mark Crispin, Cyrus Daboo, Abhijit Menon-Sen, Timo Sirainen, and Eric Burger. In particular, Peter Coates contributed lots of text and useful suggestions to this document.

Various examples are copied from other RFCs.

This document builds on one published and two unpublished drafts by the same authors.

12. Normative References

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

[RFC2177] Leiba, B., "IMAP4 IDLE command", RFC 2177, June 1997.

[RFC2342] Gahrns, M. and C. Newman, "IMAP4 Namespace", RFC 2342, May 1998.

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 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.

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

[RFC5162] Melnikov, A., Cridland, D., and C. Wilson, "IMAP4 Extensions for Quick Mailbox Resynchronization", RFC 5162, March 2008.

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

[RFC5258] Leiba, B. and A. Melnikov, "Internet Message Access Protocol version 4 - LIST Command Extensions", RFC 5258, June 2008.

[RFC5267] Cridland, D. and C. King, "Contexts for IMAP4", RFC 5267, July 2008.

[RFC5423] Newman, C. and R. Gellens, "Internet Message Store Events", RFC 5423, Month 2009.

[RFC5464] Daboo, C., "The IMAP METADATA Extension", RFC 5464, February 2009.

13. Informative References

[RFC5228] Guenther, P., Ed., and T. Showalter, Ed., "Sieve: An Email Filtering Language", RFC 5228, January 2008.

[EMAIL-ARCH] Crocker, D., "Internet Mail Architecture", Work in Progress, October 2008.

Authors' Addresses

Arnt Gulbrandsen
Oryx Mail Systems GmbH
Schweppermannstr. 8
D-81671 Muenchen
Germany

EMail: arnt@oryx.com

Curtis King
Isode Ltd
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

EMail: Curtis.King@isode.com

Alexey Melnikov
Isode Ltd
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

EMail: Alexey.Melnikov@isode.com