Network Working Group
Internet-Draft
Intended Status: Proposed Standard

Curtis King
Alexey Melnikov
Isode Ltd.
Arnt Gulbrandsen
Oryx Mail Systems GmbH

November 14, 2007

# The IMAP NOTIFY Extension
# draft-ietf-lemonade-imap-notify-01.txt

## Status of this Memo

## Copyright Notice

## Abstract

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

## 1. 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].

Formal syntax is defined by [RFC4234] as extended by [RFC3501] and [RFC4466].

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.

## 2. Overview

The IDLE command (defined in [RFC2177]) provides a way for the client to go into a mode where the IMAP server pushes 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 they are interested in, while servers know that they don't need to go into effort of generating certain types of untagged responses.

IMAP servers which support this extension advertise the X-DRAFT-W00-NOTIFY extension.

Comments regarding this draft may be sent either to the lemonade@ietf.org mailing list or to the authors.

## 3. The NOTIFY Command

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

Or
Arguments: "NONE"
```

```
Responses: Possibly untagged STATUS responses (for ADD/SET)

Result: OK - The server will notify the client as requested.
        NO - NOTIFY too complex or expensive, etc.
       BAD - Command unknown, invalid, unsupported or 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 3 forms. The NOTIFY NONE specifies that the client is not interested in any kind of event happening on the server. The NOTIFY ADD prepends one or more events to the list of events which are interesting to the client. The NOTIFY SET replaces the current list of interesting events with a new list of events. (Note that NOTIFY SET <events> is effectively the same as NOTIFY NONE followed by NOTIFY ADD <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, FlagChange and (if [ANNOTATE] is being used) AnnotationChange. It does not notify the client about any events on other mailboxes.

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

A successful NOTIFY ADD/SET command MUST cause the server to immediately return any accumulated changes to the mailbox (if any), such as flag changes, new or expunged messages. (This is equivalent to NOOP command being issued by the client just before the NOTIFY ADD/SET command.)

If the NOTIFY command enables MessageNew, MessageExpunge, AnnotationChange or FlagChange notification for a 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, 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 has 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 wastes both server resources and network traffic.

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

- If the name refers to a mailbox which the client can LIST (e.g. it has the 'l' right from [RFC4314]), but misses 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. [[Alexey: Note, the newly defined \NoAccess doesn't mean that the client doesn't have any rights other than 'l'. The \NoAccess is only meaningful in the context of the specified NOTIFY command.]]

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 (e.g. QuotaExceed and the server does not advertise the QUOTA extension defined in [RFC2087]), the server MUST refuse the command with a tagged BAD response.

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)]) (all)
            MessageExpunge) (subtree Lists MessageNew (uid) (all))
         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.)
         C: d notify add status (mailboxes misc MessageNew (uid) (all))
         S: * STATUS misc (UIDVALIDITY 1 UIDNEXT 999)
                (This command enables notification on one mailbox and
                 otherwise changes nothing, so one STATUS response is
                 sent.)
         S: d OK done
```

## 4. Interaction with the IDLE Command

If IDLE (as well as this extension) is supported, while processing IDLE the server MUST send 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 using the NOTIFY command instead of entering the IDLE mode.

## 5. Event Types

Only some of the events in [MSGEVENT] 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 MailboxCreate event that would cover the newly created mailbox, the server SHOULD NOT notify the client of the MailboxCreate change.

All event types require the 'l' and 'r' rights (see [RFC4314]) on all observed mailboxes. AdminMailbox and the quota-related event types additionally require the 'a' right.  Servers that don't implement [RFC4314] should map the above rights to their access control model.

If the client instructs the server not to send MessageNew or MessageExpunge for the selected mailbox, the server MUST still send EXISTS and EXPUNGE responses as required by IMAP (see [RFC3501] section 7). In other words, MessageExpunge instructs the server to notify the client immediately, and the lack of MessageExpunge instructs the server to notify the client during execution of the next command as specified in [RFC3501].  MessageNew is handled similarly by the server.

### 5.1. FlagChange and AnnotationChange

If the flag/annotation change happens in the selected mailbox, the server notifies 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 the change happens in another mailbox, then the response depends on whether CONDSTORE [RFC4551] is being used. If so, the server sends a STATUS (HIGHESTMODSEQ) response. (Note that whenever mailbox UIDVALIDITY changes, the server MUST also include UIDVALIDITY in the STATUS response.)  If not, the server does not notify the client.

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

[[Open Issue: Filip Navara requested for STATUS (UNSEEN) to be sent

for MessageRead. Arnt considers that unsound, since it involves processing all messages in a mailbox after an event affecting only one message, and since it's not reliable anyway.]]

Example in the selected mailbox:

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

And in another, with CONDSTORE in use:

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

## 5.2. MessageNew

This covers both MessageNew and MessageAppend in [MSGEVENT].

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. 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 the new/appended message is in another mailbox, the server sends an unsolicited STATUS (UIDNEXT MESSAGES) response for the relevant mailbox. If CONDSTORE (defined in [RFC4551]) is in use, 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, 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, without CONDSTORE:

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

## 5.3. MessageExpunge

If the expunged message(s) is/are in the selected mailbox, the server notifies the client using EXPUNGE (or VANISHED, if [QRESYNC] is being used).

If the expunged message(s) is/are in another mailbox, the server sends an unsolicited STATUS (UIDNEXT MESSAGES) response for the relevant mailbox. If CONDSTORE is being used, HIGHESTMODSEQ MUST be included in the STATUS response.

Note that if a client requests MessageExpunge, the meaning of a MSN can change at any time, so the client cannot use MSNs in commands anymore. For example, such a client cannot use FETCH (it must only use UID FETCH). The meaning of '*' can also change when messages are added or expunged. A client wishing to keep using MSNs MUST NOT request the MessageExpunge event.

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

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:

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

## 5.4. QuotaExceed, QuotaWithin and QuotaChange

[[Alexey: I liked the following version more: If the client has permission to perform GETQUOTA (defined in [RFC2087]), the server sends an unsolicited QUOTA response containing the new quotas. ]] The server sends an unsolicited QUOTA response containing the new quotas. The server also sends an unsolicited QUOTAROOT response, so that the client can correlate the affected mailbox to the quota root.

These notifications are sent if the client has requested notifications for at least one affected mailbox.

Example:

```
    S: * QUOTAROOT INBOX ""
    S: * QUOTA "" (STORAGE 10 512)
```

In this example the quota root named "" (see [RFC2087] for the definition of quota root) governs the mailbox INBOX. Note that the server may return the QUOTAROOT and the QUOTA response in any order.

## 5.5. MailboxCreate and MailboxDelete

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

These notifications are sent if the client has requested notifications for at least one affected mailbox. In the case of MailboxCreate, the mailbox itself and its parent are considered to be affected. In the case of MailboxDelete, all deleted mailboxes and their parent(s) are considered to be affected.

Example of a newly created mailbox:

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

And a deleted mailbox:

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

## 5.6. MailboxRename

For each selectable mailbox renamed, the server sends an extended LIST response [LISTEXT] 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 MailboxRename events are sent for children in this case. When INBOX is renamed, a new INBOX is assumed to be created. No MailboxCreate event must be sent for INBOX in this case.

Example:

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

## 5.7. SubscriptionChange

The server notifies the client by sending an unsolicited LIST responses for each affected mailbox name. If and only if the mailbox is subscribed after the event, the \Subscribed attribute (see [LISTEXT]) is included.

Example:

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

## 5.8. MailboxMetadataChange

The server sends an unsolicited LIST response including METADATA. If

possible, only the changed metadata should be included, but if necessary, all metadata must be included.

Example:

```
    S: * LIST "/" "INBOX" (METADATA (/comment (value.priv "My
       comment")))
```

## 5.9. AdminMailbox

If the user has the right to perform GETACL (see [RFC4314]) after the event, the server notifies the client by sending an unsolicited ACL response with the mailbox' new rights.

If the user didn't have the right to perform GETACL, but later on such right was granted, then the server MUST send the ACL response to notify the client that it has access to the corresponding mailbox.

If the user loses the right to perform GETACL as a result of an ACL change, the server MUST NOT send the ACL response. Instead it MUST send an extended LIST response containing the \NonExistent mailbox attribute.

In all other cases, the server does not notify the client.

Example:

```
    S: * ACL INBOX Fred rwipslxcetda David lrswideta
```

## 5.10. 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...
```

## 5.11. ACL Changes

Even if NOTIFY succeeds, it is still possible to lose access to the mailboxes monitoried at a later time. If this happens, the server MUST silently stop monitoring these mailboxes. If access is later

granted, the server MUST restart event monitoring.

## 6. Mailbox Specification.

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

If the client specifies monitoring of the same mailbox several times, the first specification wins. A common example is asking for events on the selected mailbox and some named mailboxes.

In this example, the client asks for MessageExpunge events for all personal mailboxes except the selected mailbox:

```
    C: a notify set (selected (MessageNew (uid flags) flagchange))
       (personal (MessageNew (uid flags) flagchange MessageExpunge))
```

### 6.1. Selected

Selected refers to the mailbox selected using either SELECT or EXAMINE (see [RFC3501] section 6.3.1 and 6.3.2). When the IMAP connection is not in selected state, selected does not refer to any mailbox.

### 6.2. Personal

Personal refers to all selectable mailboxes in the user's personal namespace(s).

### 6.3. Inboxes

Inboxes refers to all selectable mailboxes in the user's personal namespace(s) to which messages may be delivered by an 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 by the user.

If the subscription list changes, the list MUST be reevaluated.

## 6.5 Subtree

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

[[Open Issue: Making this "all selectable mailboxes" makes it easy to implement this well. The pattern can be evaluated at NOTIFY time and notification information affixed to the mailboxes in RAM. Fine. But what about "notify me if any mailboxes are created whose name contains the letters xxx"? Not useful IMO...? (writes arnt)]]

## 6.6 Mailboxes

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

## 7. Formal Syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) notation as specified in [RFC4234]. [RFC3501] defines the non-terminals "capability", "command-auth", "mailbox", "mailbox-data", "resp-text-code" and "search-key".

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.

```
    capability      =/ "X-DRAFT-W00-NOTIFY"
                    ;; [[Note to RFC Editor: change the capability
                    ;; name before publication]]

    command-auth    =/ notify

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

    notify-add      = "ADD" [status-indicator] SP event-groups
                    ; Add (prepend) registered notification
                    ; events to the list of notification
                    ; events. Newer events override older
                    ; events.
                    [[Alexey: what about "most specific" event
```

```
                         overriding a pattern?]]

       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 ")"

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

       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-match-criteria = "(" search-key ")"

       message-event   = ( "MessageNew" SP
                             "(" fetch-att *(SP fetch-att) ")"
                           SP message-match-criteria )
                         / "MessageExpunge"
                         / "FlagChange" SP message-match-criteria
                         / "AnnotationChange" SP message-match-criteria
                         ;; "MessageNew" includes "MessageAppend" from
                         ;; [MSGEVENT]. "FlagChange" is any of
                         ;; "MessageRead", "MessageTrash", "FlagsSet",
                         ;; "FlagsClear" [MSGEVENT]. "MessageExpunge"
                         ;; includes "MessageExpire" [MSGEVENT].
```

```
        mailbox-event   = "MailboxCreate" / "MailboxDelete" /
                          "MailboxRename" /
                          "SubscriptionChange" / "MailboxMetadataChange"
                          / "QuotaChange" / "AdminMailbox"
                          ; "SubscriptionChange" includes
                          ; MailboxSubscribe and MailboxUnSubscribe

        user-event      = "QuotaExceed" / "QuotaWithin"

        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 and without surrounding quotes, as per
                          ;; mbox-list-extended-item-tag production.

        resp-text-code  =/ "NOTIFICATIONOVERFLOW"
```

## 8. 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 exists in plain IMAP, however this extension magnifies the problem.

This extensions 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] or similar is much better.

## 9. IANA considerations

The IANA is requested to add NOTIFY to the list of IMAP extensions, http://www.iana.org/assignments/imap4-capabilities.

### 9.1. Initial LIST-EXTENDED extended data item registrations

It is requested that the following entry be added to the LIST-EXTENDED extended data item registry [LISTEXT]:

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 XXXX, Section 5.6.

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

## 10. Acknowedgements

The authors gratefully acknowledge the help of Peter Coates, Dave Cridland, Mark Crispin, Cyrus Daboo and Abhijit Menon-Sen. Various example lines are copied from other RFCs.

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

## 11. Normative References

[RFC2087] Myers, "IMAP4 QUOTA extension", RFC 2087, January 1997.

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

[RFC2177] Leiba, "IMAP4 IDLE Command", RFC 2177, IBM, June 1997.

[RFC3501] Crispin, "Internet Message Access Protocol - Version

4rev1", RFC 3501, University of Washington, June 2003.

[RFC4234] Crocker, Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, Brandenburg Internetworking, Demon Internet Ltd, October 2005.

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

[RFC4466] Melnikov, Daboo, "Collected Extensions to IMAP4 ABNF", RFC 4466, Isode Ltd., April 2006.

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

[ANNOTATE] Gellens, Daboo, "IMAP ANNOTATE Extension", draft-ietf-imapext-annotate-16 (work in progress).

[LISTEXT] Leiba, Melnikov, "IMAP4 List Command Extensions", draft-ietf-imapext-list-extensions-18 (work in progress), IBM, September 2006.

[METADATA] Daboo, "IMAP METADATA Extension", draft-daboo-imap-annotatemore-11 (work in progress), Apple Computer, Inc., February 2007.

[MSGEVENT] Newman, "Internet Message Store Events", draft-ietf-lemonade-msgevent-03.txt (work in progress), Sun, July 2007.

## 12. Informative References

[SIEVE] Showalter, "Sieve: A Mail Filtering Language", RFC 3028, Mirapoint Inc, January 2001.

[QRESYNC] Melnikov, Cridland, Wilson, "IMAP4 Extensions for Quick Mailbox Resynchronization", draft-ietf-lemonade-reconnect-client-05.txt (work in progress), February 2007.

[EMAIL-ARCH] Crocker, "Internet Mail Architecture", draft-crocker-email-arch-09 (work in progress), March 2007.

## 13. Authors' Addresses

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

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

Email: arnt@oryx.com

## Intellectual Property Statement

## Full Copyright Statement

## Acknowledgment

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