Network Working Group
Internet-Draft
Expires: December 11, 2006

A. Melnikov
Isode Ltd

June 9, 2006

IMAP4 extension for reporting expunged messages
draft-melnikov-imap-expunged-01.txt

Status of this Memo

Copyright Notice

Abstract

This document defines an IMAP extension, which gives a disconnected client ability to quickly learn about expunged messages. This extension also introduces a new response that allows for a more compact representation for a list of expunged messages.

Table of Contents

1. Conventions Used in this Document

In examples, "C:" and "S:" indicate lines sent by the client and server respectively. If a single "C:" or "S:" label applies to multiple lines, then the line breaks between those lines are for editorial clarity only and are not part of the actual protocol exchange.

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].

Understanding of the IMAP message sequence numbers and UIDs and the EXPUNGE response [IMAP] is essential when reading this document.

[[anchor2: Editorial comments and questions are marked like this.]]

2. Introduction and Overview

The [CONDSTORE] extension gives a disconnected client ability to quickly synchronize flag changes for previously seen messages. In order for the client to discover which messages have been expunged, the client still has to issue a UID FETCH or a UID SEARCH command. This document defines an IMAP extension, that allows a client to quickly learn about expunged messages. This extension also introduces a new response EXPUNGED that allows for a more compact representation for a list of expunged messages.

The Expunged Messages Notification extension is present in any IMAP4 implementation which advertises "X-DRAFT-I01-EXPUNGED" [[anchor4: Change upon publication]] as one of the supported capabilities in the CAPABILITY command response.

3. IMAP Protocol Changes

3.1. REPORTEXPUNGES FETCH modifier

[IMAPABNF] has extended the syntax of the FETCH and UID FETCH commands to include an optional FETCH modifier. This document defines a new UID FETCH modifier (note, it is NOT allowed with a FETCH command. The server MUST return tagged BAD response if this response is specified as a modifier to the FETCH command [[anchor7: Should this be allowed instead and can be used as "please send me EXPUNGED" in the future flag?]]): REPORTEXPUNGES

The REPORTEXPUNGES FETCH modifier instructs the server to report all

messages from the UID set parameter to the UID FETCH command that were expunged. The expunged messages are reported using the EXPUNGED response as described in Section 3.2.

Example: The following example assumes that the server supports both CONDSTORE [CONDSTORE] and the extension defined in this document.

Without the REPORTEXPUNGES FETCH modifier a CONDSTORE-aware client [CONDSTORE] must issue two commands to learn about flag changes, as well as messages expunged since the last synchronization:

```
C: s100 UID FETCH 1:* (FLAGS) (CHANGEDSINCE 12345)
S: * 1 FETCH (UID 4 MODSEQ (65402) FLAGS (\Seen))
S: * 2 FETCH (UID 6 MODSEQ (75403) FLAGS (\Deleted))
S: * 4 FETCH (UID 8 MODSEQ (29738) FLAGS ($NoJunk
    $AutoJunk $MDNSent))
S: s100 OK FETCH completed
C: s101 UID SEARCH 1:*
S: * SEARCH 4 6 7 8 10 12
S: s101 OK search completed
```

The second SEARCH response tells the client that the messages with UIDs 7, 10 and 12 are still present, but their flags haven't changed since the specified modification sequence.

Using the REPORTEXPUNGES FETCH modifier it is sufficient to issue only a single command:

```
C: s100 UID FETCH 1:* (FLAGS) (CHANGEDSINCE 12345
    REPORTEXPUNGES)
S: * 1 FETCH (UID 4 MODSEQ (65402) FLAGS (\Seen))
S: * 2 FETCH (UID 6 MODSEQ (75403) FLAGS (\Deleted))
S: * 4 FETCH (UID 8 MODSEQ (29738) FLAGS ($NoJunk
    $AutoJunk $MDNSent))
S: * EXPUNGED 1:3,5,9,11
S: s100 OK FETCH completed
```

3.2. EXPUNGED Response

Contents: list of UIDs

The EXPUNGED response reports that the specified UIDs have been permanently removed from the mailbox. This response is similar to the EXPUNGE response [RFC3501], however it can return information about multiple messages and it returns UIDs, instead of message numbers. The former allows to save bandwidth, while the latter is more convenient for clients which only use UIDs to access the IMAP server.

The EXPUNGED response is sent as a result of UID FETCH (REPORTEXPUNGES) command, if the UID set parameter to the UID FETCH (REPORTEXPUNGES) command includes UIDs of messages that are no longer in the mailbox. The EXPUNGED response SHOULD also be sent by the server instead of the EXPUNGE response, once the client has indicated that it supports the extension described in this document by issuing the UID FETCH (REPORTEXPUNGES) command on the connection. In particular this affects the EXPUNGE [RFC3501] and UID EXPUNGE [UIDPLUS] commands, as well as messages expunged in other sessions.

The EXPUNGED response caused by EXPUNGE/UID EXPUNGE/messages expunged in other sessions also decrements the number of messages in the mailbox; it is not necessary for the server to send an EXISTS and/or RECENT response with the new value. It also decrements message sequence numbers for each successive message in the mailbox (see Example at the end of this section).

An EXPUNGED response MUST NOT be sent when no command is in progress, nor while responding to a FETCH, STORE, or SEARCH command. This rule is necessary to prevent a loss of synchronization of message sequence numbers between client and server. A command is not "in progress" until the complete command has been received; in particular, a command is not "in progress" during the negotiation of command continuation.

Note: UID FETCH, UID STORE, and UID SEARCH are different commands from FETCH, STORE, and SEARCH. An EXPUNGED response MAY be sent during a UID command.

The update from the EXPUNGED response MUST be recorded by the client.

Example: Let's assume that there is the following mapping between message numbers and UIDs in the currently selected mailbox (here "X" marks messages with the \Deleted flag set, and "x" represents UIDs which are not relevant for the example):

```
Message numbers:   1  2  3  4  5  6  7  8  9 10 11
UIDs:              x  x  5  7  x  x 10  x  x  x 25
\Deleted messaged:       X  X        X           X
```

In the presence of the extension defined in this document:

```
C: A202 EXPUNGE
S: * EXPUNGED 5,7,10,25
S: A202 OK EXPUNGE completed
```

Without the X-DRAFT-I01-EXPUNGED [[anchor8: fix upon publication]] extension the same example can look like:

```
C: A202 EXPUNGE
S: * 3 EXPUNGE
S: * 3 EXPUNGE
S: * 5 EXPUNGE
S: * 8 EXPUNGE
S: A202 OK EXPUNGE completed
```

4. Updated synchronization sequence

This section updates the description of optimized synchronization in section 6.1 of the [IMAP-DISC].

An advanced disconnected mail client should use the EXPUNGED and [CONDSTORE] extensions when they are supported by the server. The client MUST cache the value from HIGHESTMODSEQ OK response code received on mailbox opening and update it whenever the server sends MODSEQ FETCH data items.

If the client receives NOMODSEQ OK untagged response instead of HIGHESTMODSEQ, it MUST remove the last known HIGHESTMODSEQ value from its cache and follow more general instructions in section 3 of the [IMAP-DISC].

When the client opens the mailbox for synchronization it first compares UIDVALIDITY as described in step d)1) in section 3 of the [IMAP-DISC]. If the cached UIDVALIDITY value matches the one returned by the server, the client MUST compare the cached value of HIGHESTMODSEQ with the one returned by the server. If the cached HIGHESTMODSEQ value also matches the one returned by the server, then the client SHOULD NOT fetch flags for cached messages, as they haven't changed. If the value on the server is higher than the cached one, the client MAY use "SEARCH MODSEQ <cached-value>" to find all messages with flags changed since the last time the client was online and had the mailbox opened. Alternatively the client MAY use "FETCH 1:* (FLAGS) (CHANGEDSINCE <cached-value> REPORTEXPUNGES)". The latter operation combines reporting expunged messages, searching for changed messages and fetching new information.

In all cases the client still needs to fetch information about new messages (if requested by the user). If the client has used SEARCH MODSEQ, it will also have to discover which messages have been expunged.

Step d) ("Server-to-client synchronization") in section 4 of the

[IMAP-DISC] in the presence of the EXPUNGED & CONDSTORE extensions is amended as follows:

d) "Server-to-client synchronization" - for each mailbox that requires synchronization, do the following:

1a) Check the mailbox UIDVALIDITY (see section 4.1 of the [IMAP-DISC] for more details) with SELECT/EXAMINE/STATUS. If the UIDVALIDITY value returned by the server differs, the client MUST

* empty the local cache of that mailbox;

* "forget" the cached HIGHESTMODSEQ value for the mailbox;

* remove any pending "actions" which refer to UIDs in that mailbox. Note, this doesn't affect actions performed on client generated fake UIDs (see section 5 of the [IMAP-DISC]);

* skip steps 1b and 2-II;

1b) Check the mailbox HIGHESTMODSEQ. If the cached value is the same as the one returned by the server, skip fetching message flags on step 2-II, i.e. the client only has to find out which messages got expunged.

2) Fetch the current "descriptors";

I) Discover new messages.

II) Discover changes to old messages and expunged messages using "UID FETCH 1:<lastseenuid> (FLAGS) (CHANGEDSINCE <cached-value> REPORTEXPUNGES)".

(Note, if <lastseenuid> is replaced with "*", this command will return flags for new messages as well)

3) Fetch the bodies of any "interesting" messages that the client doesn't already have.

Example: The UIDVALIDITY value is the same, but the HIGHESTMODSEQ value has changed on the server while the client was offline:

```
C: A142 SELECT INBOX
S: * 172 EXISTS
S: * 1 RECENT
S: * OK [UNSEEN 12] Message 12 is first unseen
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: * OK [UIDNEXT 201] Predicted next UID
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Deleted \Seen \*)] Limited
S: * OK [HIGHESTMODSEQ 20010715194045007]
S: A142 OK [READ-WRITE] SELECT completed

           after that:

C: A143 UID FETCH 1:20 (FLAGS)
    (CHANGEDSINCE 20010715194032001 REPORTEXPUNGES)
S: * 2 FETCH (UID 6 MODSEQ (20010715205008000)
    FLAGS (\Deleted))
S: * 5 FETCH (UID 9 MODSEQ (20010715195517000)
    FLAGS ($NoJunk $AutoJunk $MDNSent))
   ...
S: * EXPUNGED 1:5,7:8,10:15
S: A143 OK FETCH completed
```

5. Formal Syntax

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

Non-terminals referenced but not defined below are as defined by [RFC3501], or [IMAPABNF].

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-I01-EXPUNGED"
                       ;; [[Note to RFC Editor: fix before
                       ;; publication]]

message-data        =/ expunged-resp
```

```
expunged-resp      =  "EXPUNGED" SP known-uids

known-uids         =  sequence-set
                      ;; sequence of UIDs, "*" is not allowed

rexpunges-fetch-mod = "REPORTEXPUNGES"
                      ;; REPORTEXPUNGES FETCH modifier conforms
                      ;; to the fetch-modifier syntax
                      ;; defined in [IMAPABNF].  It is only
                      ;; allowed in the UID FETCH command.
```

6. Security Considerations

It is believed that this extension doesn't raise any additional security concerns not already discussed in [RFC3501].

As always, it is important to thoroughly test clients and servers implementing this extension, as it changes how the server reports expunged messages to the client.

7. IANA Considerations

IMAP4 capabilities are registered by publishing a standards track or IESG approved experimental RFC. The registry is currently located at:

http://www.iana.org/assignments/imap4-capabilities

This document defines the X-DRAFT-I01-EXPUNGED [[anchor13: Note to RFC Editor: fix before publication]] IMAP capability. IANA is requested to add it to the registry.

8. Acknowledgments

Thanks to Steve Hole, Cyrus Daboo, David Cridland and Michael Wener for encouraging me to write this document.

Thanks to David Cridland, Timo Sirainen and Michael Wener for comments and corrections.

This document takes substantial text from [RFC3501] by Mark Crispin.

9. References

9.1. Normative References

[ABNF] Crocker, D., Ed. and P. Overell, Ed., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

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

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

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

[UIDPLUS] Crispin, M., "Internet Message Access Protocol (IMAP) - UIDPLUS extension", RFC 4315, December 2005.

9.2. Informative References

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

[IMAP-DISC] Melnikov, A., "Synchronization Operations For Disconnected Imap4 Clients", RFC 4549, June 2006.

Author's Address

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

Email: Alexey.Melnikov@isode.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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