Stream:	Internet Engineering Task Force (IETF)			
RFC:	9738			
Category:	Experimental			
Published:	February 2025			
ISSN:	2070-1721			
Authors:	A. Melnikov *Isode*	A. P. Achuthan *Yahoo!*	V. Nagulakonda *Yahoo!*	L. Alves

RFC 9738
IMAP MESSAGELIMIT Extension

Abstract

The MESSAGELIMIT extension of the Internet Message Access Protocol (RFC 3501/RFC 9051) allows servers to announce a limit on the number of messages that can be processed in a single FETCH, SEARCH, STORE, COPY, or MOVE command (or their UID variants), or in a single APPEND or UID EXPUNGE command. This helps servers to control resource usage when performing various IMAP operations. This helps clients to know the message limit enforced by the corresponding IMAP server and avoid issuing commands that would exceed such limit.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9738.

Copyright Notice

Table of Contents

1. Introduction and Overview

This document defines an extension to the Internet Message Access Protocol [RFC3501] for announcing a server limit on the number of messages that can be processed in a single FETCH, SEARCH, STORE, COPY, or MOVE command (or their UID variants), or a single APPEND or UID EXPUNGE command. This extension is compatible with both IMAP4rev1 [RFC3501] and IMAP4rev2 [RFC9051].

2. Document Conventions

In protocol examples, this document uses a prefix of "C: " to denote lines sent by the client to the server, and "S: " for lines sent by the server to the client. Lines prefixed with "// " are comments explaining the previous protocol line. These prefixes and comments are not part of the protocol. Lines without any of these prefixes are continuations of the previous line, and no line break is present in the protocol unless specifically mentioned.

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Other capitalized words are IMAP key words [RFC3501][RFC9051] or key words from this document.

3. The MESSAGELIMIT Extension

An IMAP server advertises support for the MESSAGELIMIT extension by including "MESSAGELIMIT=<limit>" capability in the CAPABILITY response or response code, where "<limit>" is a positive integer that conveys the maximum number of messages that can be processed in a single SEARCH, FETCH, STORE, COPY, MOVE command (or their UID variants), or in a single APPEND or UID EXPUNGE command.

An IMAP server that only enforces the message limit on COPY and APPEND commands (and their UID variants) would include the "SAVELIMIT=<limit>" capability (instead of the "MESSAGELIMIT=<limit>") in the CAPABILITY response or response code.

The limit advertised in the MESSAGELIMIT or SAVELIMIT capability **SHOULD NOT** be lower than 1000 messages.

3.1. Returning Limits on the Number of Messages Processed in a Single SEARCH/FETCH/STORE/COPY/MOVE/APPEND/EXPUNGE Command

If a server implementation doesn't allow more than <N> messages to be operated on by a single COPY or UID COPY command, it **MUST** fail the command by returning a tagged NO response with the MESSAGELIMIT response code defined below. No messages are copied in this case. If a server implementation doesn't allow more than <N> messages to be operated on by a single SEARCH, FETCH, STORE, or MOVE command (or their UID variants), or an APPEND or UID EXPUNGE command, it **MUST** return the MESSAGELIMIT response code defined below:

MESSAGELIMIT
: The server doesn't allow more than <N> messages to be operated on by a single SEARCH, FETCH, STORE, COPY, or MOVE command (or their UID variants). The lowest processed UID is <LastUID>. The client needs to repeat the operation for remaining messages, if required.

  The server doesn't allow more than <N> \Deleted messages to be operated on by a single UID EXPUNGE command. The lowest processed UID is <LastUID>. The client needs to repeat the operation for remaining messages, if required.

  Note that when the MESSAGELIMIT response code is returned, the server is **REQUIRED** to process messages from highest to lowest UID.

  Note that the MESSAGELIMIT response code is similar to the LIMIT response code [RFC9051], but it provides more details on the exact type of the limit and how to resume the command once the limit is exceeded.

  In the following example, the <N> value is 1000, and the lowest processed UID <LastUID> is 23221.

```
C: 03 FETCH 10000:14589 (UID FLAGS)
S: * 14589 FETCH (FLAGS (\Seen) UID 25000)
S: * 14588 FETCH (FLAGS (\Answered) UID 24998)
S: ... further 997 fetch responses
S: * 13590 FETCH (FLAGS () UID 23221)
S: 03 OK [MESSAGELIMIT 1000 23221] FETCH completed with 1000
    partial results
```

  In the following example the client searches for UNDELETED UIDs between 22000:25000. The total number of searched messages (note, NOT the number of matched messages) exceeds the server's published 1000-message limit.

```
C: 04 UID SEARCH UID 22000:25000 UNDELETED
S: * SEARCH 25000 24998 (... UIDs ...) 23221
S: 04 OK [MESSAGELIMIT 1000 23221] SEARCH completed with 1000
    partial results
```

The following example demonstrates the copy of messages with UIDs between 18000:21000. The total message count exceeds the server's published 1000-message limit. As COPY or UID COPY needs to be atomic (as per [RFC3501]/[RFC9051]), no messages are copied.

```
C: 05 UID COPY 18000:21000 "Trash"
S: 05 NO [MESSAGELIMIT 1000 20001] Too many messages to copy,
    try a smaller subset
```

The following example shows the move of messages with UIDs between 18000:21000. The total message count exceeds the server's published 1000-message limit. (Unlike COPY or UID COPY, MOVE or UID MOVE don't need to be atomic.) The client that wants to move all messages in the range and observes a MESSAGELIMIT response code can repeat the UID MOVE command with the same parameter. (For the MOVE command, the message set parameter needs to be updated before repeating the command.) The client needs to keep doing this until the MESSAGELIMIT response is not returned (or until a tagged NO or BAD is returned).

```
C: 06 UID MOVE 18000:21000 "Archive/2021/2021-12"
S: * OK [COPYUID 1397597919 20001:21000 22363:23362] Some
    messages were not moved
S: * 12336 EXPUNGE
S: * 12335 EXPUNGE
...
S: * 11337 EXPUNGE
S: 06 OK [MESSAGELIMIT 1000 20001] MOVE completed for the last
    1000 messages
```

The following example shows the update of flags for messages with UIDs between 18000:20000. The total number of existing messages in the UID range exceeds the server's published 1000-message limit. The client that wants to change flags for all messages in the range and observes a MESSAGELIMIT response code can repeat the UID STORE command with the updated UID range that doesn't include the UID returned in the MESSAGELIMIT response code. (For the STORE command, the message set parameter also needs to be updated before repeating the command.) The client needs to keep doing this until the MESSAGELIMIT response is not returned (or until a tagged NO or BAD is returned).

```
C: 07 UID STORE 18000:20000 +FLAGS (\Seen)
S: * 11215 FETCH (FLAGS (\Seen \Deleted) UID 20000)
S: * 11214 FETCH (FLAGS (\Seen \Answered \Deleted) UID 19998)
...
S: * 10216 FETCH (FLAGS (\Seen) UID 19578)
S: 07 OK [MESSAGELIMIT 1000 19578] STORE completed for the last
    1000 messages
```

The following example shows the removal of messages (using UID EXPUNGE) that have the \Deleted flag set with UIDs between 11000:13000. The total message count of messages with the \Deleted flag set exceeds the server's published 1000-message limit. The client that wants

to remove all messages marked as \Deleted in the range and observes a MESSAGELIMIT response code can repeat the UID EXPUNGE command with the same parameter. The client needs to keep doing this until the MESSAGELIMIT response is not returned (or until a tagged NO or BAD is returned).

```
C: 08 UID EXPUNGE 11000:13000
S: * 4306 EXPUNGE
S: * 4305 EXPUNGE
...
S: * 3307 EXPUNGE
S: 08 OK [MESSAGELIMIT 1000 11627] UID EXPUNGE completed for
    the last 1000 messages
```

The following example shows removal of messages (using EXPUNGE) that have the \Deleted flag set. Unlike UID EXPUNGE, the server **MUST NOT** impose any message limit when processing EXPUNGE.

```
C: 09 EXPUNGE
S: * 4306 EXPUNGE
S: * 4305 EXPUNGE
...
S: * 3307 EXPUNGE
S: * 112 EXPUNGE
S: 09 OK EXPUNGE completed
```

Similarly, the server **MUST NOT** impose any message limit when processing a "CLOSE" or a "STATUS UNSEEN" command.

The following example shows use of the MESSAGELIMIT response code together with the PARTIAL [RFC9394] extension. The total message count (as specified by the PARTIAL range) exceeds the server's published 1000-message limit, so the server refuses to do any work in this case.

```
C: 10 UID FETCH 22000:25000 (UID FLAGS MODSEQ)
    (PARTIAL -1:-1500)
S: 10 NO [MESSAGELIMIT 1000] FETCH exceeds the maximum 1000-
    message limit
```

Without the PARTIAL parameter, the above UID FETCH can look like this:

```
C: 10 UID FETCH 22000:25000 (UID FLAGS MODSEQ)
S: * 12367 FETCH (FLAGS (\Seen \Deleted) UID 23007)
S: * 12366 FETCH (FLAGS (\Seen \Answered \Deleted) UID 23114)
...
S: * 13366 FETCH (FLAGS (\Seen) UID 24598)
S: 10 OK [MESSAGELIMIT 1000 23007] FETCH exceeds the maximum
    1000-message limit
```

Note that when the server needs to return both EXPUNGEISSUED [RFC9051] and MESSAGELIMIT response codes, the former **MUST** be returned in the tagged OK response, while the latter **MUST** be returned in an untagged NO response. The following example demonstrates that:

```
C: 11 FETCH 10000:14589 (UID FLAGS)
S: * 14589 FETCH (FLAGS (\Seen) UID 25000)
S: * 14588 FETCH (FLAGS (\Answered) UID 24998)
S: ... further 997 fetch responses
S: * 13590 FETCH (FLAGS () UID 23221)
S: * NO [MESSAGELIMIT 1000 23221] FETCH completed with 1000 partial
    results
S: 11 OK [EXPUNGEISSUED] Some messages were also expunged
```

When the IMAP MULTIAPPEND extension [RFC3502] is also supported by the server, the message limit also applies to the APPEND command. As MULTIAPPEND APPEND needs to atomic (as per [RFC3502]), no messages are appended when the server MESSAGELIMIT is exceeded.

3.2. UIDAFTER and UIDBEFORE SEARCH Criteria

The MESSAGELIMIT extension also defines two extra SEARCH keys, UIDAFTER and UIDBEFORE, which make it easier to convert a single UID to a range of UIDs.

"UIDAFTER <uid>" Messages that have a UID greater than the specified UID. This is semantically the same as "UID <uid>+1:*".

"UIDBEFORE <uid>" Messages that have a UID less than the specified UID. This is semantically the same as "UID 1:<uid>-1" (or if <uid> has the value 1, then the empty set).

These two SEARCH keys are particularly useful when the SEARCHRES extension [RFC5182] is also supported, but they can be used without it. For example, this allows a SEARCH that sets the "$" marker to be converted to a range of messages in a subsequent SEARCH, and both SEARCH requests can be pipelined.

```
C: 12 UID SEARCH UIDAFTER 25000 UNDELETED
S: * SEARCH 27800 27798 (... 250 UIDs ...) 25001
S: 12 OK SEARCH completed
```

3.3. Interaction with SORT and THREAD Extensions

Servers that advertise MESSAGELIMIT N will be unable to execute a THREAD command [RFC5256] in a mailbox with more than N messages.

Servers that advertise MESSAGELIMIT N might be unable to execute a SORT command [RFC5256] in a mailbox with more than N messages, unless they maintain indices for different SORT orders they support. In absence of such indices, server implementors will need to decide whether their server advertises SORT or MESSAGELIMIT capability.

3.4. Interaction with SEARCHRES Extension and IMAP4rev2

Servers that support both MESSAGELIMIT and SEARCHRES extensions [RFC5182] **MUST** truncate SEARCH SAVE result stored in the $ variable when the SEARCH command succeeds, but the MESSAGELIMIT response code is returned. For example, if the following SEARCH would have returned 1200 results in absence of MESSAGELIMIT, and the MESSAGELIMIT is 1000, only 1000 matching results will be saved in the $ variable:

```
C: D0004 UID SEARCH RETURN (SAVE) SINCE 1-Jan-2004 NOT FROM "Smith"
    UID 22000:25000 UNDELETED
S: D0004 OK [MESSAGELIMIT 1000 1179] SEARCH completed with 1000
    partial results saved
```

4. Interoperability Considerations

4.1. Effects of MESSAGELIMIT and SAVELIMIT Extensions on Noncompliant Clients

A server that advertises the MESSAGELIMIT=N capability would have the following effect on clients that don't support this capability:

- Operations on a mailbox that has <= N messages are not affected.
- In a mailbox with more than N messages:
  - An attempt to COPY or UID COPY more than N messages will always fail.
  - EXPUNGE and CLOSE will always operate on the full mailbox, so they are not affected.
  - Other commands like FETCH, SEARCH, and MOVE will be effectively restricted to the last N messages of the mailbox. In particular, unextended SEARCHes (intended for counting of messages with or without a particular set of flags) would return incorrect counts.

4.2. Maintaining Compatibility

It is important to understand that the above effects essentially abandon existing clients with respect to operation on large mailboxes. Suppose, for example, that a user is accessing a large and active mailing list via IMAP, and the mailing list gets on the order of 1500 posts per week. When the user returns from a week-long vacation and catches up on the mailing list, the user's client will be fetching information about 1500 messages. If the server has a MESSAGELIMIT of 1000, the client will only be able to download 1000 of the most recent messages; the client will not understand why, will not be prepared to recover from the situation, and will act as if it is interacting with a broken server.

In order to give clients time to implement this extension, servers should not be strict about applying the MESSAGELIMIT at first. One possible approach is to advertise a MESSAGELIMIT but not enforce it at all for a while. Clients that understand this extension will comply, reducing load on the server, but clients that do not understand the limit will continue to work in all situations.

Another approach, which perhaps could be phased in later, is to advertise one limit but to treat it as a soft limit and to begin enforcement at a higher, unadvertised hard limit. In the above example, perhaps the server might advertise 1000 but actually enforce a limit of 10,000. Again, clients that understand MESSAGELIMIT will comply with the limit of 1000, but other clients will still interoperate up to the higher threshold.

Attempts to go beyond the advertised limit can be logged so that client understanding of MESSAGELIMIT can be tracked. If implementation and deployment of this extension becomes common, it may at some point be acceptable to strictly enforce the advertised limit and to accept that the remaining clients will, indeed, no longer work properly with mailboxes above that limit.

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 IMAP4 [RFC3501].

Except as noted otherwise, all alphabetic characters are case-insensitive. The use of uppercase or lowercase characters to define token strings is for editorial clarity only. Implementations **MUST** accept these strings in a case-insensitive fashion.

```
capability          =/ "MESSAGELIMIT=" message-limit /
                       "SAVELIMIT=" message-limit
                       ;; <capability> from [RFC3501]

message-limit       = nz-number

resp-text-code      =/ "MESSAGELIMIT" SP message-limit [SP uniqueid]
    ;; No more than nz-number messages can be processed
    ;; by any command at a time. The last (lowest) processed
    ;; UID is uniqueid.
    ;; The last parameter is omitted when not known.
```

6. Security Considerations

This document defines an additional IMAP4 capability. As such, it does not change the underlying security considerations of IMAP4rev1 [RFC3501] or IMAP4rev2 [RFC9051].

This document defines an optimization that can both reduce the amount of work performed by the server, as well at the amount of data returned to the client. Use of this extension is likely to cause the server and the client to use less memory than when the extension is not used, which can in turn help to protect from denial-of-service attacks. However, as this is going to be new code in both the client and the server, rigorous testing of such code is required in order to avoid introducing new implementation bugs.

7. IANA Considerations

7.1. Additions to the IMAP Capabilities Registry

IMAP4 capabilities are registered by publishing a Standards Track or IESG-approved Informational or Experimental RFC. The "IMAP Capabilities" registry is currently located at: <https://www.iana.org/assignments/imap-capabilities/>.

IANA has added "MESSAGELIMIT=" and "SAVELIMIT=" capabilities to this registry, with this document as the reference.

8. References

8.1. Normative References

[ABNF] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, DOI 10.17487/RFC3501, March 2003, <https://www.rfc-editor.org/info/rfc3501>.

[RFC3502] Crispin, M., "Internet Message Access Protocol (IMAP) - MULTIAPPEND Extension", RFC 3502, DOI 10.17487/RFC3502, March 2003, <https://www.rfc-editor.org/info/rfc3502>.

[RFC5182] Melnikov, A., "IMAP Extension for Referencing the Last SEARCH Result", RFC 5182, DOI 10.17487/RFC5182, March 2008, <https://www.rfc-editor.org/info/rfc5182>.

[RFC5256] Crispin, M. and K. Murchison, "Internet Message Access Protocol - SORT and THREAD Extensions", RFC 5256, DOI 10.17487/RFC5256, June 2008, <https://www.rfc-editor.org/info/rfc5256>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC9051] Melnikov, A., Ed. and B. Leiba, Ed., "Internet Message Access Protocol (IMAP) - Version 4rev2", RFC 9051, DOI 10.17487/RFC9051, August 2021, <https://www.rfc-editor.org/info/rfc9051>.

8.2. Informative References

[RFC9394] Melnikov, A., Achuthan, A. P., Nagulakonda, V., and L. Alves, "IMAP PARTIAL Extension for Paged SEARCH and FETCH", RFC 9394, DOI 10.17487/RFC9394, June 2023, <https://www.rfc-editor.org/info/rfc9394>.

Acknowledgments

This document was motivated by the Yahoo! team and their questions about best client practices for dealing with large mailboxes.

The editor of this document would like to thank the following people who provided useful comments, contributed text, or participated in discussions of this document: Timo Sirainen, Barry Leiba, Ken Murchison, and Arnt Gulbrandsen.

Authors' Addresses

Alexey Melnikov
Isode Limited
Email: alexey.melnikov@isode.com
URI: https://www.isode.com

Arun Prakash Achuthan
Yahoo!
Email: arunprakash@myyahoo.com

Vikram Nagulakonda
Yahoo!
Email: nvikram_imap@yahoo.com

Luis Alves
Email: luis.alves@lafaspot.com