Authors: A. Melnikov   A. P. Achuthan   V. Nagulakonda
         Isode   Yahoo!   Yahoo!
         L. Alves

IMAP Paged SEARCH & FETCH Extension

Abstract

The PARTIAL extension of the Internet Message Access Protocol (RFC 3501/RFC 9051) allows clients to limit the number of search results returned, as well as to perform incremental (paged) searches. This also helps servers to optimize resource usage when performing searches.

This document extends PARTIAL SEARCH return option originally specified in RFC 5267. It also clarifies some interactions between RFC 5267 and RFC 4731/RFC 9051.

This document also describes the MESSAGELIMIT extension for announcing a limit on the number of messages that can be processed in a single FETCH/SEARCH/STORE/COPY/MOVE command.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 11 November 2022.

Copyright Notice

Table of Contents

1. Introduction and Overview

This document defines an extension to the Internet Message Access Protocol [RFC3501] for performing incremental searches and fetches. This extension is compatible with both IMAP4rev1 [RFC3501] and IMAP4rev2 [RFC9051].

The PARTIAL extension of the Internet Message Access Protocol (RFC 3501/RFC 9051) allows clients to limit the number of search results returned, as well as to perform incremental (paged) searches. This also helps servers to optimize resource usage when performing searches.

This document extends PARTIAL SEARCH return option originally specified in RFC 5267. It also clarifies some interactions between RFC 5267 and RFC 4731/RFC 9051.

This document also describes the MESSAGELIMIT extension for announcing a limit on the number of messages that can be processed in a single FETCH/SEARCH/STORE/COPY/MOVE command.

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 capitalised words are IMAP keywords [RFC3501][RFC9051] or keywords from this document.

3. The PARTIAL extension

An IMAP server advertises support for the PARTIAL extension by including "PARTIAL" capability in the CAPABILITY response/response code.

Clients that implement support for PARTIAL extension MUST also support the MESSAGELIMIT response code (see Section 4).

3.1. Incremental SEARCH and partial results

The PARTIAL search return option causes the server to provide in an
ESEARCH response a subset of the results denoted by the sequence
range given as the mandatory argument. The first result (message
with the lowest matching UID) is 1; thus, the first 500 results
would be obtained by a return option of "PARTIAL 1:500", and the
second 500 by "PARTIAL 501:1000". This intentionally mirrors message
sequence numbers.

It is also possible to direct the server to start SEARCH from the
latest matching (with the highest UID) message. This can be done by
prepeding "-" to the index. For example -1 is the last message, -2
is next to the last and so on. Using this syntax helps server
implementations to optimize their SEARCHes.

A single command MUST NOT contain more than one PARTIAL or ALL
search return option -- that is, either one PARTIAL, one ALL, or
neither PARTIAL nor ALL is allowed.

For SEARCH results, the entire result list MUST be ordered in
mailbox order, that is, in UID or message sequence number order.

Where a PARTIAL search return option references results that do not
exist, by using a range which starts or ends higher (or lower) than
the current number of results, then the server returns the results
that are in the set. This yields a PARTIAL return data item that
has, as payload, the original range and a potentially missing set of
results that may be shorter than the extent of the range. If the
whole range references results that do not exist, a special value
"NIL" is returned by the server instead of the sequence set.

Clients need not request PARTIAL results in any particular order.
Because mailboxes may change, clients might wish to use PARTIAL in
combination with UPDATE (see [RFC5267] if the server also advertises
CONTEXT=SEARCH capability, especially if the intent is to walk a
large set of results; however, these return options do not interact
-- the UPDATE will provide notifications for all matching results.

```
// Let's assume that the A01 SEARCH without PARTIAL would return
// 23764 results.
C: A01 UID SEARCH RETURN (PARTIAL -1:-100) UNDELETED
   UNKEYWORD $Junk
S: * ESEARCH (TAG "A01") UID PARTIAL (-1:-100 ...)
// 100 most recent results in set syntax elided.
S: A01 OK Completed.
```

```
// Let's assume that the A02 SEARCH without PARTIAL would return
// 23764 results.
C: A02 UID SEARCH RETURN (PARTIAL 23500:24000) UNDELETED
   UNKEYWORD $Junk
C: A03 UID SEARCH RETURN (PARTIAL 1:500) UNDELETED
   UNKEYWORD $Junk
C: A04 UID SEARCH RETURN (PARTIAL 24000:24500) UNDELETED
   UNKEYWORD $Junk
S: * ESEARCH (TAG "A02") UID PARTIAL (23500:24000 ...)
// 264 results in set syntax elided,
// this spans the end of the results.
S: A02 OK Completed.
S: * ESEARCH (TAG "A03") UID PARTIAL (1:500 ...)
// 500 results in set syntax elided.
S: A03 OK Completed.
S: * ESEARCH (TAG "A04") UID PARTIAL (24000:24500 NIL)
// No results are present, this is beyond the end of the results.
S: A04 OK Completed.
```

3.2. Interaction between PARTIAL, MIN, MAX and SAVE SEARCH return options

This section only applies if the server advertises PARTIAL IMAP
capability or CONTEXT=SEARCH [RFC5267], together with ESEARCH
[RFC4731] and/or IMAP4rev2"[RFC9051].

The SAVE result option doesn't change whether the server would
return items corresponding to PARTIAL SEARCH result options.

As specified in Section 3.1, it is an error to specify both PARTIAL
and ALL result options in the same SEARCH command.

When the SAVE result option is combined with the PARTIAL result
option, and none of MIN/MAX/COUNT result options is present, the
corresponding PARTIAL is returned, and the "$" marker would contain
all messages returned by the PARTIAL result option.

When the SAVE + PARTIAL result options are combined with the MIN or
the MAX result option, and the COUNT result option is absent, the
corresponding PARTIAL result and MIN/MAX is returned (if the search
result is not empty), and the "$" marker would contain all messages
returned by the PARTIAL result option + the corresponding MIN/MAX
message.

If the SAVE + PARTIAL result options are combined with both MIN and
MAX result options, and the COUNT result options is absent, the
PARTIAL, MIN and MAX are returned (if the search result is not
empty), and the "$" marker would contain all messages returned by
the PARTIAL result option plus MIN and MAX messages.

If the SAVE + PARTIAL result options are combined with the COUNT
result option, the PARTIAL and COUNT are returned, and the "$"
marker would always contain all messages found by the SEARCH or UID
SEARCH command.

The following table summarizes the additional requirement on ESEARCH
server implementations described in this section.

Combination of Result option	"$" marker value
SAVE PARTIAL	PARTIAL
SAVE PARTIAL MIN	PARTIAL & MIN
SAVE PARTIAL MAX	PARTIAL & MAX
SAVE PARTIAL MIN MAX	PARTIAL & MIN & MAX
SAVE PARTIAL COUNT [m]	all found messages

Table 1

where '[m]' means optional "MIN" and/or "MAX"

3.3. Extension to UID FETCH

The PARTIAL extension also extends the UID FETCH command with a
PARTIAL FETCH modifier. The PARTIAL FETCH modifier has the same
syntax as the PARTIAL SEARCH result option. Presence of the PARTIAL
FETCH modifier instructs the server to only return FETCH results for
messages in the specified range. It is useful when the sequence-set
(first) parameter to the UID FETCH command includes unknown number
of messages.

```
// Returning information for the last 3 messages in the UID range
C: 10 UID FETCH 25900:26600 (UID FLAGS) (PARTIAL -1:-3)
S: * 12888 FETCH (FLAGS (\Seen) UID 25996)
S: * 12889 FETCH (FLAGS (\Flagged \Answered) UID 25997)
S: * 12890 FETCH (FLAGS () UID 26600)
S: 10 OK FETCH completed

// Returning information for the first 5 messages in the UID range
C: 11 UID FETCH 25900:26600 (UID FLAGS) (PARTIAL 1:5)
S: * 12591 FETCH (FLAGS (\Seen) UID 25900)
S: * 12592 FETCH (FLAGS (\Flagged) UID 25902)
S: * 12593 FETCH (FLAGS (\Answered) UID 26310)
S: * 12594 FETCH (FLAGS () UID 26311)
S: * 12595 FETCH (FLAGS (\Answered) UID 26498)
S: 11 OK FETCH completed
```

3.4. Use of PARTIAL and CONDSTORE IMAP extensions together

This section is informative.

The PARTIAL FETCH modifier can be combined with the CHANGEDSINCE
FETCH modifier.

```
// Returning information for the last 30 messages in the UID range
// that have any flag/keyword modified since modseq 98305
C: 101 UID FETCH 25900:26600 (UID FLAGS) (PARTIAL -1:-30 CHANGEDSINCE
S: * 12888 FETCH (FLAGS (\Flagged \Answered) MODSEQ (98306) UID 25997)
S: * 12890 FETCH (FLAGS () MODSEQ (98312) UID 26600)
S: 101 OK FETCH completed
```

The above example causes the server to first select the last 30
messages and then only return flag changes for subset of these
messages which have MODSEQ higher than 98305.

Note that the order of PARTIAL and CHANGEDSINCE FETCH modifiers in
the UID FETCH command is not important, i.e. the above example can
also use "UID FETCH 25900:26600 (UID FLAGS) (CHANGEDSINCE 98305
PARTIAL -1:-30)" command and it would result in the same responses.

4. The MESSAGELIMIT extension

An IMAP server advertises support for the MESSAGELIMIT extension by
including "MESSAGELIMIT=<limit>" capability in the CAPABILITY
response/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.

4.1. Returning limits on the number of messages processed in a single SEARCH/FETCH/STORE/COPY/MOVE command

Do we need a way to specify SEARCH criterion for "all UIDs after" or
"all UIDs before" a specific UID?

If a server implementation doesn't allow more than <N> messages to
be operated on by a single SEARCH/FETCH/STORE/COPY/MOVE 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/MOVE command. The

lowest processed UID is <LastUID>. The client needs to repeat
the operation for remaining messages, if required.

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 matching
messages exceeds the server's published 1000 messages limit.

```
C: 04 UID SEARCH UID 22000:25000 UNDELETED
S: * SEARCH 25000 24998 (... 997 UIDs ...) 23221
S: 04 OK [MESSAGELIMIT 1000 23221] SEARCH completed with 1000 partial
```

The following example demonstrates copy of messages with UIDs
between 18000:21000. The total message count exceeds the
server's published 1000 messages limit.

```
C: 05 UID COPY 18000:21000 "Trash"
S: * NO [MESSAGELIMIT 1000 20001] Too many messages to copy
S: 05 OK [COPYUID 1397597919 20001:21000 21363:22362] COPY completed f
```

Open Issue: Note that the above example shows a UID COPY that
partially fails. This is assumed to be better for clients that
don't understand the MESSAGELIMIT response code. However this
might cause naive clients to believe that the COPY fully
completed and that all messages were copied. (An alternative
would be to return MESSAGELIMIT in the tagged NO response,
meaning that no messages could be copied. However this
wouldn't work well with clients that don't support
MESSAGELIMIT response code.)

The following example shows MOVE of messages with UIDs between
18000:21000. The total message count exceeds the server's
published 1000 messages limit. The client that wants to move
all messages in the range and observes a MESSAGELIMIT response
code, can repeat the command by updating the UID set parameter
specified in the command. The client needs to keep doing this

until MESSAGELIMIT response is not returned (or until a tagged
           NO/BAD is returned).

             C: 06 UID MOVE 18000:21000 "Archive/2021/2021-12"
             S: * OK [COPYUID 1397597919 20001:21000 22363:23362] Some messages wer
             S: * 12336 EXPUNGE
             S: * 12335 EXPUNGE
             ...
             S: * 11335 EXPUNGE
             S: 06 OK [MESSAGELIMIT 1000 20001] MOVE completed for the last 1000 me

           The following example shows update of flags for messages with
           UIDs between 18000:20000. The total message count exceeds the
           server's published 1000 messages limit. The client that wants
           to change flags for all messages in the range and observes a
           MESSAGELIMIT response code, can repeat the command by updating
           the UID set parameter specified in the command. The client
           needs to keep doing this until MESSAGELIMIT response is not
           returned (or until a tagged NO/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 m

           The following example shows use of MESSAGELIMIT response code
           together with the PARTIAL extension. The total message count
           exceeds the server's published 1000 messages limit.

             C: 08 UID FETCH 22000:25000 (UID FLAGS MODSEQ) (PARTIAL -1:-1500)
             S: 08 NO [MESSAGELIMIT 1000] FETCH exceeds the maximum 1000 message li

   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: 031 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: 031 OK [EXPUNGEISSUED] Some messages were also expunged

4.2. Interaction with SORT and THREAD extensions

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

Servers that advertise MESSAGELIMIT N might be unable to execute a
SORT command in a mailbox with more than N messages, unless they
maintain indices for different SORT orders they support.

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 upper or lower case characters to define
token strings is for editorial clarity only. Implementations MUST
accept these strings in a case-insensitive fashion.

```
SP                  = <Defined in RFC 5234>
MINUS               = "-"

capability          =/ "PARTIAL"
                        ;; <capability> from [RFC3501]

modifier-partial    = "PARTIAL" SP partial-range

partial-range-first = nz-number ":" nz-number
    ;; Request to search from oldest (lowest UIDs) to
    ;; more recent messages.
    ;; A range 500:400 is the same as 400:500.
    ;; This is similar to <seq-range> from [RFC3501],
    ;; but cannot contain "*".

partial-range-last  = MINUS nz-number ":" MINUS nz-number
    ;; Request to search from newest (highest UIDs) to
    ;; oldest messages.
    ;; A range -500:-400 is the same as -400:-500.

partial-range       = partial-range-first / partial-range-last

search-return-opt   =/ modifier-partial
    ;; All conform to <search-return-opt>, from [IMAP-ABNF]/[RFC9051]

search-return-data  =/ ret-data-partial

ret-data-partial    = "PARTIAL"
                        SP "(" partial-range SP partial-results ")"
    ;; <partial-range> is the requested range.

partial-results     = sequence-set / "NIL"
    ;; <sequence-set> from [RFC3501].
    ;; NIL indicates no results correspond to the requested range.

tagged-ext-simple   =/ partial-range-last

fetch-modifier      =/ modifier-partial



capability          =/ "MESSAGELIMIT=" 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

TBD.

7. IANA Considerations

7.1. Changes/additions to the IMAP4 capabilities registry

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

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

IANA is requested to add definition of the PARTIAL extension to point to this document.

8. Acknowledgments

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

Editor of this document would like to thank the following people who provided useful comments or participated in discussions of this document: Timo Sirainen.

This document uses lots of text from RFC 5267. Thus work of the RFC 5267 authors Dave Cridland and Curtis King is appreciated.

9. References

9.1. Normative References

[ABNF]   Crocker, D., Ed. and P. Overell, Ed., "Augmented BNF for
           Syntax Specifications: ABNF", RFC 5234, 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>.

[RFC4731] Melnikov, A. and D. Cridland, "IMAP4 Extension to SEARCH
           Command for Controlling What Kind of Information Is
           Returned", RFC 4731, DOI 10.17487/RFC4731, November 2006,
           <https://www.rfc-editor.org/info/rfc4731>.

[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>.

[RFC5267]   Cridland, D. and C. King, "Contexts for IMAP4", RFC 5267,
            DOI 10.17487/RFC5267, July 2008, <https://www.rfc-
            editor.org/info/rfc5267>.

[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>.

9.2. Informative References

[RFC7162]   Melnikov, A. and D. Cridland, "IMAP Extensions: Quick
            Flag Changes Resynchronization (CONDSTORE) and Quick
            Mailbox Resynchronization (QRESYNC)", RFC 7162, DOI
            10.17487/RFC7162, May 2014, <https://www.rfc-editor.org/
            info/rfc7162>.

Index

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