MailMaint
Internet-Draft
Intended status: Standards Track
Expires: 22 June 2026

D. Eggert, Ed.
Apple Inc

19 December 2025

IMAP UIDBATCHES Extension
draft-ietf-mailmaint-imap-uidbatches-20

Abstract

The UIDBATCHES extension of the Internet Message Access Protocol (IMAP) allows clients to retrieve UID ranges that partition a mailbox’s messages into equally sized batches. This enables clients to perform operations such as FETCH, SEARCH, and STORE on specific message batches, providing better control over resource usage and response sizes. The extension is particularly useful with the UIDONLY mode where sequence numbers are unavailable.

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 22 June 2026.

Copyright Notice

Table of Contents

1. Introduction

This document defines an extension to the Internet Message Access Protocol [RFC9051] that enables clients to retrieve UID ranges which partition a mailbox’s messages into evenly sized batches. This extension is compatible with both IMAP4rev1 [RFC3501] and IMAP4rev2 [RFC9051].

The primary purpose of this extension is to allow clients to predetermine UID ranges that limit the number of messages each command operates on. This capability is especially beneficial when used with the [RFC9586] UIDONLY mode, where sequence numbers are unavailable to the client, making it difficult to create message batches using traditional methods.

2. Document Conventions

In protocol examples , "C:" indicates lines sent by a client that is connected to a server. "S:" indicates lines sent by the server to the client. These prefixes are not part of the protocol. Long lines in examples are wrapped using "The Single Backslash Strategy" described in [RFC8792].

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 [RFC9051] or keywords from this document.

3. The UIDBATCHES extension

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

3.1. UIDBATCHES Command

Arguments:
- Message count per batch.
- OPTIONAL batch range.

Responses:
- REQUIRED untagged response: UIDBATCHES

```
   Result:
      OK uidbatches completed
      NO command exceeds limits
      BAD command unknown or arguments invalid
```

The UIDBATCHES command requests UID ranges that partition the messages in the currently selected mailbox into equally sized batches. The server returns these ranges in descending UID order, with batch 1 containing the highest UIDs (most recent messages), batch 2 containing the next highest set of UIDs, and so on.

For a mailbox with M messages, requesting batches of size N returns UID ranges corresponding to the following sequence number ranges (where sequence numbers are ordered from 1 to M, with M being the most recent message):

```
Batch 1: M:(M-N+1)     // Most recent N messages
Batch 2: (M-N):(M-2*N+1)  // Next N messages
Batch 3: (M-2*N):(M-3*N+1) // Next N messages
...and so on
```

3.1.1. Example Usage

The following example demonstrates how a client uses UIDBATCHES to partition a mailbox into manageable batches:

```
========== NOTE: '\' line wrapping per RFC 8792 ===========

C: A142 SELECT INBOX
S: * 6823 EXISTS
S: * 1 RECENT
S: * OK [UNSEEN 12] Message 12 is first unseen
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: * OK [UIDNEXT 215296] Predicted next UID
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Deleted \Seen \*)] Limited
S: A142 OK [READ-WRITE] SELECT completed
C: A143 UIDBATCHES 2000
S: * UIDBATCHES (TAG "A143") \
       215295:99696,99695:20351,20350:7830,7829:1
S: A143 OK UIDBATCHES Completed
```

The server's response provides four UID ranges:

1. 215295:99696

2. 99695:20351

3. 20350:7830

4. 7829:1

Each range contains up to 2,000 messages, except the last range, which contains the remaining 823 messages.

As new messages cannot appear within these UID ranges, the number of messages in each range will not increase. It may decrease, though, as messages are deleted.

To prevent server overload, the client MUST NOT resend UIDBATCHES unless at least one of the following conditions is met:

1. A different mailbox has been selected

2. More than N/2 messages have been expunged from the mailbox (where N is the batch size)

3. More than N/2 new messages have been received into the mailbox

These restrictions define the only circumstances under which re-running UIDBATCHES is appropriate, as computing message batches may be resource-intensive for servers.

The client can keep track of the number of EXPUNGE or VANISHED messages and re-run UIDBATCHES if many messages are deleted.

As new messages arrive into the mailbox, the client should add these to a new message batch (starting at UID 215296 in the above example). Once N/2 or more new messages have been added to the mailbox, the client MAY ask for updated batches by re-running the UIDBATCHES command.

The server MAY reject UIDBATCHES commands with a NO response with the LIMIT response code if the client exceeds this limit.

3.1.2. Response Format

The server MUST reply with a UIDBATCHES response, even if no ranges are returned (see Section 3.1.6). The UIDBATCHES response MUST include the tag of the command it relates to (similar to an ESEARCH response defined in [RFC4731]).

The UID ranges in the response MUST be ordered in descending sequence, from the highest to the lowest UIDs.

3.1.3. Batch Sizes

To ensure efficient server operation and prevent abuse, this extension enforces constraints on batch sizes. The design balances server efficiency requirements with the primary use case of working effectively with the [RFC9586] UIDONLY mode, without creating a mechanism that circumvents the sequence number restrictions of that mode.

3.1.3.1. Batch Size Summary

The following tables provide a quick reference for batch size constraints:

Constraint	Client Request	Server Response
Minimum	500 messages	Aim for >= 90% of requested size
Maximum	No limit	Never exceed requested size
Exception	-	May be smaller during mailbox changes

Table 1: Batch Size Constraints

Key principles:

* Clients MUST request at least 500 messages per batch

* Servers MUST NOT return more messages than requested

* Servers SHOULD return batches close to the requested size (>= 90% when possible)

* Exact batch sizes may vary due to implementation efficiency or mailbox changes

3.1.3.2. Minimum Batch Size

The server MUST support batch sizes of 500 messages or larger. This minimum size prevents clients from misusing the extension to effectively reconstruct sequence numbers while still allowing reasonable batch operations.

Note that clients MUST be prepared to handle batches smaller than requested, as detailed in Section 3.1.3.3.

The server MUST respond with NO and a response code TOOFEW if the client uses a batch size smaller than the minimum allowed by the server:

S: A302 NO [TOOFEW] Minimum batch size is 500

3.1.3.3. Server Response Flexibility

While servers SHOULD ideally return batches that correspond exactly to the requested size, they have flexibility in specific circumstances to enable efficient implementations.

3.1.3.3.1. Hard Constraints

The server MUST NOT return ranges that contain more than the number of messages per batch requested by the client. This is a strict upper bound that cannot be exceeded.

If the requested batch size equals or exceeds the total number of messages in the mailbox, the server MUST return a single UID range spanning all messages.

3.1.3.3.2. Permitted Variations

Servers MAY return fewer messages per range in two specific circumstances:

1. When doing so makes the implementation substantially simpler and/or more efficient

2. When there are changes in mailbox state during the execution of the UIDBATCHES command, particularly when messages are expunged

However, servers SHOULD NOT return batches that are substantially smaller than requested and SHOULD aim to stay within 90% of the requested size. This guideline reflects the fact that clients typically choose batch sizes based on their intended use, such as displaying a specific number of messages to users.

3.1.3.3.3. Dynamic Changes

Mailbox state changes during UIDBATCHES execution can result in servers returning substantially fewer messages in each batch, particularly when message expungement reduces the overall mailbox size. Clients can detect these situations through the EXPUNGE, VANISHED, or EXISTS responses they receive.

3.1.3.3.4. Practical Implications

Due to these flexibility provisions, servers may return batches of varying sizes. For instance, when returning 3 batches of a requested size of 1,000, one might contain 990 messages, another 977, and the third 1,000 messages. Clients MUST be prepared to handle such variations.

When the total number of messages is not evenly divisible by the requested batch size, the final batch will contain the remainder. Therefore, the last batch in the mailbox (containing the lowest UIDs) will typically have fewer messages than requested.

3.1.3.4. Design Rationale

These restrictions provide servers with implementation flexibility while preventing clients from misusing the extension to effectively reconstruct sequence numbers. Appendix A.1 also outlines some reasoning for these limitations.

The flexibility regarding batch sizes is designed to enable efficient server implementations while maintaining predictable behavior for clients. This leeway is not intended as a general permission to return arbitrarily sized batches, but rather to accommodate implementation constraints and dynamic mailbox changes.

3.1.4. UIDs

The server MAY return UID ranges with UIDs that do not exist on the server. The client as a result MUST NOT make assumptions about the existence of messages. If the server returns the response

========== NOTE: '\' line wrapping per RFC 8792 ===========

```
S: * UIDBATCHES (TAG "A302") \
       163886:99703,99696:20358,20351:7841,7830:1
S: A302 OK UIDBATCHES Completed
```

there may not be any messages on the server with the UIDs such as 163886, 99703, 99696, etc.

The range 163886:99703 will span approximately the requested number of messages (may be less, see Section 3.1.3), but its start and end UIDs may not correspond to messages on the server.

This gives the server implementation some flexibility as to which UID ranges to return. They might, e.g., return 163886:99697 and 99696:20358 instead of 163886:99703 and 99696:20358 -- assuming that there are no messages in the range 99702:99697.

If there are fewer messages in the mailbox than the requested batch size, the server would return a single batch that contains all messages in the mailbox.

When applying the flexibility described above to the last batch in the mailbox, ending that batch with UID 1 makes it unambiguous to the client that this range is in fact the last range. For example, if the message with the lowest UID is 302, the server can return 7829:1 instead of 7829:302.

3.1.5. Batch Ranges

A client can optionally provide a batch range. The server limits its response to UID ranges corresponding to the specified batch indices. For example, if the client sends

C: A302 UIDBATCHES 2000 10:20

for a mailbox with 100,000 messages, the server would return the 10th to 20th batches. The 10th batch would correspond to message sequence numbers ‘82000:80001‘ and the 20th batch would correspond to message sequence numbers ‘62000:60001‘.

Batches start at the highest UIDs: batch 1 is the batch with the highest UIDs.

The UID ranges that the server returns would still split the mailbox’s messages into batches of the requested size (2,000 in the example).

If the client requests more batches than exist on the server, the server would return those that do exist. For example if the client sends

C: A302 UIDBATCHES 2000 1:5

and the selected mailbox has 7,000 messages, the server would then return a UIDBATCHES response with only 4 UID ranges.

Batch ranges such as 1:4 in the above example MUST be ordered lowest to highest, i.e. be sent as 1:4 and not as 4:1. Servers MUST reject batch ranges that are in the wrong order with BAD and a response code CLIENTBUG:

```
C: A302 UIDBATCHES 2000 4:1
S: A302 BAD [CLIENTBUG] Invalid batch range
```

If the client requests a range of batches that do not exist on the server, the server MUST still return an empty response. See section Section 3.1.6.

The number of messages per batch returned by the server may be approximate as detailed in Section 3.1.3. As a result, if the client needs to request consecutive batch ranges such as 1:100, 101:200, 201:300, and so on, the client may want to make these batch ranges overlap by e.g. requesting 1:100, 100:200, and 200:300. While the UIDs returned may not correspond to existing messages (as described in Section 3.1.4) and mailbox state can change between requests, checking whether the returned UID ranges overlap can help clients detect potential inconsistencies, though how to handle such situations depends on the specific client implementation requirements.

Clients MUST NOT request batch ranges that span more than 100,000 messages, i.e. the number of batches multiplied by the batch size MUST NOT be larger than 100,000. This restriction applies only when a batch range is specified; when no batch range is provided, the client is requesting all batches but the server may limit its response as described in Section 3.1.7. The server MAY reject UIDBATCHES commands with a NO response with the TOOMANY response code if the client exceeds this limit.

```
C: A302 UIDBATCHES 2000 1:100
S: A302 NO [TOOMANY] Too many messages
```

3.1.6. Empty Responses

When the client issues any valid UIDBATCHES command and the mailbox is empty, the server MUST reply with a UIDBATCHES response, e.g.

```
S: * UIDBATCHES (TAG "A302")
S: A302 OK UIDBATCHES Completed
```

If the client requests a range of batches that do not exist, the server MUST reply with an empty UIDBATCHES response. If the mailbox has 7,000 messages, and the client sends

```
C: A302 UIDBATCHES 2000 6:8
```

the server would respond with

```
S: * UIDBATCHES (TAG "A302")
S: A302 OK UIDBATCHES Completed
```

3.1.7. Large Mailboxes

The server may not be able to return all UID ranges if the mailbox contains an extremely large number of messages.

The server MUST at least support returning UID ranges spanning 100,000 messages. See Section 3.1.5 for details on this limit.

If the server can not return all of the requested UID ranges, it MUST respond with a NO response with the TOOMANY response code. Notably, when the client requests all UID ranges and the mailbox has more than 100,000 messages, the server MAY reply with a NO response. For example:

```
C: A302 UIDBATCHES 2000
S: A302 NO [TOOMANY] Too many messages in mailbox
```

The client should know what the message count in the mailbox is, and if the message count exceeds 100,000 it may choose to always request batch ranges as discussed in Section 3.1.5 instead of requesting all batches.

3.2. Interaction with MESSAGELIMIT Extension

When the server supports both the [RFC9738] MESSAGELIMIT and UIDBATCHES extension, the client SHOULD request batches no larger than the specified maximum number of messages that can be processed in a single command. The client MAY choose to use a smaller batch size.

Additionally, since servers MAY limit the number of UIDs returned in response to UIDBATCHES, it is reasonable to assume that they would at most return N UIDs where N is the limit the server announced as its MESSAGELIMIT.

3.3. Interaction with UIDONLY Extension

The UIDBATCHES extension allows clients to create UID ranges for message batches even when the connection operates in UIDONLY mode, which otherwise doesn’t allow for using message sequence numbers.

This interaction is particularly important because the UIDONLY extension disallows the use of sequence numbers. While the PARTIAL extension [RFC9394] provides paged SEARCH and FETCH operations, some clients need to predetermine UID ranges for batches upfront.

UIDBATCHES enables such clients to use the same overall batching strategy regardless of whether the server supports UIDONLY, PARTIAL, or neither, making client implementations simpler and more consistent.

When operating in UIDONLY mode, clients SHOULD use UIDBATCHES to determine appropriate UID ranges for batch operations rather than attempting to construct batches using sequence-number-based approaches that would violate UIDONLY restrictions.

The batch size constraints defined in Section 3.1.3 serve dual purposes: ensuring server efficiency and preventing UIDBATCHES from becoming a mechanism to effectively reconstruct sequence numbers. Without these constraints, clients could request very small batch sizes to obtain fine-grained positional information about messages, which would circumvent the sequence number restrictions of UIDONLY mode. The extension is designed specifically to support the legitimate need of predetermining message batches upfront, while maintaining the architectural intent of UIDONLY mode.

3.4. Interaction with SEARCHRES Extension

UIDBATCHES is not a SEARCH nor UID SEARCH command. Servers that support SEARCHRES [RFC5182] MUST NOT store the result of UIDBATCHES in the $ variable.

4. Formal syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) notation as specified in [RFC5234].

Non-terminals referenced but not defined below are as defined by IMAP4 [RFC9051].

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          =/ "UIDBATCHES"
                       ;; <capability> from [RFC9051]

command-select      =/ message-batches

message-batches     = "UIDBATCHES" SP nz-number
                      [SP nz-number ":" nz-number]

uidbatches-response = "UIDBATCHES" search-correlator
                      [SP uid-range *("," uid-range) ]

mailbox-data        =/ uidbatches-response

resp-text-code      =/ "TOOFEW" / "TOOMANY"
```

5. Operational Considerations

This document defines an optimization that can reduce both the amount of work performed by the server and 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. 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 the introduction of new implementation bugs.

6. Implementation Status

This section is to be removed before publishing as an RFC.

[RFC EDITOR: Please remove this section and the reference to RFC 7942 before publication.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation

and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

6.1. Cyrus Server

Cyrus Server is a production-level open source scalable enterprise mail system from Computing Services at Carnegie Mellon University. This implementation supports all of the requirements described in this document and is freely distributable under a BSD-style license. More information is available at https://cyrusimap.org and https://www.cmu.edu/computing/.

6.2. Apple Mail

Mail for iOS, iPadOS, and visionOS is an email client included by Apple with these operating systems. As of version 26, this production-level implementation from Apple Inc supports all of the requirements described in this document. More information is available at https://www.apple.com and https://developer.apple.com/documentation/technotes/tn3191-imap-extensions-supported-by-mail.

7. Security Considerations

This document defines an additional IMAP4 capability. As such, it does not change the underlying security considerations of IMAP4rev1 [RFC3501] and IMAP4rev2 [RFC9051]. The authors and reviewers believe that no new security issues are introduced with this additional IMAP4 capability.

One consideration during the design of this extension was the potential for clients to cause servers to perform excessive computational work by repeatedly requesting batch calculations. The restrictions on when clients may re-run UIDBATCHES (Section 3.1) and the batch size constraints (Section 3.1.3) are designed to mitigate this concern. Servers may reject requests that exceed these limits with the LIMIT, TOOFEW, or TOOMANY response codes.

As this extension involves new code in both clients and servers, rigorous testing of such code is required in order to avoid introducing new implementation bugs.

8. IANA Considerations

8.1. 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 registrations of the "UIDBATCHES" capability to this registry, pointing to this document.

8.2. Changes/additions to the IMAP response codes registry

IMAP4 response codes 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/imap-response-codes

IANA is requested to add registrations of TOOMANY and TOOFEW to this registry, pointing to this document.

9. References

9.1. Normative References

[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>.

[RFC5234] 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>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[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>.

[RFC9738] Melnikov, A., Achuthan, A. P., Nagulakonda, V., and L. Alves, "IMAP MESSAGELIMIT Extension", RFC 9738, DOI 10.17487/RFC9738, March 2025, <https://www.rfc-editor.org/info/rfc9738>.

9.2. Informative References

[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>.

[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>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/info/rfc8792>.

[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>.

[RFC9586] Melnikov, A., Achuthan, A. P., Nagulakonda, V., Singh, A., and L. Alves, "IMAP Extension for Using and Returning Unique Identifiers (UIDs) Only", RFC 9586, DOI 10.17487/RFC9586, May 2024, <https://www.rfc-editor.org/info/rfc9586>.

Appendix A. Comparison with Existing Commands and Extensions

A.1. Similarity to UID SEARCH Command

The UIDBATCHES is in effect nothing more than shorthand for a UID SEARCH command of the form

C: A145 UID SEARCH RETURN () <M>,<M-N>,<M-2*N>,<M-3*N>,...

where M is the number of messages in the mailbox and N is the requested batch count.

The special purpose UIDBATCHES command, though, tries to address two problems:

(a) for many servers, UID SEARCH commands specifying sequence numbers are costly, especially for mailboxes with many messages.

(b) the UIDONLY extension disallows the use of sequence numbers and thus makes it difficult for the client to split its commands into batches of a size that works well for the client and server.

By providing a special purpose command, servers can implement a different, optimized code path for determining message batches. And servers using the UIDONLY extension can provide a facility to let the client determine message batches without using sequence numbers in a UID SEARCH command.

Section 3.1.3 describes some implementation restrictions to ensure this.

A.2. Similarity to PARTIAL Extension

The PARTIAL extension in [RFC9394] provides a different way for the client to split its commands into batches by using paged SEARCH and FETCH.

The intention of the UIDBATCHES command is to let the client pre-determine message batches of a desired size.

This makes it easier for the client to share implementation between servers regardless of their support of PARTIAL. And additionally, because the client can issue a corresponding UID SEARCH command to servers that do not implement UIDBATCHES, the client can use similar batching implementations for servers that support UIDBATCHES and those that do not.

Author’s Address

Daniel Eggert (editor)
Apple Inc
One Apple Park Way
Cupertino, CA 95014
United States of America
Email: deggert@apple.com