Internet Engineering Task Force (IETF)
Request for Comments: 8474
Updates: 3501
Category: Standards Track
ISSN: 2070-1721

B. Gondwana, Ed.
FastMail

September 2018

# IMAP Extension for Object Identifiers

## Abstract

This document updates RFC 3501 (IMAP4rev1) with persistent identifiers on mailboxes and messages to allow clients to more efficiently reuse cached data when resources have changed location on the server.

## Status of This Memo

This is an Internet Standards Track document.

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). Further information on Internet Standards is available in 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/rfc8474.

Table of Contents



## 1. Introduction

IMAP stores are often used by many clients. Each client may cache data from the server so that it doesn't need to redownload information. [RFC3501] states that a mailbox can be uniquely referenced by its name and UIDVALIDITY, and a message within that mailbox can be uniquely referenced by its mailbox (name + UIDVALIDITY) and unique identifier (UID). The triple of mailbox name, UIDVALIDITY, and UID is guaranteed to be immutable.

[RFC4315] defines a COPYUID response that allows a client that copies messages to know the mapping between the UIDs in the source and destination mailboxes and, hence, update its local cache.

If a mailbox is successfully renamed by a client, that client will know that the same messages exist in the destination mailbox name as previously existed in the source mailbox name.

The result is that the client that copies (or moves [RFC6851]) messages or renames a mailbox can update its local cache, but any other client connected to the same store cannot know with certainty that the messages are identical, so it will redownload everything.

This extension adds new properties to a message (EMAILID) and mailbox (MAILBOXID). These properties allow a client to quickly identify messages or mailboxes that have been renamed by another client.

This extension also adds an optional thread identifier (THREADID) to messages, which can be used by the server to indicate messages that it has identified to be related. A server that does not implement threading will return NIL to all requests for THREADID.

## 2. Conventions Used in This Document

In examples, "C:" indicates lines sent by a client that is connected to a server. "S:" indicates lines sent by the server to the client.

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.

## 3. CAPABILITY Identification

IMAP servers that support this extension MUST include "OBJECTID" in the response list to the CAPABILITY command.

## 4. MAILBOXID Object Identifier

The MAILBOXID is a server-allocated unique identifier for each mailbox.

The server MUST return the same MAILBOXID for a mailbox with the same name and UIDVALIDITY.

The server MUST NOT report the same MAILBOXID for two mailboxes at the same time.

The server MUST NOT reuse the same MAILBOXID for a mailbox that does not obey all the invariants that [RFC3501] defines for a mailbox that does not change name or UIDVALIDITY.

The server MUST keep the same MAILBOXID for the source and destination when renaming a mailbox in a way that keeps the same messages (but see [RFC3501] for the special case regarding the renaming of INBOX, which is treated as creating a new mailbox and moving the messages).

### 4.1. New Response Code for CREATE

This document extends the CREATE command to have the response code MAILBOXID on successful mailbox creation.

A server advertising the OBJECTID capability MUST include the MAILBOXID response code in the tagged OK response to all successful CREATE commands.

Syntax: "MAILBOXID" SP "(" objectid ")"

Response code in tagged OK response for successful CREATE command.

Example:

```
C: 3 create foo
S: 3 OK [MAILBOXID (F2212ea87-6097-4256-9d51-71338625)] Completed
C: 4 create bar
S: 4 OK [MAILBOXID (F6352ae03-b7f5-463c-896f-d8b48ee3)] Completed
C: 5 create foo
S: 5 NO Mailbox already exists
```

### 4.2. New OK Untagged Response for SELECT and EXAMINE

This document adds a new untagged response code to the SELECT and EXAMINE commands.

A server advertising the OBJECTID capability MUST return an untagged OK response with the MAILBOXID response code on all successful SELECT and EXAMINE commands.

Syntax: "OK" SP "[" "MAILBOXID" SP "(" objectid ")" "]" SP text

Untagged OK response to SELECT or EXAMINE.

Example:

```
C: 27 select "foo"
[...]
S: * OK [MAILBOXID (F2212ea87-6097-4256-9d51-71338625)] Ok
[...]
S: 27 OK [READ-WRITE] Completed
```

### 4.3. New Attribute for STATUS

This document adds the MAILBOXID attribute to the STATUS command using the extended syntax defined in [RFC4466].

A server that advertises the OBJECTID capability MUST support the MAILBOXID status attribute.

Syntax: "MAILBOXID"

The attribute in the STATUS command.

Syntax: "MAILBOXID" SP "(" objectid ")"

The response item in the STATUS response contains the ObjectID assigned by the server for this mailbox.

Example:

```
 C: 6 status foo (mailboxid)
 S: * STATUS foo (MAILBOXID (F2212ea87-6097-4256-9d51-71338625))
 S: 6 OK Completed
 C: 7 status bar (mailboxid)
 S: * STATUS bar (MAILBOXID (F6352ae03-b7f5-463c-896f-d8b48ee3))
 S: 7 OK Completed
 C: 8 rename foo renamed
 S: * OK rename foo renamed
 S: 8 OK Completed
 C: 9 status renamed (mailboxid)
 S: * STATUS renamed (MAILBOXID (F2212ea87-6097-4256-9d51-71338625))
 S: 9 OK Completed
 C: 10 status bar (mailboxid)
 S: * STATUS bar (MAILBOXID (F6352ae03-b7f5-463c-896f-d8b48ee3))
 S: 10 OK Completed
```

When the LIST-STATUS IMAP capability defined in [RFC5819] is also available, the STATUS command can be combined with the LIST command.

Example:

```
C: 11 list "" "*" return (status (mailboxid))
S: * LIST (\HasNoChildren) "." INBOX
S: * STATUS INBOX (MAILBOXID (Ff8e3ead4-9389-4aff-adb1-d8d89efd8cbf))
S: * LIST (\HasNoChildren) "." bar
S: * STATUS bar (MAILBOXID (F6352ae03-b7f5-463c-896f-d8b48ee3))
S: * LIST (\HasNoChildren) "." renamed
S: * STATUS renamed (MAILBOXID (F2212ea87-6097-4256-9d51-71338625))
S: 11 OK Completed (0.001 secs 3 calls)
```

## 5. EMAILID Object Identifier and THREADID Correlator

### 5.1. EMAILID Identifier for Identical Messages

The EMAILID data item is an ObjectID that uniquely identifies the content of a single message. Anything that must remain immutable on a {name, uidvalidity, uid} triple must also be the same between messages with the same EMAILID.

The server MUST return the same EMAILID for the same triple; hence, EMAILID is immutable.

The server MUST return the same EMAILID as the source message for the matching destination message in the COPYUID pairing after a COPY or MOVE command [RFC6851].

The server MAY assign the same EMAILID as an existing message upon APPEND (e.g., if it detects that the new message has exactly identical content to that of an existing message).

NOTE: EMAILID only identifies the immutable content of the message. In particular, it is possible for different messages with the same EMAILID to have different keywords. This document does not specify a way to STORE by EMAILID.

### 5.2. THREADID Identifier for Related Messages

The THREADID data item is an ObjectID that uniquely identifies a set of messages that the server believes should be grouped together when presented.

THREADID calculation is generally based on some combination of References, In-Reply-To, and Subject, but the exact logic is left up to the server implementation. [RFC5256] describes some algorithms that could be used; however, this specification does not mandate any particular strategy.

The server MUST return the same THREADID for all messages with the same EMAILID.

The server SHOULD return the same THREADID for related messages, even if they are in different mailboxes; for example, messages that would appear in the same thread if they were in the same mailbox SHOULD have the same THREADID, even if they are in different mailboxes.

The server MUST NOT change the THREADID of a message once reported.

THREADID is OPTIONAL; if the server doesn't support THREADID or is unable to calculate relationships between messages, it MUST return NIL to all FETCH responses for the THREADID data item, and a SEARCH for THREADID MUST NOT match any messages.

The server MUST NOT use the same ObjectID value for both EMAILIDs and THREADIDs. If they are stored with the same value internally, the server can generate prefixed values (as shown in the examples below with M and T prefixes) to avoid clashes.

## 5.3. New Message Data Items in FETCH and UID FETCH Commands

This document defines two FETCH items:

Syntax: "EMAILID"

The EMAILID message data item causes the server to return EMAILID FETCH response data items.

Syntax: "THREADID"

The THREADID message data item causes the server to return THREADID FETCH response data items.

This document defines the following responses:

Syntax: "EMAILID" SP "(" objectid ")"

The EMAILID response data item contains the server-assigned ObjectID for each message.

Syntax: "THREADID" SP "(" objectid ")"

The THREADID response data item contains the server-assigned ObjectID for the set of related messages to which this message belongs.

Syntax: "THREADID" SP nil

The NIL value is returned for the THREADID response data item when the server mailbox does not support THREADID calculation.

Example:

```
C: 5 append inbox "20-Mar-2018 03:07:37 +1100" {733}
[...]
Subject: Message A
Message-ID: <fake.1521475657.54797@example.com>
[...]
S: 5 OK [APPENDUID 1521475658 1] Completed

C: 11 append inbox "20-Mar-2018 03:07:37 +1100" {793}
[...]
Subject: Re: Message A
Message-ID: <fake.1521475657.21213@example.org>
References: <fake.1521475657.54797@example.com>
[...]
S: 11 OK [APPENDUID 1521475658 2] Completed

C: 17 append inbox "20-Mar-2018 03:07:37 +1100" {736}
[...]
Subject: Message C
Message-ID: <fake.1521475657.60280@example.com>
[...]
S: 17 OK [APPENDUID 1521475658 3] Completed

C: 22 fetch 1:* (emailid threadid)
S: * 1 FETCH (EMAILID (M6d99ac3275bb4e) THREADID (T64b478a75b7ea9))
S: * 2 FETCH (EMAILID (M288836c4c7a762) THREADID (T64b478a75b7ea9))
S: * 3 FETCH (EMAILID (M5fdc09b49ea703) THREADID (T11863d02dd95b5))
S: 22 OK Completed (0.000 sec)

C: 23 move 2 foo
S: * OK [COPYUID 1521475659 2 1] Completed
S: * 2 EXPUNGE
S: 23 OK Completed

C: 24 fetch 1:* (emailid threadid)
S: * 1 FETCH (EMAILID (M6d99ac3275bb4e) THREADID (T64b478a75b7ea9))
S: * 2 FETCH (EMAILID (M5fdc09b49ea703) THREADID (T11863d02dd95b5))
S: 24 OK Completed (0.000 sec)
C: 25 select "foo"

C: 25 select "foo"
[...]
S: 25 OK [READ-WRITE] Completed
C: 26 fetch 1:* (emailid threadid)
S: * 1 FETCH (EMAILID (M288836c4c7a762) THREADID (T64b478a75b7ea9))
S: 26 OK Completed (0.000 sec)
```

Example: (no THREADID support)

```
C: 26 fetch 1:* (emailid threadid)
S: * 1 FETCH (EMAILID (M00000001) THREADID NIL)
S: * 2 FETCH (EMAILID (M00000002) THREADID NIL)
S: 26 OK Completed (0.000 sec)
```

## 6. New Filters on SEARCH Command

This document defines the filters EMAILID and THREADID on the SEARCH command.

Syntax: "EMAILID" SP objectid

Messages whose EMAILID is exactly the specified ObjectID.

Syntax: "THREADID" SP objectid

Messages whose THREADID is exactly the specified ObjectID.

Example: (as if run before the MOVE shown above when the mailbox had three messages)

```
C: 27 search emailid M6d99ac3275bb4e
S: * SEARCH 1
S: 27 OK Completed (1 msgs in 0.000 secs)
C: 28 search threadid T64b478a75b7ea9
S: * SEARCH 1 2
S: 28 OK Completed (2 msgs in 0.000 secs)
```

## 7. Formal Syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) [RFC5234] notation. Elements not defined here can be found in the formal syntax of the ABNF [RFC5234], IMAP [RFC3501], and IMAP ABNF extensions [RFC4466] specifications.

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.

Please note specifically that ObjectID values are case sensitive.

```
capability =/ "OBJECTID"

fetch-att =/ "EMAILID" / "THREADID"

fetch-emailid-resp = "EMAILID" SP "(" objectid ")"
        ; follows tagged-ext production from [RFC4466]

fetch-threadid-resp = "THREADID" SP ( "(" objectid ")" / nil )
        ; follows tagged-ext production from [RFC4466]

msg-att-static =/ fetch-emailid-resp / fetch-threadid-resp

objectid = 1*255(ALPHA / DIGIT / "_" / "-")
        ; characters in object identifiers are case
        ; significant

resp-text-code =/ "MAILBOXID" SP "(" objectid ")"
        ; incorporated before the expansion rule of
        ;  atom [SP 1*<any TEXT-CHAR except "]">]
        ; that appears in [RFC3501]

search-key =/ "EMAILID" SP objectid / "THREADID" SP objectid

status-att =/ "MAILBOXID"

status-att-val =/ "MAILBOXID" SP "(" objectid ")"
        ; follows tagged-ext production from [RFC4466]
```

# 8. Implementation Considerations

## 8.1. Assigning Object Identifiers

All ObjectID values are allocated by the server.

In the interest of reducing the possibilities of encoding mistakes, ObjectIDs are restricted to a safe subset of possible byte values; in order to allow clients to allocate storage, they are restricted in length.

An ObjectID is a string of 1 to 255 characters from the following set of 64 codepoints: a-z, A-Z, 0-9, _, -. These characters are safe to use in almost any context (e.g., filesystems, URIs, IMAP atoms). These are the same characters defined as base64url in [RFC4648].

For maximum safety, servers should also follow defensive allocation strategies to avoid creating risks where glob completion or data type detection may be present (e.g., on filesystems or in spreadsheets). In particular, it is wise to avoid:

- IDs starting with a dash
- IDs starting with digits
- IDs that contain only digits
- IDs that differ only by ASCII case (for example, A vs. a)
- the specific sequence of three characters NIL in any case (because this sequence can be confused with the IMAP protocol expression of the null value)

A good solution to these issues is to prefix every ID with a single alphabetical character.

### 8.2. Interaction with Special Cases

The case of RENAME INBOX may need special handling because it has special behavior, as defined in [RFC3501], Section 6.3.5.

It is advisable (though not required) to have MAILBOXID be globally unique, but it is only required to be unique within messages offered to a single client login to a single server hostname. For example, a proxy that aggregates multiple independent servers MUST NOT advertise the OBJECTID capability unless it can guarantee that different objects will never use the same identifiers, even if backend object identifiers collide.

### 8.3. Client Usage

Servers that implement both RFC 6154 and this specification should optimize their execution of commands like UID SEARCH OR EMAILID 1234 EMAILID 4321.

Clients can assume that searching the all-mail mailbox using OR/EMAILID or OR/THREADID is a fast way to find messages again if some other client has moved them out of the mailbox where they were previously seen.

Clients that cache data offline should fetch the EMAILID of all new messages to avoid redownloading already-cached message details.

Clients should fetch the MAILBOXID for any new mailboxes before discarding cache data for any mailbox that is no longer present on the server so that they can detect renames and avoid redownloading data.

### 8.4. Advice to Client Implementers

In cases of server failure and disaster recovery, or misbehaving servers, it is possible that a client will be sent invalid information, e.g., identical ObjectIDs or ObjectIDs that have changed where they MUST NOT change according to this document.

In a case where a client detects inconsistent ObjectID responses from a server, it SHOULD fall back to relying on the guarantees of RFC 3501. For simplicity, a client MAY instead choose to discard its entire cache and resync all state from the server.

Client authors protecting against server misbehavior MUST ensure that their design cannot get into an infinite loop of discarding cache and fetching the same data repeatedly without user interaction.

## 9. Future Considerations

This extension is intentionally defined to be compatible with the data model in [JMAP-MAIL].

A future extension could be proposed to give a way to SELECT a mailbox by MAILBOXID rather than name.

A future extension to [RFC5228] could allow fileinto by MAILBOXID rather than name.

An extension to allow fetching message content directly via EMAILID and message listings by THREADID could be proposed.

## 10. IANA Considerations

IANA has added "OBJECTID" to the "IMAP Capabilities" registry located at <https://www.iana.org/assignments/imap-capabilities> with a reference to this document.

IANA has added "MAILBOXID" to the "IMAP Response Codes" registry located at <https://www.iana.org/assignments/imap-response-codes> with a reference to this document.

## 11. Security Considerations

It is strongly advised that servers generate ObjectIDs that are safe to use as filesystem names and unlikely to be autodetected as numbers. See implementation considerations.

If a digest is used for ID generation, it must have a collision-resistant property, so server implementations are advised to monitor current security research and choose secure digests. As the IDs are generated by the server, it will be possible to migrate to a new hash by just using the new algorithm when creating new IDs. This is particularly true if a prefix is used on each ID, which can be changed when the algorithm changes.

The use of a digest for ID generation may be used as proof that a particular sequence of bytes was seen by the server. However, this is only a risk if IDs are leaked to clients who don't have permission to fetch the data directly. Servers that are expected to handle highly sensitive data should consider this when choosing how to create IDs.

See also the security considerations in [RFC3501], Section 11.

## 12. References

### 12.1. 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>.

[RFC4315] Crispin, M., "Internet Message Access Protocol (IMAP) - UIDPLUS extension", RFC 4315, DOI 10.17487/RFC4315, December 2005, <https://www.rfc-editor.org/info/rfc4315>.

[RFC4466] Melnikov, A. and C. Daboo, "Collected Extensions to IMAP4 ABNF", RFC 4466, DOI 10.17487/RFC4466, April 2006, <https://www.rfc-editor.org/info/rfc4466>.

[RFC5228] Guenther, P., Ed. and T. Showalter, Ed., "Sieve: An Email Filtering Language", RFC 5228, DOI 10.17487/RFC5228, January 2008, <https://www.rfc-editor.org/info/rfc5228>.

[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>.

[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>.

[RFC5819] Melnikov, A. and T. Sirainen, "IMAP4 Extension for Returning STATUS Information in Extended LIST", RFC 5819, DOI 10.17487/RFC5819, March 2010, <https://www.rfc-editor.org/info/rfc5819>.

[RFC6851] Gulbrandsen, A. and N. Freed, Ed., "Internet Message Access Protocol (IMAP) - MOVE Extension", RFC 6851, DOI 10.17487/RFC6851, January 2013, <https://www.rfc-editor.org/info/rfc6851>.

[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>.

## 12.2. Informative References

[JMAP-MAIL] Jenkins, N. and C. Newman, "JMAP for Mail", Work in Progress, draft-ietf-jmap-mail-07, August 2018.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

## Appendix A. Ideas for Implementing Object Identifiers

Ideas for calculating MAILBOXID:

- Universally Unique Identifier (UUID) [RFC4122]
- Server-assigned sequence number (guaranteed not to be reused)

Ideas for implementing EMAILID:

- Digest of message content (RFC822 bytes) -- expensive unless cached
- UUID [RFC4122]
- Server-assigned sequence number (guaranteed not to be reused)

Ideas for implementing THREADID:

- Derive from EMAILID of first seen message in the thread.
- UUID [RFC4122]
- Server-assigned sequence number (guaranteed not to be reused)

There is a need to index and look up reference/in-reply-to data at message creation to efficiently find matching messages for threading. Threading may be either across mailboxes or within each mailbox only. The server has significant leeway here.

## Acknowledgments

The author would like to thank the EXTRA working group at IETF for feedback and advice -- in particular, Arnt Gulbrandsen, Brandon Long, Chris Newman, and Josef Sipek.

This document drew inspiration from the Gmail X-GM-THRID and X-GM-MSGID implementations as currently defined at <https://developers.google.com/gmail/imap/imap-extensions>, as well as the X-GUID implementation in the Dovecot server.

## Author's Address

Bron Gondwana (editor)
FastMail
Level 2, 114 William St
Melbourne VIC 3000
Australia

Email: brong@fastmailteam.com
URI: https://www.fastmail.com