mailmaint
Internet-Draft
Obsoletes: RFC8474 (if approved)
Updates: RFC3501, RFC9051, RFC9698 (if approved)
Intended status: Standards Track
Expires: 25 September 2026

B. Gondwana
Fastmail
M. De Gennaro
Stalwart Labs

24 March 2026

IMAP Extension for Object Identifiers
draft-ietf-mailmaint-imap-objectid-bis-03

Abstract

This document defines the OBJECTID+ extension for IMAP, which obsoletes [RFC8474]. OBJECTID+ introduces a compound OBJECTID response format that bundles object identifiers into key-value pairs, an ACCOUNTID identifier for account-level context, OBJECTID response codes for the RENAME command, and identifier-based mailbox selection via SELECT and EXAMINE. The OBJECTID+ extension is activated implicitly when a client uses any OBJECTID+-specific feature, ensuring backward compatibility with clients that only support [RFC8474]. This document also updates [RFC9698]: when JMAPACCESS is advertised alongside OBJECTID+, ACCOUNTID values MUST correspond to JMAP accountIds.

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 25 September 2026.

Table of Contents

1. Introduction

This document obsoletes [RFC8474] and defines persistent identifiers on mailboxes and messages to allow clients to more efficiently reuse cached data when resources have changed location on the server. It also updates [RFC9698]: when JMAPACCESS is advertised alongside OBJECTID+, ACCOUNTID values MUST correspond to JMAP accountIds.

The OBJECTID+ extension builds upon the identifier framework established by [RFC8474] and introduces several new capabilities. It defines a compound OBJECTID response format that bundles multiple identifiers into a parenthesized list of key-value pairs; identifiers that the server does not support are simply omitted from the response. This compound format is used uniformly across SELECT, EXAMINE, CREATE, RENAME, STATUS, and FETCH responses once the extension has been activated.

Four types of object identifiers may appear within the compound OBJECTID response. MAILBOXID is a server-allocated identifier for each mailbox that persists across renames, allowing clients to detect that a mailbox has been renamed rather than deleted and recreated. EMAILID is an identifier for message content that persists across COPY and MOVE operations, allowing clients to avoid redownloading messages that have changed location. THREADID is an optional identifier grouping related messages, allowing clients to display conversations. ACCOUNTID is a new identifier for account-level context, enabling disambiguation of mailboxes in environments where multiple accounts are accessible through a single IMAP session.

The extension also introduces identifier-based mailbox selection via the OBJECTID parameter on SELECT and EXAMINE, allowing clients to reliably reselect mailboxes after renames. Additionally, the RENAME command now returns an OBJECTID response code, providing the server-allocated identifiers for the renamed mailbox.

All identifier types are optional within the compound OBJECTID response; a server that does not support a particular identifier simply omits it. The empty compound response "OBJECTID ()" is valid and indicates that the server supports the OBJECTID+ extension but does not have any identifiers to return in a given context.

1.1. Notational Conventions

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.

2. CAPABILITY Identification

2.1. OBJECTID and OBJECTID+ Capabilities

This document obsoletes [RFC8474] and defines the OBJECTID+ capability. The OBJECTID+ capability is independent of the OBJECTID capability defined in [RFC8474]: a server MAY advertise OBJECTID+ alone, or it MAY advertise both OBJECTID and OBJECTID+ to provide backward compatibility with clients that only support [RFC8474].

A server that advertises both capabilities MUST behave as defined in [RFC8474] until the client activates OBJECTID+ (Section 2.2). A server that advertises only OBJECTID+ is not required to support the individual MAILBOXID, EMAILID, or THREADID attributes defined in [RFC8474].

The OBJECTID+ extension adds the ACCOUNTID identifier (Section 4), the compound OBJECTID response format (Section 3), the OBJECTID SELECT/EXAMINE parameter (Section 7.1), the OBJECTID FETCH data item (Section 7.5), the OBJECTID STATUS attribute (Section 7.4), the OBJECTID response code on CREATE (Section 7.2) and RENAME (Section 7.3), and the JMAPACCESS capability and GETJMAPACCESS command (Section 9).

2.2. Activation of OBJECTID+

A client activates the OBJECTID+ extension by using any OBJECTID+-specific feature. The server MUST NOT send OBJECTID+-specific responses until the extension has been activated.

The extension is activated by any of the following:

* The client issues ENABLE OBJECTID+ ([RFC5161])

* The client uses the OBJECTID parameter on SELECT or EXAMINE (Section 7.1)

* The client requests the OBJECTID status attribute (Section 7.4)

* The client requests the OBJECTID FETCH data item (Section 7.5)

When the extension is activated by any mechanism other than ENABLE, the server MUST send an untagged ENABLED response listing OBJECTID+ before any response that is affected by the activation:

```
* ENABLED OBJECTID+
```

Once activated, the OBJECTID+ extension remains active for the duration of the IMAP session. Activation MUST NOT be reversed.

Once OBJECTID+ is activated, the server MUST use the compound OBJECTID response code (Section 3) in place of the MAILBOXID response code in all subsequent SELECT, EXAMINE, CREATE, and RENAME responses.

3. OBJECTID Compound Format

The OBJECTID+ extension introduces the compound OBJECTID format, which bundles multiple identifiers into a parenthesized list of key-value pairs.

Each key identifies the type of object identifier (e.g., MAILBOXID, ACCOUNTID, EMAILID, THREADID), and each value is the corresponding ObjectID. Keys that the server does not support or that are not applicable in a given context are simply omitted from the response. An empty compound response "OBJECTID ()" is valid and indicates that the server supports the OBJECTID+ extension but does not have any identifiers to return in this context.

Once OBJECTID+ has been activated, the compound OBJECTID format is used as a response code in SELECT and EXAMINE untagged OK responses, as a response code in tagged OK responses to CREATE and RENAME, as a STATUS attribute, and as a FETCH data item.

The contents of the compound OBJECTID vary by context:

* For mailbox context (SELECT, EXAMINE, CREATE, RENAME, STATUS): the server SHOULD include MAILBOXID and ACCOUNTID.

* For message context (FETCH): the server SHOULD include EMAILID and THREADID. ACCOUNTID is not included in FETCH OBJECTID responses because the account context is already established by the SELECT or EXAMINE response for the current mailbox.

Identifiers that the server does not support are omitted rather than returned as NIL. This allows the compound format to self-describe the server's capabilities without requiring clients to handle placeholder values.

Clients MUST ignore any unrecognised key-value pairs in a compound OBJECTID response. This allows future extensions to add new identifier types without breaking existing clients.

3.1. Relationship to Individual Attributes

The OBJECTID compound is functionally equivalent to requesting each of its constituent identifiers individually. A server MUST return the same values for identifiers whether they are requested individually or as part of an OBJECTID compound. For example, the MAILBOXID returned within an OBJECTID STATUS response MUST be identical to the MAILBOXID returned when requested as a standalone STATUS attribute.

The OBJECTID compound is provided as a convenience for clients that wish to retrieve all available identifiers in a single request without enumerating each attribute separately.

4. ACCOUNTID Object Identifier

The ACCOUNTID is a server-allocated identifier that specifies the account to which a mailbox belongs. When used in conjunction with MAILBOXID, the ACCOUNTID provides complete disambiguation of mailboxes in environments where multiple accounts are accessible through a single IMAP session.

The ACCOUNTID is represented as an opaque string using the same character set and syntactic constraints as other object identifiers defined in this specification (see Section 10).

The server MUST return the same ACCOUNTID for all mailboxes that belong to the same account. Conversely, the server MUST NOT return the same ACCOUNTID for mailboxes that belong to different accounts, even if accessed within the same IMAP session.

When a server advertises both JMAPACCESS and OBJECTID+, the ACCOUNTID for each mailbox MUST correspond to the JMAP accountId for that account (see Section 9).

When a mailbox is accessed exclusively through IMAP and does not have a corresponding representation in JMAP, the server MAY still assign an ACCOUNTID to maintain consistency in the IMAP representation. However, such ACCOUNTIDs need not correspond to any JMAP account identifier.

The ACCOUNTID is conceptually immutable for a given account within an IMAP session. However, if the underlying account is deleted or the user's access to that account is revoked, the associated mailboxes will no longer be accessible via IMAP, and their ACCOUNTIDs become irrelevant.

5. MAILBOXID Object Identifier

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

This document relaxes the uniqueness requirement from [RFC8474], which required MAILBOXID to be unique across the entire server; MAILBOXID is now only required to be unique within the scope of a single ACCOUNTID.

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

The server MUST NOT report the same (ACCOUNTID, MAILBOXID) pair for two different 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 SHOULD 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).

When OBJECTID+ has been activated (Section 2.2), the server returns MAILBOXID within the compound OBJECTID response code for SELECT, EXAMINE, CREATE, and RENAME commands, and within the compound OBJECTID STATUS attribute. Servers that also advertise the OBJECTID capability continue to support the standalone MAILBOXID attribute as defined in [RFC8474].

6. EMAILID Object Identifier and THREADID Correlator

6.1. 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.

EMAILID uniqueness is scoped to a single ACCOUNTID; the same EMAILID value MAY appear in different accounts referring to different messages.

The server MUST return the same EMAILID for the same {name, uidvalidity, uid} triple; hence, EMAILID is immutable.

Messages with the same EMAILID MUST have identical immutable content. Messages with identical content SHOULD have the same EMAILID, but the server is not required to detect content duplication.

A COPY or MOVE command [RFC6851] is allowed to create a new EMAILID for the destination message. The server SHOULD preserve the EMAILID when the source and destination mailboxes have the same ACCOUNTID, but is not required to do so.

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.

6.2. 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 uniqueness is scoped to a single ACCOUNTID, the same as EMAILID.

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 does not support THREADID, it omits THREADID from the compound OBJECTID response in FETCH. A SEARCH for THREADID MUST NOT match any messages when the server does not support THREADID.

Within a compound OBJECTID FETCH response, the server MUST NOT return the same ObjectID value as both the EMAILID and the THREADID for different messages. 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 collisions.

Servers that also advertise the OBJECTID capability continue to support the standalone EMAILID and THREADID FETCH data items as defined in [RFC8474].

7. OBJECTID+ Extensions to Existing Commands

7.1. OBJECTID Parameter on SELECT and EXAMINE

This document extends SELECT and EXAMINE to accept an OBJECTID parameter in the optional parameters list, as defined in [RFC4466].

The OBJECTID parameter has two forms:

1. Without arguments: SELECT "mailbox" (OBJECTID) activates the OBJECTID+ extension (Section 2.2) and requests the compound OBJECTID response code in place of the MAILBOXID response code.

2. With arguments: SELECT "mailbox" (OBJECTID (MAILBOXID id ACCOUNTID id)) additionally requests that the server select the mailbox identified by the given MAILBOXID and ACCOUNTID rather than by name. The mailbox name serves as a fallback if no mailbox matches the given identifiers.

In the second form, the parenthesized list after OBJECTID contains the same key-value pairs that the server returns in its compound OBJECTID response (Section 3). The client SHOULD include all identifiers that the server provided in the most recent compound OBJECTID response for the mailbox.

When the server receives the second form, it MUST attempt to locate a mailbox matching the provided identifiers. If a match is found, the server selects that mailbox regardless of whether the mailbox name in the command still refers to it. If no match is found, the server falls back to selecting the mailbox by name, following the normal SELECT semantics.

This mechanism allows clients to reliably reselect a mailbox after it has been renamed by another client, following the same pattern as the Sieve :mailboxid extension in [RFC9042].

Example (activation only, no ID-based selection):

```
C: 27 select "foo" (OBJECTID)
S: * ENABLED OBJECTID+
[...]
S: * OK [OBJECTID (MAILBOXID F2212ea87-6097-4256-9d51-71338625 \
        ACCOUNTID u1a48e8e3)] Ok
[...]
S: 27 OK [READ-WRITE] Completed
```

Example (ID-based selection after a rename):

```
C: 28 select "foo" (OBJECTID (MAILBOXID \
        F2212ea87-6097-4256-9d51-71338625 \
        ACCOUNTID u1a48e8e3))
[...]
S: * OK [OBJECTID (MAILBOXID F2212ea87-6097-4256-9d51-71338625 \
        ACCOUNTID u1a48e8e3)] Ok
[...]
S: 28 OK [READ-WRITE] Completed
```

Here the mailbox was previously named "foo" but may have been renamed. The server locates it by MAILBOXID and ACCOUNTID regardless of its current name.

Example (ID-based selection, fallback to name):

```
C: 29 select "foo" (OBJECTID (MAILBOXID \
        Fno-longer-exists ACCOUNTID u1a48e8e3))
[...]
S: * OK [OBJECTID (MAILBOXID F9999new-id-for-foo \
        ACCOUNTID u1a48e8e3)] Ok
[...]
S: 29 OK [READ-WRITE] Completed
```

The MAILBOXID did not match any mailbox, so the server fell back to selecting "foo" by name. The response contains the actual OBJECTID of the selected mailbox.

Example (shared mailbox with different ACCOUNTID):

```
C: 30 select "shared/team"
[...]
S: * OK [OBJECTID (MAILBOXID F8839dca12-3ef8-4a72-b63d-54f9e8a1 \
        ACCOUNTID u2b59f9f4)] Ok
[...]
S: 30 OK [READ-WRITE] Completed
```

Note that in this example, the server does not send ENABLED again because the extension was already activated. The shared mailbox has a different ACCOUNTID, indicating it belongs to a different account.

7.2. OBJECTID Response Code for CREATE

When OBJECTID+ has been activated (Section 2.2), the server MUST use the compound OBJECTID response code instead of MAILBOXID in the tagged OK response to successful CREATE commands.

Example:

```
C: 3 create foo
S: 3 OK [OBJECTID (MAILBOXID \
        F2212ea87-6097-4256-9d51-71338625 \
        ACCOUNTID u1a48e8e3)] Completed
C: 4 create bar
S: 4 OK [OBJECTID (MAILBOXID \
        F6352ae03-b7f5-463c-896f-d8b48ee3 \
        ACCOUNTID u1a48e8e3)] Completed
C: 5 create shared/team
S: 5 OK [OBJECTID (MAILBOXID \
        F8839dca12-3ef8-4a72-b63d-54f9e8a1 \
        ACCOUNTID u2b59f9f4)] Completed
```

7.3. OBJECTID Response Code for RENAME

When OBJECTID+ has been activated (Section 2.2), the server MUST include the compound OBJECTID response code in the tagged OK response to successful RENAME commands.

The MAILBOXID in the response SHOULD be the same as the source mailbox when the rename preserves all mailbox invariants. The ACCOUNTID reflects the account to which the mailbox belongs after the rename.

When a mailbox is renamed within the same account, the server SHOULD return the same MAILBOXID and ACCOUNTID as the source mailbox.

When a mailbox is renamed across account boundaries (for example, from a personal namespace to a shared namespace belonging to a different account), the server MAY return a different ACCOUNTID, a different MAILBOXID, or both, reflecting the new account context and any server-specific identifier allocation policy.

Example (local rename, identifiers preserved):

```
C: 8 rename foo renamed
S: 8 OK [OBJECTID (MAILBOXID \
        F2212ea87-6097-4256-9d51-71338625 \
        ACCOUNTID u1a48e8e3)] Completed
```

Example (cross-account rename, new identifiers issued):

```
C: 13 rename bar "Other Users.shared.bar"
S: 13 OK [OBJECTID (MAILBOXID \
        Fa77c2e19-84d3-4b0f-9e12-67df5c8a \
        ACCOUNTID u2b59f9f4)] Completed
```

7.4. OBJECTID Attribute for STATUS

The OBJECTID STATUS attribute requests the compound OBJECTID response, which includes the MAILBOXID and ACCOUNTID for the queried mailbox (when supported by the server).

Syntax: "OBJECTID"

Requesting the OBJECTID STATUS attribute activates the OBJECTID+ extension (Section 2.2).

Example:

```
C: 6 status foo (objectid)
S: * ENABLED OBJECTID+
S: * STATUS foo (OBJECTID (MAILBOXID \
       F2212ea87-6097-4256-9d51-71338625 \
       ACCOUNTID u1a48e8e3))
S: 6 OK Completed

C: 7 status bar (objectid)
S: * STATUS bar (OBJECTID (MAILBOXID \
       F6352ae03-b7f5-463c-896f-d8b48ee3 \
       ACCOUNTID u1a48e8e3))
S: 7 OK Completed

C: 8 status shared/team (objectid)
S: * STATUS shared/team (OBJECTID (MAILBOXID \
       F8839dca12-3ef8-4a72-b63d-54f9e8a1 \
       ACCOUNTID u2b59f9f4))
S: 8 OK Completed
```

Servers that also advertise the OBJECTID capability continue to support the standalone MAILBOXID STATUS attribute as defined in [RFC8474].

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 (objectid))
S: * ENABLED OBJECTID+
S: * LIST (\HasNoChildren) "." INBOX
S: * STATUS INBOX (OBJECTID (MAILBOXID \
       Ff8e3ead4-9389-4aff-adb1-d8d89efd8cbf \
       ACCOUNTID u1a48e8e3))
S: * LIST (\HasNoChildren) "." bar
S: * STATUS bar (OBJECTID (MAILBOXID \
       F6352ae03-b7f5-463c-896f-d8b48ee3 \
       ACCOUNTID u1a48e8e3))
S: * LIST (\HasNoChildren) "." "Other Users.other.sub.folder"
S: * STATUS "Other Users.other.sub.folder" (OBJECTID ( \
       MAILBOXID F8839dca12-3ef8-4a72-b63d-54f9e8a1 \
       ACCOUNTID u2b59f9f4))
S: 11 OK Completed (0.001 secs 3 calls)
```

This example demonstrates how clients can efficiently retrieve object identifiers for multiple mailboxes, including mailboxes belonging to different accounts, using the extended LIST command with STATUS return option.

Example:

```
C: 11 list "" "*" return (status (objectid))
S: * ENABLED OBJECTID+
S: * LIST (\HasNoChildren) "." INBOX
S: * STATUS INBOX (OBJECTID (MAILBOXID \
        Ff8e3ead4-9389-4aff-adb1-d8d89efd8cbf \
        ACCOUNTID u1a48e8e3))
S: * LIST (\HasNoChildren) "." bar
S: * STATUS bar (OBJECTID (MAILBOXID \
        F6352ae03-b7f5-463c-896f-d8b48ee3 \
        ACCOUNTID u1a48e8e3))
S: * LIST (\HasNoChildren) "." "Other Users.other.sub.folder"
S: * STATUS "Other Users.other.sub.folder" (OBJECTID ( \
        MAILBOXID F8839dca12-3ef8-4a72-b63d-54f9e8a1 \
        ACCOUNTID u2b59f9f4))
S: 11 OK Completed (0.001 secs 3 calls)
```

This example demonstrates how clients can efficiently retrieve object identifiers for multiple mailboxes, including mailboxes belonging to different accounts, using the extended LIST command with STATUS return option.

7.5. OBJECTID Data Item for FETCH

The OBJECTID FETCH data item causes the server to return a compound OBJECTID response containing the EMAILID and, if supported, the THREADID for each message.

Syntax: "OBJECTID"

Requesting the OBJECTID FETCH data item activates the OBJECTID+ extension (Section 2.2).

ACCOUNTID is not included in the FETCH OBJECTID response because the account context is already established by the SELECT or EXAMINE response for the current mailbox.

Example:

```
C: 30 fetch 1:* (objectid)
S: * ENABLED OBJECTID+
S: * 1 FETCH (OBJECTID (EMAILID M6d99ac3275bb4e \
        THREADID T64b478a75b7ea9))
S: * 2 FETCH (OBJECTID (EMAILID M5fdc09b49ea703 \
        THREADID T11863d02dd95b5))
S: 30 OK Completed (0.000 sec)
```

Example (no THREADID support):

```
C: 31 fetch 1:* (objectid)
S: * 1 FETCH (OBJECTID (EMAILID M00000001))
S: * 2 FETCH (OBJECTID (EMAILID M00000002))
S: 31 OK Completed (0.000 sec)
```

Example (server supports no message identifiers):

```
C: 32 fetch 1:* (objectid)
S: * 1 FETCH (OBJECTID ())
S: * 2 FETCH (OBJECTID ())
S: 32 OK Completed (0.000 sec)
```

Servers that also advertise the OBJECTID capability continue to support the individual EMAILID and THREADID FETCH data items as defined in [RFC8474].

8. 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.

When using the MULTISEARCH extension defined in [RFC7377] to search across multiple mailboxes, clients SHOULD only search for EMAILID or THREADID across mailboxes that share the same ACCOUNTID. Since object identifiers are only guaranteed to be unique within the scope of a single ACCOUNTID, searching across mailboxes with different ACCOUNTIDs may produce incorrect results if identifiers from different accounts happen to collide.

Example:

```
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)
```

9. Additional Conditions for JMAPACCESS

The JMAPACCESS capability and GETJMAPACCESS command are defined in [RFC9698]. This document updates those semantics: when a server advertises both JMAPACCESS and OBJECTID+, it additionally asserts that the IMAP ACCOUNTID for each mailbox corresponds directly to the JMAP accountId for that account, as defined in Section 1.6.2 of [RFC8620].

A server that advertises both JMAPACCESS and OBJECTID+ is not required to also advertise OBJECTID ([RFC8474]); OBJECTID+ is sufficient to satisfy the capability prerequisite for JMAPACCESS.

Clients that encounter JMAPACCESS without OBJECTID+ should interpret it as defined in [RFC9698].

10. 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], IMAP ABNF extensions [RFC4466], and IMAP ENABLE [RFC5161] 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" / "OBJECTID+"

enable-data =/ "OBJECTID+"
        ; extends the enable-data production from [RFC5161]

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

objectid-key = "MAILBOXID" / "ACCOUNTID" / "EMAILID" / "THREADID"
             / atom
        ; future extensions may define additional keys
        ; clients MUST ignore unrecognised keys

objectid-kvpair = objectid-key SP objectid

objectid-compound = "OBJECTID" SP "(" [objectid-kvpair
```

```
           *(SP objectid-kvpair)] ")"
           ; space-separated key-value pairs of identifiers
           ; keys not supported by the server are omitted
           ; an empty list "OBJECTID ()" is valid

   ; --- OBJECTID+ extensions to SELECT/EXAMINE ---

   select-param =/ "OBJECTID" [SP "(" objectid-kvpair
           *(SP objectid-kvpair) ")"]
           ; without arguments: activation only
           ; with arguments: ID-based mailbox selection
           ;   with fallback to the mailbox name

   ; --- OBJECTID+ extensions to FETCH ---

   fetch-att =/ "OBJECTID"

   msg-att-static =/ objectid-compound

   ; --- OBJECTID+ extensions to STATUS ---

   status-att =/ "OBJECTID"

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

   ; --- OBJECTID+ response code ---

   resp-text-code =/ objectid-compound

   ; --- OBJECTID+ extensions to SEARCH ---

   search-key =/ "EMAILID" SP objectid
           ; matches messages whose EMAILID is exactly
           ; the specified ObjectID

   search-key =/ "THREADID" SP objectid
           ; matches messages whose THREADID is exactly
           ; the specified ObjectID
```

11. Implementation Considerations

11.1. 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.

11.2. Interaction with Special Cases

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

It is advisable (though not required) to have object identifier values be globally unique as an implementation convenience. A proxy that aggregates multiple independent backend servers MUST return a different ACCOUNTID for each set of mailboxes served by different backends, unless it can guarantee that all object identifiers are unique across those backends. This ensures that clients can rely on the combination of ACCOUNTID and any other object identifier being unique within the IMAP session, even when the backend servers independently assign identifiers that might otherwise collide.

11.3. Client Usage

Servers that implement both [RFC6154] 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.

Clients that support both IMAP and JMAP SHOULD use the ACCOUNTID when available to maintain accurate mappings between IMAP mailboxes and JMAP Mailbox objects. This is particularly important for clients that use JMAP Email Delivery Push notifications, as these notifications include the accountId property. By correlating the accountId from a push notification with the ACCOUNTID, clients can efficiently determine which IMAP mailbox corresponds to a newly delivered message without requiring additional synchronization operations.

11.4. Interaction with the OBJECTID Capability

A server MAY advertise both OBJECTID and OBJECTID+ to provide backward compatibility with clients that only support [RFC8474]. When both capabilities are advertised, the server MUST behave as defined in [RFC8474] until the client activates OBJECTID+ (Section 2.2). Once OBJECTID+ has been activated, the server MUST use compound OBJECTID response codes in place of MAILBOXID response codes for CREATE, RENAME, SELECT, and EXAMINE commands, and MUST support the OBJECTID STATUS attribute and FETCH data item.

A server that advertises only OBJECTID+ is not required to support the individual MAILBOXID, EMAILID, or THREADID attributes defined in [RFC8474]. Such a server uses exclusively the compound OBJECTID format defined in this specification.

11.5. Interaction with IMAP4rev2

This specification is written in terms of [RFC3501] (IMAP4rev1) but applies equally to [RFC9051] (IMAP4rev2). IMAP4rev2 incorporates the ENABLE command and the MOVE extension natively, so no separate capability negotiation is needed for those features.

The formal syntax in this document extends the ABNF productions defined in [RFC3501]. Servers implementing IMAP4rev2 SHOULD apply the same extensions to the corresponding productions in [RFC9051].

11.6. Interaction with MOVE

The MOVE command [RFC6851] atomically moves messages between mailboxes. As specified in Section 6, MOVE is allowed to create new EMAILIDs and THREADIDs for the destination messages. The server SHOULD preserve the EMAILID when the source and destination mailboxes share the same ACCOUNTID, but is not required to do so.

The MOVE command does not receive an OBJECTID response code. The COPYUID response code [RFC4315] already provides the UID mapping between source and destination.

11.7. Interaction with NAMESPACE

The NAMESPACE extension [RFC2342] exposes that a single IMAP connection may provide access to mailboxes from different namespaces, including personal, other users', and shared namespaces.

The ACCOUNTID returned for a mailbox SHOULD reflect the account that owns the mailbox data, not the account of the authenticated user accessing it. For example:

* Mailboxes in the personal namespace have the authenticated user's ACCOUNTID.

* Mailboxes in the "Other Users" namespace that belong to a different user SHOULD have that other user's ACCOUNTID.

* Mailboxes in a shared namespace SHOULD have the ACCOUNTID of the account that owns the shared data.

This ensures that ACCOUNTID provides meaningful account-level disambiguation and, when JMAPACCESS is advertised, correctly correlates with the JMAP accountId that owns the corresponding Mailbox objects.

11.8. Interaction with UIDONLY

When the UIDONLY extension [RFC9586] is active, FETCH responses are replaced with UIDFETCH responses. The OBJECTID FETCH data item works identically in UIDFETCH responses. A server that supports both OBJECTID+ and UIDONLY MUST include the OBJECTID data item in UIDFETCH responses when requested.

11.9. Interaction with SORT and THREAD

The THREAD command defined in [RFC5256] computes thread relationships algorithmically based on message headers and returns a thread structure for display purposes. The THREADID defined in this document is a persistent identifier assigned by the server to group related messages.

THREADID and the THREAD command are independent. A server MAY use different algorithms for THREAD responses and THREADID assignment, and the thread groupings need not correlate. Clients MUST NOT assume that messages sharing a THREADID will appear in the same thread structure returned by the THREAD command, or vice versa.

11.10. 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 [RFC3501]. 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.

12. Future Considerations

This extension is intentionally defined to be compatible with the data model in JMAP for Mail.

A future extension to the Sieve :mailboxid extension [RFC9042] could add ACCOUNTID support for multi-account environments.

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

13. IANA Considerations

13.1. IMAP Capabilities Registry

IANA is requested to add the following entry to the "IMAP Capabilities" registry located at https://www.iana.org/assignments/imap-capabilities (https://www.iana.org/assignments/imap-capabilities):

Capability	Reference
OBJECTID+	This document

Table 1

IANA is requested to update the reference for the existing "JMAPACCESS" entry in the "IMAP Capabilities" registry from [RFC9698] to this document.

The existing "OBJECTID" entry registered by [RFC8474] remains unchanged. Servers MAY advertise OBJECTID alongside OBJECTID+ for backward compatibility as described in this document.

13.2. IMAP Response Codes Registry

IANA is requested to add the following entry to the "IMAP Response Codes" registry located at https://www.iana.org/assignments/imap-response-codes (https://www.iana.org/assignments/imap-response-codes):

Response Code	Reference
OBJECTID	This document

Table 2

The existing "MAILBOXID" entry in the "IMAP Response Codes" registry, registered by [RFC8474], remains unchanged.

14. Security Considerations

14.1. Object Identifier Generation

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 Section 11 of [RFC3501].

14.2. Account Identifier Exposure

The ACCOUNTID reveals information about the account structure of the server and which mailboxes belong to which accounts. While this information is generally not considered sensitive in the context of an authenticated IMAP session, servers that wish to minimize information disclosure MAY choose to generate account identifiers using unpredictable values (such as UUIDs) rather than sequential numbers or other patterns that might reveal information about account creation order or the total number of accounts on the server.

14.3. Cross-Account Information Leakage

Servers MUST ensure that the ACCOUNTID mechanism does not inadvertently grant users access to information about accounts they are not authorized to access. In particular, servers MUST NOT return account identifiers for accounts that the authenticated user does not have permission to access, even if such accounts exist on the server.

14.4. Consistency with JMAP Authentication

A server MUST NOT advertise JMAPACCESS unless the authentication credentials used for the IMAP session are sufficient to also authenticate via JMAP. Inconsistencies in authentication or authorization between IMAP and JMAP could lead to situations where a client receives account identifiers that it cannot subsequently use to access the corresponding JMAP resources, potentially revealing the existence of accounts the user cannot access.

The JMAP session URL returned by GETJMAPACCESS is available to any authenticated IMAP client. This reveals that a JMAP server exists for the user, but since an authenticated client with valid credentials could discover this independently via [RFC8620] Section 2.2, this does not represent a meaningful increase in exposure.

14.5. Privacy in Multi-Tenant Environments

In multi-tenant or hosted environments, servers SHOULD generate account identifiers in a manner that does not reveal relationships between accounts or organizational structures that users should not be aware of. For example, if multiple accounts belong to the same organization, the account identifier generation mechanism should not use patterns that would allow users to infer these relationships unless such information is explicitly intended to be visible.

15. References

15.1. 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/rfc/rfc2119>.

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

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

[RFC5161] Gulbrandsen, A., Ed. and A. Melnikov, Ed., "The IMAP ENABLE Extension", RFC 5161, DOI 10.17487/RFC5161, March 2008, <https://www.rfc-editor.org/rfc/rfc5161>.

[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/rfc/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/rfc/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/rfc/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/rfc/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/rfc/rfc8174>.

[RFC8474] Gondwana, B., Ed., "IMAP Extension for Object Identifiers", RFC 8474, DOI 10.17487/RFC8474, September 2018, <https://www.rfc-editor.org/rfc/rfc8474>.

[RFC8620] Jenkins, N. and C. Newman, "The JSON Meta Application Protocol (JMAP)", RFC 8620, DOI 10.17487/RFC8620, July 2019, <https://www.rfc-editor.org/rfc/rfc8620>.

[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/rfc/rfc9051>.

[RFC9698] Gulbrandsen, A. and B. Gondwana, "The JMAPACCESS Extension for IMAP", RFC 9698, DOI 10.17487/RFC9698, January 2025, <https://www.rfc-editor.org/rfc/rfc9698>.

15.2. Informative References

[RFC2342] Gahrns, M. and C. Newman, "IMAP4 Namespace", RFC 2342, DOI 10.17487/RFC2342, May 1998, <https://www.rfc-editor.org/rfc/rfc2342>.

[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/rfc/rfc4122>.

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

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

[RFC6154] Leiba, B. and J. Nicolson, "IMAP LIST Extension for Special-Use Mailboxes", RFC 6154, DOI 10.17487/RFC6154, March 2011, <https://www.rfc-editor.org/rfc/rfc6154>.

[RFC7377] Leiba, B. and A. Melnikov, "IMAP4 Multimailbox SEARCH Extension", RFC 7377, DOI 10.17487/RFC7377, October 2014, <https://www.rfc-editor.org/rfc/rfc7377>.

[RFC9042] Gondwana, B., Ed., "Sieve Email Filtering: Delivery by MAILBOXID", RFC 9042, DOI 10.17487/RFC9042, June 2021, <https://www.rfc-editor.org/rfc/rfc9042>.

[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/rfc/rfc9586>.

Appendix A. Ideas for Implementing Object Identifiers

Ideas for calculating account identifiers:

* Universally Unique Identifier (UUID) [RFC4122]

* Server-assigned sequence number (guaranteed not to be reused)

* Hash of the JMAP accountId (if JMAP integration is provided)

Ideas for calculating mailbox identifiers:

* 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.

Appendix B. Changes from RFC 8474 and RFC 9698

This document obsoletes [RFC8474], updates [RFC9698], and introduces the following changes:

The OBJECTID+ capability and extension is defined as an independent extension that may be advertised alongside or in place of the OBJECTID capability from [RFC8474]. Servers that advertise only OBJECTID+ are not required to support the individual MAILBOXID, EMAILID, or THREADID attributes defined in [RFC8474].

The compound OBJECTID response format is introduced, using key-value pairs where unsupported identifiers are omitted rather than returned as NIL. This compound format is used uniformly for SELECT, EXAMINE, CREATE, RENAME, STATUS, and FETCH responses.

The ACCOUNTID identifier is defined for account-level context, enabling disambiguation of mailboxes in environments where multiple accounts are accessible through a single IMAP session.

The RENAME command now returns an OBJECTID response code containing the identifiers of the renamed mailbox, which is new behavior not present in [RFC8474].

The OBJECTID SELECT/EXAMINE parameter is introduced, supporting both activation of the OBJECTID+ extension and identifier-based mailbox selection with fallback to the mailbox name.

An implicit activation model replaces mandatory ENABLE: OBJECTID+ is activated when the client uses any OBJECTID+-specific feature (OBJECTID in SELECT, EXAMINE, FETCH, or STATUS, or ENABLE OBJECTID+), with an untagged ENABLED response to signal activation.

The OBJECTID FETCH data item provides EMAILID and THREADID in compound form. The OBJECTID STATUS attribute provides MAILBOXID and ACCOUNTID in compound form.

The JMAPACCESS capability and GETJMAPACCESS command defined in [RFC9698] are updated: when a server advertises both JMAPACCESS and OBJECTID+, it additionally asserts that IMAP ACCOUNTIDs correspond directly to JMAP accountIds.

Security considerations are added for account identifier exposure, cross-account information leakage, JMAP authentication consistency, and privacy in multi-tenant environments.

IANA registrations are updated to include the OBJECTID+ capability, JMAPACCESS capability, and OBJECTID response code.

Appendix C. Acknowledgements

The authors would like to thank the members of the IETF mailmaint working group for their contributions to this specification.

Appendix D. Changes

[[This section to be removed by RFC Editor]]

draft-ietf-mailmaint-imap-objectid-bis-03

* Relaxed uniqueness scope for MAILBOXID, EMAILID, and THREADID from server-wide ([RFC8474]) to within a single ACCOUNTID

* Updated JMAPACCESS ([RFC9698]): when advertised with OBJECTID+, server additionally asserts ACCOUNTID corresponds to JMAP accountId

draft-ietf-mailmaint-imap-objectid-bis-02

* Extended SELECT/EXAMINE OBJECTID parameter to support ID-based mailbox selection with fallback to mailbox name, following the pattern established by [RFC9042]

* Removed restatement of [RFC8474] behavior for MAILBOXID, EMAILID, and THREADID; this document now references [RFC8474] for base OBJECTID behavior and focuses on OBJECTID+ extensions

* Reduced introduction length

* Clients MUST ignore unrecognised key-value pairs in compound OBJECTID responses (extensibility)

* ABNF objectid-key extended to allow future keys via atom

* Clarified COPY/MOVE EMAILID semantics: COPY/MOVE MAY create new EMAILIDs; same EMAILID MUST have same content; same content SHOULD have same EMAILID

draft-ietf-mailmaint-imap-objectid-bis-01

* Replaced mandatory ENABLE with implicit activation model: OBJECTID+ is activated when the client uses any OBJECTID+-specific feature (OBJECTID in SELECT/FETCH/STATUS, or ENABLE OBJECTID+)

* Changed compound OBJECTID format from positional with NIL to key-value pairs where unsupported identifiers are omitted

* Removed ACCOUNTID from FETCH OBJECTID (redundant with SELECT)

* Removed standalone ACCOUNTID STATUS attribute, FETCH data item, and SEARCH filter; ACCOUNTID is only available through compound OBJECTID responses

* Added OBJECTID parameter for SELECT/EXAMINE as an activation trigger

* MAILBOXID reverted to single objectid format in individual items (compatible with RFC 8474)

* Renamed capability from OBJECTIDBIS to OBJECTID+

* Clarified that object identifiers only need to be unique within the scope of a single ACCOUNTID; proxies MUST assign different ACCOUNTIDs for different backends

draft-ietf-mailmaint-imap-objectid-bis-00

* Initial version

Authors' Addresses

Bron Gondwana
Fastmail
Level 2, 114 William St
Melbourne VIC 3000
Australia
Email: brong@fastmailteam.com
URI: https://www.fastmail.com

Mauro De Gennaro
Stalwart Labs LLC
1309 Coffeen Avenue, Suite 1200
Sheridan, WY 82801
United States of America
Email: mauro@stalw.art
URI: https://stalw.art