mailmaint
Internet-Draft
Obsoletes: RFC8474 (if approved)
Intended status: Standards Track
Expires: 14 August 2026

B. Gondwana
Fastmail
M. De Gennaro
Stalwart Labs

10 February 2026

IMAP Extension for Object Identifiers
draft-gondwana-degennaro-imap-objectid-bis-00

Abstract

This document updates [RFC3501] (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.

This document obsoletes [RFC8474] by making all object identifier types optional, introducing the OBJECTIDBIS capability with mandatory ENABLE negotiation, and extending the framework to include account-level context through the ACCOUNTID identifier. The ENABLE requirement ensures that servers can implement the extension without altering the IMAP grammar for clients that do not understand the extension. The account identifier extension enables IMAP servers to provide account-level context for mailboxes when multiple accounts are accessible through a single IMAP connection, facilitating interoperability with the JSON Meta Application Protocol (JMAP) in multi-account scenarios.

Table of Contents

1. Introduction

IMAP stores are often used by many clients. Each client may cache data from the server so that it does not 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.

Additionally, this document introduces the ACCOUNTID object identifier, which specifies the account to which a mailbox or message belongs. This is particularly relevant for environments where IMAP mailboxes include shared mailboxes from multiple JMAP accounts, as defined in [RFC8620].

All object identifier types defined in this specification are optional. A server that supports this extension MAY return NIL for any object identifier type that it does not implement. This design permits partial implementation, allowing servers to provide identifiers for the types they support without requiring full implementation of all identifier types.

This extension requires the use of the ENABLE command, as defined in [RFC5161], to activate the extended response syntax. The ENABLE requirement ensures that servers can implement the extension without introducing incompatible changes to the IMAP grammar for clients that do not understand the extension. Until the client issues ENABLE OBJECTIDBIS, the server MUST NOT return any of the extended response formats defined in this document.

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

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

A server MAY advertise both the "OBJECTID" capability defined in [RFC8474] and the "OBJECTIDBIS" capability defined in this document. When both capabilities are advertised, the server MUST conform to the behaviour specified in [RFC8474] unless and until the client issues ENABLE OBJECTIDBIS, at which point the server MUST conform to the behaviour specified in this document.

If the client has not issued ENABLE OBJECTIDBIS, the server MUST NOT return any of the extended response formats defined in this document, including OBJECTID response codes, OBJECTID status attributes, OBJECTID fetch data items, or ACCOUNTID attributes. This restriction ensures that the IMAP grammar remains unaltered for clients that do not understand the extension.

2.1. ENABLE Requirement

The OBJECTIDBIS extension MUST be activated by the client through the ENABLE command as defined in [RFC5161]. A client that wishes to use the facilities defined in this document MUST issue ENABLE OBJECTIDBIS before relying on any of the extended responses.

Example:

```
C: 7 enable objectidbis
S: * ENABLED OBJECTIDBIS
S: 7 OK Completed
```

After the server confirms that OBJECTIDBIS has been enabled, the extended response formats defined in this document become active for the remainder of the IMAP session.

3. OBJECTID Compound Identifier

The OBJECTID is a compound data item that aggregates all object identifiers relevant to a given context into a single response element. The composition of the OBJECTID depends on whether it pertains to a mailbox or a message.

3.1. Mailbox Context

When used in the context of a mailbox (response codes for CREATE, SELECT, and EXAMINE; STATUS attributes), the OBJECTID compound contains the following elements:

Syntax: "OBJECTID" SP "(" "MAILBOXID" SP objectid-or-nil SP \
"ACCOUNTID" SP objectid-or-nil ")"

The MAILBOXID element contains the mailbox identifier, and the ACCOUNTID element contains the account identifier. Either element MAY be NIL if the server does not support the corresponding identifier type.

3.2. Message Context

When used in the context of a message (FETCH data items), the OBJECTID compound contains the following elements:

Syntax: "OBJECTID" SP "(" "EMAILID" SP objectid-or-nil SP \
"ACCOUNTID" SP objectid-or-nil SP \ "THREADID" SP objectid-or-nil ")"

The EMAILID element contains the email identifier, the ACCOUNTID element contains the account identifier, and the THREADID element contains the thread identifier. Any element MAY be NIL if the server does not support the corresponding identifier type.

3.3. 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 or message 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 8).

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

When a server advertises the "JMAPACCESS" capability as defined in [RFC9698], it MUST ensure that the ACCOUNTID returned via IMAP matches the accountId property of the corresponding account in JMAP, as defined in Section 1.6.2 of [RFC8620]. This correspondence is essential for clients to correlate mailboxes and messages across the two protocols.

When a mailbox or message 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.

The server MAY return NIL for ACCOUNTID if it does not support account identifiers.

5. 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).

The server MAY return NIL for MAILBOXID if it does not support mailbox identifiers for a given mailbox (for example, for virtual folders).

5.1. New Response Code for CREATE

This document extends the CREATE command to include object identifiers in the response code on successful mailbox creation.

When OBJECTIDBIS has been enabled, the server MUST include the OBJECTID response code in the tagged OK response to all successful CREATE commands.

Example:

```
C: 8 create bar
S: 8 OK [OBJECTID (MAILBOXID \
        (F6352ae03-b7f5-463c-896f-d8b48ee3) \
        ACCOUNTID (u1a48e8e3))] Completed
```

Example (MAILBOXID not supported):

```
C: 9 create virtual-folder
S: 9 OK [OBJECTID (MAILBOXID NIL \
        ACCOUNTID (u1a48e8e3))] Completed
```

5.2. New Response Code for RENAME

This document extends the RENAME command to include the OBJECTID response code in the tagged OK response on successful mailbox rename when OBJECTIDBIS has been enabled.

The OBJECTID response code in the RENAME response conveys the object identifiers of the mailbox after the rename operation has been completed. The MAILBOXID and ACCOUNTID returned MAY differ from those of the source mailbox, depending on server implementation and whether the rename operation crosses account boundaries.

When a mailbox is renamed within the same account, the server SHOULD return the same MAILBOXID and ACCOUNTID as the source mailbox, unless the server does not support persistent mailbox identifiers across rename operations.

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: 12 rename foo renamed-foo
S: 12 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
```

In the first example, the mailbox "foo" is renamed to "renamed-foo" within the same account, and the server preserves both the MAILBOXID and the ACCOUNTID. In the second example, the mailbox "bar" is renamed into a shared namespace belonging to a different account, and the server issues both a new MAILBOXID and a new ACCOUNTID reflecting the destination account.

5.3. New OK Untagged Response for SELECT and EXAMINE

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

When OBJECTIDBIS has been enabled, the server MUST return an untagged OK response with the OBJECTID response code on all successful SELECT and EXAMINE commands.

Example:

```
C: 9 select "bar"
[...]
S: * OK [OBJECTID (MAILBOXID \
        (F6352ae03-b7f5-463c-896f-d8b48ee3) \
        ACCOUNTID (u1a48e8e3))] Ok
[...]
S: 9 OK [READ-WRITE] Completed
```

5.4. New Attributes for STATUS

This document adds the MAILBOXID, ACCOUNTID, and OBJECTID attributes to the STATUS command using the extended syntax defined in [RFC4466].

A server that has OBJECTIDBIS enabled MUST support the MAILBOXID, ACCOUNTID, and OBJECTID status attributes.

The MAILBOXID status attribute returns the mailbox identifier for the specified mailbox. The ACCOUNTID status attribute returns the account identifier for the specified mailbox. The OBJECTID status attribute returns both the mailbox identifier and the account identifier as a compound value.

The server MAY return NIL for any identifier that it does not support.

Example:

```
C: 6 status foo (mailboxid)
S: * STATUS foo (MAILBOXID \
       (F2212ea87-6097-4256-9d51-71338625))
S: 6 OK Completed
C: 7 status bar (mailboxid accountid)
S: * STATUS bar (MAILBOXID \
       (F6352ae03-b7f5-463c-896f-d8b48ee3) \
       ACCOUNTID (u1a48e8e3))
S: 7 OK Completed
C: 8 rename foo renamed
S: * OK rename foo renamed
S: 8 OK Completed
C: 9 status renamed (objectid)
S: * STATUS renamed (OBJECTID (MAILBOXID \
       (F2212ea87-6097-4256-9d51-71338625) \
       ACCOUNTID (u1a48e8e3)))
S: 9 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 (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) "." renamed
S: * STATUS renamed (OBJECTID (MAILBOXID \
       (F2212ea87-6097-4256-9d51-71338625) \
       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 4 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.

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.

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.

The server MAY return NIL for EMAILID if it does not support email identifiers (for example, for virtual messages).

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 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 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.

6.3. New Message Data Items in FETCH and UID FETCH Commands

This document defines the following FETCH request 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.

Syntax: "ACCOUNTID"

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

Syntax: "OBJECTID"

The OBJECTID message data item causes the server to return OBJECTID FETCH response data items containing all email-related object identifiers as a compound value.

This document defines the following FETCH response items:

Syntax: "EMAILID" SP objectid-or-nil

The EMAILID response data item contains the server-assigned ObjectID for each message, or NIL if the server does not support email identifiers.

Syntax: "THREADID" SP objectid-or-nil

The THREADID response data item contains the server-assigned ObjectID for the set of related messages to which this message belongs. NIL is returned when the server does not support THREADID calculation.

Syntax: "ACCOUNTID" SP objectid-or-nil

The ACCOUNTID response data item contains the account identifier for the message, or NIL if the server does not support account identifiers.

Syntax: "OBJECTID" SP "(" "EMAILID" SP objectid-or-nil SP \
"ACCOUNTID" SP objectid-or-nil SP \ "THREADID" SP objectid-or-nil ")"

The OBJECTID response data item contains all email-related object identifiers as a compound value.

Example (EMAILID and THREADID):

```
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"
[...]
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 (OBJECTID compound value):

```
C: 30 fetch 1:* (objectid)
S: * 1 FETCH (OBJECTID (EMAILID (M6d99ac3275bb4e) \
        ACCOUNTID (u1a48e8e3) \
        THREADID (T64b478a75b7ea9)))
S: * 2 FETCH (OBJECTID (EMAILID (M5fdc09b49ea703) \
        ACCOUNTID (u1a48e8e3) \
        THREADID (T11863d02dd95b5)))
S: 30 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)
```

Example (OBJECTID with no THREADID support):

```
C: 31 fetch 1:* (objectid)
S: * 1 FETCH (OBJECTID (EMAILID (M00000001) \
        ACCOUNTID (u1a48e8e3) THREADID NIL))
S: * 2 FETCH (OBJECTID (EMAILID (M00000002) \
        ACCOUNTID (u1a48e8e3) THREADID NIL))
S: 31 OK Completed (0.000 sec)
```

Example (no EMAILID support):

```
C: 32 fetch 1:* (objectid)
S: * 1 FETCH (OBJECTID (EMAILID NIL \
        ACCOUNTID (u1a48e8e3) THREADID NIL))
S: 32 OK Completed (0.000 sec)
```

7. New Filters on SEARCH Command

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

The EMAILID and THREADID filters require ENABLE OBJECTIDBIS to be in effect. However, if the server also advertises the OBJECTID capability defined in [RFC8474], it MAY support the EMAILID and THREADID filters without requiring ENABLE OBJECTIDBIS, using the syntax defined in [RFC8474].

The ACCOUNTID filter is only available when OBJECTIDBIS has been enabled.

Syntax: "EMAILID" SP objectid-atom

Messages whose EMAILID is exactly the specified ObjectID.

Syntax: "THREADID" SP objectid-atom

Messages whose THREADID is exactly the specified ObjectID.

Syntax: "ACCOUNTID" SP objectid-atom

Messages whose ACCOUNTID 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)
C: 29 search accountid u1a48e8e3
S: * SEARCH 1 2 3
S: 29 OK Completed (3 msgs in 0.000 secs)
```

8. 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 =/ "OBJECTIDBIS"

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

objectid-atom = 1*255(ALPHA / DIGIT / "_" / "-")
        ; the raw object identifier string
        ; characters are case significant

objectid = "(" objectid-atom ")"
```

```
            ; parenthesized object identifier

   objectid-or-nil = objectid / nil
            ; parenthesized object identifier or NIL

   ; --- FETCH request items ---

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

   ; --- FETCH response items ---

   fetch-emailid-resp = "EMAILID" SP objectid-or-nil

   fetch-threadid-resp = "THREADID" SP objectid-or-nil

   fetch-accountid-resp = "ACCOUNTID" SP objectid-or-nil

   fetch-objectid-email-resp = "OBJECTID" SP "(" \
            "EMAILID" SP objectid-or-nil SP \
            "ACCOUNTID" SP objectid-or-nil SP \
            "THREADID" SP objectid-or-nil ")"
            ; compound email object identifier response

   msg-att-static =/ fetch-emailid-resp / fetch-threadid-resp /
            fetch-accountid-resp / fetch-objectid-email-resp

   ; --- Response codes ---

   resp-text-code =/ "OBJECTID" SP "(" \
            "MAILBOXID" SP objectid-or-nil SP \
            "ACCOUNTID" SP objectid-or-nil ")"
            ; compound mailbox object identifier response code;
            ; returned in tagged OK for CREATE and RENAME,
            ; and in untagged OK for SELECT and EXAMINE

   ; --- SEARCH filters ---

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

   ; --- STATUS attributes ---

   status-att =/ "MAILBOXID" / "ACCOUNTID" / "OBJECTID"

   status-att-val =/ "MAILBOXID" SP objectid-or-nil

   status-att-val =/ "ACCOUNTID" SP objectid-or-nil
```

```
status-att-val =/ "OBJECTID" SP "(" \
        "MAILBOXID" SP objectid-or-nil SP \
        "ACCOUNTID" SP objectid-or-nil ")"
        ; compound mailbox object identifier status value
```

9. Implementation Considerations

9.1. 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.

9.2. 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 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 OBJECTIDBIS capability unless it can guarantee that different objects will never use the same identifiers, even if backend object identifiers collide.

The ACCOUNTID provides additional disambiguation in multi-account environments, ensuring that even if two backend servers use the same MAILBOXID, the combination of ACCOUNTID and MAILBOXID remains unique within the IMAP session.

9.3. 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.

Clients SHOULD be prepared to handle servers that return NIL for any of the object identifier types. In such cases, clients should gracefully degrade functionality to use only the supported object identifier types and fall back to the guarantees of [RFC3501].

9.4. Interaction with the OBJECTID Capability

A server that advertises both the OBJECTID capability defined in [RFC8474] and the OBJECTIDBIS capability defined in this document MUST behave as follows:

* When OBJECTIDBIS has not been enabled, the server MUST conform to the behaviour specified in [RFC8474] for all OBJECTID-related responses. The server MUST NOT return OBJECTID response codes, OBJECTID status attributes, ACCOUNTID attributes, or NIL values for MAILBOXID or EMAILID.

* When OBJECTIDBIS has been enabled, the server MUST conform to the behaviour specified in this document. The server MUST use OBJECTID response codes in place of MAILBOXID response codes for CREATE, SELECT, and EXAMINE commands. The server MUST support ACCOUNTID, MAILBOXID, and OBJECTID as STATUS attributes and FETCH data items.

This design allows servers to support both the original and extended specifications without breaking the IMAP grammar for clients that understand only one of the two extensions.

9.5. 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.

10. Future Considerations

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

11. IANA Considerations

11.1. 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
OBJECTIDBIS	This document

Table 1

The existing "OBJECTID" entry in the "IMAP Capabilities" registry, registered by [RFC8474], remains unchanged. Servers MAY advertise both OBJECTID and OBJECTIDBIS as described in this document.

11.2. 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.

12. Security Considerations

12.1. 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].

12.2. Account Identifier Exposure

The account identifier component of MAILBOXID 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.

12.3. Cross-Account Information Leakage

Servers MUST ensure that the MAILBOXID 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.

12.4. Consistency with JMAP Authentication

When a server advertises both "OBJECTID" (or "OBJECTID=MAILBOXID") and "JMAPACCESS" capabilities, the server MUST ensure that the same authentication credentials used for the IMAP session would grant access to the corresponding JMAP accounts. 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.

12.5. 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.

13. References

13.1. 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>.

[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>.

[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>.

[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/rfc/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/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>.

[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>.

13.2. Informative References

[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>.

[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>.

[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>.

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

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

* Replaced the OBJECTID capability with the OBJECTIDBIS capability, which requires activation through the ENABLE command.

* Introduced the OBJECTID compound response format, which combines multiple object identifiers into a single response element. For mailboxes, the OBJECTID response includes MAILBOXID and ACCOUNTID. For emails, the OBJECTID response includes EMAILID, ACCOUNTID, and THREADID.

* Added the ACCOUNTID object identifier, which specifies the account to which a mailbox or message belongs, enabling correlation with JMAP account identifiers.

* Added ACCOUNTID as a standalone STATUS attribute, FETCH data item, and SEARCH filter.

* Added the OBJECTID response code to the RENAME command, enabling clients to track mailbox identifier changes across rename operations, including renames that cross account boundaries.

* Added OBJECTID as a compound STATUS attribute and FETCH data item that returns all relevant identifiers for the context.

* Added the ENABLE requirement to ensure backward compatibility with clients that do not understand the extended response syntax.

* Defined coexistence rules for servers that advertise both the OBJECTID capability from [RFC8474] and the OBJECTIDBIS capability from this document.

* Revised the formal syntax to introduce the objectid-atom, objectid, and objectid-or-nil productions, providing consistent parenthesization and NIL handling across all response types.

* Corrected the THREADID response syntax from [RFC8474], which inconsistently applied parenthesization between the objectid and NIL alternatives.

* Updated IANA registrations to include the OBJECTIDBIS capability and OBJECTID response code.

* Added security considerations for account identifier exposure, cross-account information leakage, JMAP authentication consistency, and privacy in multi-tenant environments.

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-gondwana-degennaro-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