Stream: Internet Engineering Task Force (IETF)
RFC: 9979
Category: Informational
Published: May 2026
ISSN: 2070-1721
Authors: N. Jenkins, *Fastmail*; D. Eggert, *Apple, Inc*

RFC 9979
Registration of Further IMAP/JMAP Keywords and Mailbox Name Attributes

Abstract

This document defines a number of keywords and mailbox name attributes that have been in use across different server and client implementations. It defines the intended use of these keywords and mailbox name attributes. This document registers all of these with IANA to avoid name collisions.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

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). Not all documents approved by the IESG are candidates for any level of Internet Standard; see 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/rfc9979.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

The Internet Message Access Protocol (IMAP) specification [RFC9051] defines the use of message keywords, and [RFC5788] describes the creation of the "IMAP and JMAP Keywords" registry. Similarly, [RFC8457] describes the creation of the "IMAP Mailbox Name Attributes" registry. [RFC8621] updates these registries to apply to messages and mailboxes over the JMAP as well.

This document defines 17 message keywords:

- `$autosent`
- `$canunsubscribe`
- `$followed`
- `$hasattachment`
- `$hasmemo`
- `$hasnoattachment`
- `$imported`
- `$istrusted`
- `$MailFlagBit0`
- `$MailFlagBit1`
- `$MailFlagBit2`
- `$maskedemail`
- `$memo`
- `$muted`
- `$new`
- `$notify`
- `$unsubscribed`

And defines three mailbox name attributes:

- `Memos`
- `Scheduled`
- `Snoozed`

This document also registers these in the "IMAP and JMAP Keywords" registry and "IMAP Mailbox Name Attributes" registry.

2. Requirements Language

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. Flag Color Keywords

The Internet Message Access Protocol (IMAP) specification [RFC9051] defines a `\Flagged` system flag to mark a message as urgent / in need of special attention. The new keywords defined in Section 3.1 allow such a flagged message to have that flag be of one of seven colors.

3.1. Definition of the `$MailFlagBit_` Message Keywords

The three flag color keywords:

- `$MailFlagBit0`
- `$MailFlagBit1`
- `$MailFlagBit2`

make up a bit pattern that defines the color of the flag as such:

Bit 0	Bit 1	Bit 2	Color
0	0	0	red
1	0	0	orange
0	1	0	yellow
1	1	0	green
0	0	1	blue
1	0	1	purple
0	1	1	gray

Table 1: Flag Colors

Note that the bit combination 111 (all three bits set) is not assigned a color, resulting in seven distinct flag colors instead of eight possible combinations.

These flags **MUST** be ignored if the `\Flagged` system flag is not set. If the `\Flagged` system flag is set, the flagged status **MAY** be presented to the user in the color corresponding to the combination of the three flag color keywords.

3.2. Implementation Notes

A mail client that is aware of these flag color keywords **MUST** clear all three flag color keywords when the user unflags the message, i.e., when clearing the `\Flagged` system flag, all three flag color keywords **MUST** also be cleared.

A mail client **MUST NOT** set any of these flags unless the `\Flagged` system flag is already set or is being set.

Servers **MAY** clear these flag color keywords when a client clears the `\Flagged` system flag.

While these keywords are defined in terms of colors, clients **SHOULD** provide alternatives for users who cannot perceive colors. This could include using different shapes, patterns, text labels, audio cues, or other distinguishing characteristics that convey the same semantic meaning as the color variations. The specific bit patterns can serve as identifiers for such alternative representations.

4. Attachment Detection Keywords

The following keywords help clients determine whether a message contains attachments without having to fetch and parse the entire message structure. This is particularly useful for displaying attachment indicators in message lists or implementing attachment-based filtering.

The `$hasattachment` and `$hasnoattachment` keywords are mutually exclusive. A message **MUST NOT** have both keywords set simultaneously.

4.1. `$hasattachment`

The `$hasattachment` keyword indicates that a message has one or more attachments. It is set by the server during message delivery; otherwise, it is set by the client (if neither `$hasattachment` nor `$hasnoattachment` is currently set).

This keyword enables clients to indicate attachments (e.g., paperclip icons) in message lists without having to fetch the full MIME structure for each message. It also facilitates attachment-based filtering and search operations.

When using JMAP, the "hasAttachment" Email property **MUST** reflect the same information as this keyword for consistency across protocols.

4.2. `$hasnoattachment`

The `$hasnoattachment` keyword explicitly indicates that a message does not have any attachments. It is set by the server during message delivery; otherwise, it is set by the client (if neither `$hasattachment` nor `$hasnoattachment` is currently set).

This keyword complements `$hasattachment` by providing definitive information about messages that have been analyzed but found to contain no attachments. The absence of `$hasattachment` alone is inconclusive, as it might simply mean the message hasn't been processed for attachment detection.

This explicit distinction is important for clients that need to know whether attachment detection has been performed on a message. When using JMAP, the "hasAttachment" Email property **MUST** reflect the same information as this keyword.

5. Memos Keywords

The following keywords are used to support user-created annotations or memos attached to messages. They allow for contextual notes to be added to conversations while maintaining proper cross-referencing between messages and their annotations.

The `$memo` keyword is applied to the memo message itself (the note-to-self), while the `$hasmemo` keyword is applied to the original message being annotated. This creates a bidirectional relationship that allows clients to efficiently locate memos and their associated messages.

The `$memo` and `$hasmemo` keywords are mutually exclusive. A message **MUST NOT** have both keywords set simultaneously.

5.1. $hasmemo

The `$hasmemo` keyword indicates that a message has an associated memo (identified by the `$memo` keyword) in the same thread. This allows clients to efficiently identify messages with annotations without having to examine all messages in a thread.

When a client creates a memo for a message, it **MUST** add this keyword to the message being annotated while adding the `$memo` keyword to the memo message itself. Conversely, when a memo is deleted, the client **MUST** remove this keyword from the annotated message.

This keyword enables several optimizations for clients. It allows for efficient searching for messages with annotations, enables indicators in message lists to indicate which messages have memos, and helps clients decide whether to fetch entire conversation threads when loading a mailbox to ensure memos are properly presented.

5.2. $memo

The `$memo` keyword identifies a message as a note-to-self created by the user regarding another message in the same thread. It allows for contextual annotations to be added to conversations.

Messages with this keyword should be constructed similarly to a reply to the message being annotated, with appropriate Subject and Reply-To headers set to maintain context and threading. Servers **SHOULD** store messages with this keyword in a mailbox with the "Memos" mailbox name attribute (see Section 9.2.3), if available.

Supporting clients **MUST** present these messages differently from regular emails. Rather than presenting them as standalone messages, they **MUST** be presented as annotations attached to the message they're commenting on. Clients may provide special UI affordances for editing or deleting these memos, which typically requires replacing the message since email messages are immutable.

When a client creates or removes a memo, it **MUST** also set or clear the related `$hasmemo` keyword on the message being annotated to maintain proper cross-referencing.

6. Subscription Management Keywords

The following keywords help clients manage mailing list subscriptions. They provide mechanisms for tracking unsubscription attempts and identifying messages with valid unsubscribe options.

6.1. `$canunsubscribe`

The `$canunsubscribe` keyword indicates that a message contains a valid, List-Unsubscribe header compliant with [RFC8058] that clients can use to provide one-click unsubscribe functionality.

This keyword is set by servers during message delivery when they detect a valid List-Unsubscribe header and the message passes implementation-specific reputation checks. This pre-verification is important, as not all List-Unsubscribe headers are trustworthy: some might lead to phishing sites or generate additional spam.

The primary benefit of this keyword is efficiency: clients can know whether to offer unsubscribe functionality in their user interface without having to fetch and validate the List-Unsubscribe header for every message. It also provides an extra layer of safety since the server has already performed reputation checks on the unsubscribe mechanism.

6.2. `$unsubscribed`

The `$unsubscribed` keyword indicates that the user has attempted to unsubscribe from the mailing list associated with a message. It provides a persistent record of unsubscription attempts across multiple clients.

This keyword is set by a client after a user attempts to unsubscribe from a mailing list, typically via a one-click List-Unsubscribe action as defined in [RFC8058]. It serves as a record that an unsubscription attempt has been made, even if confirmation of successful unsubscription hasn't been received. It **MUST NOT** be set if the unsubscription attempt definitely failed.

Supporting clients can use this to provide an indicator on messages with this keyword to remind the user they have previously attempted to unsubscribe from this sender or mailing list. This can be helpful when users revisit old messages and might otherwise attempt to unsubscribe again or when they receive additional messages despite unsubscribing and need to take further action.

7. Other Message Keywords

7.1. `$autosent`

The `$autosent` keyword identifies messages that were automatically generated and sent by the system on behalf of the user, typically in response to user-defined filtering rules or settings.

This keyword is set by the server on the user's copy of vacation auto-replies, automated responses, or other system-generated messages sent on behalf of the user. It allows clients to clearly distinguish these messages from those directly composed and sent by the user.

Supporting clients can present these messages differently in the \Sent mailbox, such as with a distinct indicator (e.g., icon or label) indicating their automated nature. This helps prevent user confusion when they encounter messages they don't remember writing, particularly in the case of vacation responses or other automated replies that may have been configured and then forgotten.

7.2. `$followed`

The `$followed` keyword indicates that a user is particularly interested in future messages in a specific thread. This enables special handling to ensure these messages receive appropriate attention.

When a new message arrives in a thread where this keyword is present, supporting servers **MAY** take actions to prioritize the message. This could include ensuring it is available in the inbox regardless of filtering rules that might otherwise affect it, automatically adding the `$notify` keyword to ensure notifications, or applying other handling that increases its prominence. The specific actions taken and whether they can be configured by the user depend on the implementation.

Thread identification for this keyword follows the same definition as described in Section 7.6.

This keyword is set or cleared by clients based on user actions to follow or unfollow a thread. When unfollowing a thread, clients **MUST** remove this keyword from all messages in the thread that have it. The `$followed` keyword is mutually exclusive with the `$muted` keyword. If both are present on messages in the same thread, servers **MUST** treat the thread as followed rather than muted.

7.3. `$imported`

The `$imported` keyword identifies messages that were imported into the mailbox from another system rather than received through normal mail delivery channels.

This keyword is set by servers during import operations from other mail systems, data migrations, or manual imports from local files. It helps distinguish messages that didn't arrive through normal mail delivery and may have different characteristics or behaviors.

Supporting clients can provide an indicator for imported messages, especially if they have unusual properties such as unexpectedly old dates, unusual header structures, or other anomalies that might otherwise confuse users. Some clients might also offer filtering or search capabilities specifically for imported messages.

7.4. `$istrusted`

The `$istrusted` keyword indicates that a message's sender identity has been verified by the server with a high degree of confidence. It provides an indication that the message is likely authentic.

This advisory keyword is set by the server during message delivery when the authenticity of both the sender's name and email address can be verified with a high degree of confidence. This level of verification typically applies to messages sent by the mailbox provider itself to its customers, where the provider has strong evidence of the message's origin.

Supporting clients can provide a verification indicator (e.g., check mark icon) for messages with this keyword, helping users identify legitimate communications from their service provider. This is particularly important for distinguishing authentic provider messages from phishing attempts that impersonate the provider.

Servers **MUST** exercise caution when applying this keyword, as it conveys trust to the user. If misapplied, it could lead users to trust fraudulent messages.

It **MUST NOT** be used for messages that have only passed standard authentication mechanisms like Sender Policy Framework (SPF), DomainKeys Identified Mail (DKIM), or Domain-based Message Authentication, Reporting, and Conformance (DMARC).

7.5. `$maskedemail`

The `$maskedemail` keyword indicates that a message was received through a masked email address -- an alias created specifically for communications with a particular sender to protect the user's primary email address.

This advisory keyword is set by the server during message delivery on messages that arrive via such aliases. These might be automatically generated single-use addresses, service-specific addresses, or user-created aliases for specific correspondents.

Supporting clients can provide an indicator (e.g., mask icon) for messages with this keyword. This helps users understand that the message was received through a privacy-enhancing alias rather than their primary address. Clients might also provide related functionality, such as the ability to disable the specific alias used if it starts receiving unwanted messages.

7.6. `$muted`

The `$muted` keyword provides a way for users to indicate they are not interested in future messages in a particular conversation thread. This enables automated processing of subsequent messages in that thread to reduce their prominence.

When a new message arrives that belongs to the same thread as one marked with this keyword, supporting servers **MAY** automatically deprioritize the message. This could include moving it to an archive or trash folder, marking it as read, or applying other handling that reduces its prominence. The specific actions taken and whether they can be configured by the user depends on the implementation.

For thread identification purposes, messages are considered part of the same thread when they share the same thread identifier as defined in Section 5.2 of [RFC8474] for IMAP or Section 3 of [RFC8621] for JMAP. This definition applies to all thread-related keywords in this document.

This keyword is set or cleared by clients based on user actions to mute or unmute a thread. When unmuting a thread, clients **MUST** remove this keyword from all messages in the thread that have it. The `$muted` keyword is mutually exclusive with the `$followed` keyword. If both are present on messages in the same thread, both servers and clients **MUST** treat the thread as followed rather than muted.

Implementers should note the security considerations in the IANA registration: if an attacker gains access to a user's account, they could mute threads to hide important communications. However, this represents only a small increase in risk since compromised accounts allow many other similar actions.

7.7. $new

The `$new` keyword indicates that a message should be emphasized or made more prominent to the user due to a recent system action, even if the message itself is not new.

This advisory keyword is typically set by servers when a previously snoozed message "awakens" and returns to the inbox after its snooze period has elapsed. It signals to clients that, although this message might have an older date, it should be treated as effectively new in terms of user attention.

Supporting clients can present these messages with special treatment to draw user attention, such as emphasizing, bolding, or placing them at the top of the message list, even if strict date sorting would place them elsewhere.

Clients **MUST** clear this keyword when the user interacts with the message, such as by opening, replying to, or otherwise acknowledging it. This prevents the message from remaining emphasized indefinitely after the user has accessed it.

7.8. $notify

The `$notify` keyword indicates that a client should present a notification for a message. This provides a mechanism for more selective notifications than the common approach of notifying for every message arriving in the inbox.

When a message with this keyword is delivered to a mailstore, supporting clients **SHOULD** present a notification to the user unless configured otherwise by the user. This notification may occur alongside other message notifications according to user preferences.

This keyword enables both users (through filtering rules) and servers (through automated processing) to identify specific messages that warrant user attention, even if they might otherwise be filtered or sorted in a way that wouldn't normally trigger a notification.

Servers typically set this keyword during message delivery when a message meets certain criteria indicating importance to the user. Clients may clear the keyword when the user interacts with the message by opening it, archiving it, or performing other actions. When one client clears this keyword, other clients connected to the same account may choose to automatically dismiss their corresponding notification.

8. Mailbox Name Attributes

8.1. `Snoozed`

The `Snoozed` mailbox name attribute identifies a mailbox that is used to store messages that have been temporarily removed from the user's attention and scheduled to reappear at a later time.

When a user chooses to "snooze" a message, the supporting server moves the message to a mailbox with this attribute. At the specified "awaken" time, the server automatically moves the message back to its original location (typically the inbox) and may mark it with the `$new` keyword to ensure it receives proper attention.

This attribute standardizes the location of snoozed messages across clients, allowing them to identify and manage snoozed messages consistently. However, this attribute merely identifies the storage location for snoozed messages and does not itself define the snoozing mechanism or interface.

Supporting clients may present the contents of this mailbox differently from regular mailboxes, such as organizing messages by their scheduled "awaken" time rather than received date or providing specialized interfaces for adjusting the snooze duration.

8.2. `Scheduled`

The `Scheduled` mailbox name attribute identifies a mailbox that contains messages that have been composed but are scheduled to be sent at a future time rather than immediately.

When a user composes a message and chooses to send it at a later date or time, the supporting server stores the message in a mailbox with this attribute until the scheduled sending time. Once that time is reached, the server sends the message and typically moves it to the \Sent mailbox or delete it according to server policy.

This attribute standardizes the location of scheduled messages across clients, enabling users to review, edit, or cancel scheduled messages before they are sent, regardless of which client was used to schedule them.

Supporting clients may present the contents of this mailbox in a way that highlights the scheduled sending time and allows users to modify or cancel scheduled messages before they are sent.

8.3. `Memos`

The `Memos` mailbox name attribute identifies a mailbox designated for storing messages that have the `$memo` keyword. These messages are user-created notes or annotations related to other messages.

When a user creates a memo (a note-to-self regarding another message), clients **SHOULD** store these messages in a mailbox with this attribute. This separation allows clients to handle memos differently from regular messages, presenting them as annotations rather than standalone communications.

Storing memos in a designated mailbox helps prevent them from cluttering the user's inbox or other message folders, while still maintaining them as proper email messages for compatibility and synchronization purposes.

Supporting clients **MUST NOT** present the contents of this mailbox as regular messages in their interface; instead, they **MUST** present them contextually alongside the messages they annotate when those messages are accessed.

9. IANA Considerations

The following 17 IMAP/JMAP keywords have been registered in the "IMAP and JMAP Keywords" registry, as established in [RFC5788].

9.1. IMAP/JMAP Keyword Registrations

9.1.1. `$autosent` Keyword Registration

IMAP/JMAP keyword name: `$autosent`

Purpose: Indicate to the client that a message was sent automatically as a response due to a user rule or setting.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: This keyword is set by the server on delivery.

Related keywords: None

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.2. `$canunsubscribe` Keyword Registration

IMAP/JMAP keyword name: `$canunsubscribe`

Purpose: Indicate to the client that this message has a List-Unsubscribe header that is compliant with [RFC8058].

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: This keyword is set by the server on delivery.

Related keywords: None

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.3. `$followed` Keyword Registration

IMAP/JMAP keyword name: `$followed`

Purpose: Indicate to the server that the user is particularly interested in future replies to a particular thread.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword can cause automatic action.

When/by whom the keyword is set/cleared: This keyword is set and cleared by the client.

Related keywords: Mutually exclusive with `$muted`. If both are specified on a thread, servers **MUST** behave as though only `$followed` were set.

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.4. `$hasattachment` Keyword Registration

IMAP/JMAP keyword name: `$hasattachment`

Purpose: Indicate to the client that a message has an attachment.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: It is set by the server during message delivery; otherwise, it is set by the client (if neither `$hasattachment` nor `$hasnoattachment` is currently set).

Related keywords: Mutually exclusive with `$hasnoattachment`

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.5. `$hasmemo` Keyword Registration

IMAP/JMAP keyword name: `$hasmemo`

Purpose: Indicate to the client that a message has an associated memo with the `$memo` keyword.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: This keyword is set and cleared by the client based on user interaction.

Related keywords: The `$memo` and `$hasmemo` keywords are mutually exclusive.

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.6. `$hasnoattachment` Keyword Registration

IMAP/JMAP keyword name: `$hasnoattachment`

Purpose: Indicate to the client that a message does not have an attachment.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: It is set by the server during message delivery; otherwise, it is set by the client (if neither `$hasattachment` nor `$hasnoattachment` is currently set).

Related keywords: Mutually exclusive with `$hasattachment`

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.7. `$imported` Keyword Registration

IMAP/JMAP keyword name: `$imported`

Purpose: Indicate to the client that this message was imported from another mailbox.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: This keyword is set by the server on delivery.

Related keywords: None

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.8. `$istrusted` Keyword Registration

IMAP/JMAP keyword name: `$istrusted`

Purpose: Indicate to the client that the authenticity of the from name and email address have been verified by the server.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: This keyword is set by the server on delivery.

Related keywords: None

Related IMAP capabilities: None

Security considerations: Servers should make sure this keyword is only set for messages that really are trusted.

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.9. `$MailFlagBit0` Keyword Registration

IMAP/JMAP keyword name: `$MailFlagBit0`

Purpose: Bit 0 part of a 3-bit bitmask that defines the color of the flag when the message has the system flag `\Flagged` set. See Section 3 for details.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: This keyword is set by a client as the result of a user action to "flag" a message as urgent / in need of special attention.

Related keywords: `$MailFlagBit1`, `$MailFlagBit2`

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.10. `$MailFlagBit1` Keyword Registration

IMAP/JMAP keyword name: `$MailFlagBit1`

Purpose: Bit 1 part of a 3-bit bitmask that defines the color of the flag when the message has the system flag `\Flagged` set. See Section 3 for details.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: This keyword is set by a client as the result of a user action to "flag" a message as urgent / in need of special attention.

Related keywords: `$MailFlagBit0`, `$MailFlagBit2`

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.11. `$MailFlagBit2` Keyword Registration

IMAP/JMAP keyword name: `$MailFlagBit2`

Purpose: Bit 2 part of a 3-bit bitmask that defines the color of the flag when the message has the system flag `\Flagged` set. See Section 3 for details.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: This keyword is set by a client as the result of a user action to "flag" a message as urgent / in need of special attention.

Related keywords: `$MailFlagBit0`, `$MailFlagBit1`

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.12. `$maskedemail` Keyword Registration

IMAP/JMAP keyword name: `$maskedemail`

Purpose: Indicate to the client that the message was received via an alias created for an individual sender.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: This keyword is set by the server on delivery.

Related keywords: None

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: LIMITED USE

Scope: BOTH

Owner/Change controller: IESG

9.1.13. `$memo` Keyword Registration

IMAP/JMAP keyword name: `$memo`

Purpose: Indicate to the client that a message is a note-to-self from the user regarding another message in the same thread.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: This keyword is set and cleared by the client based on user interaction.

Related keywords: The `$memo` and `$hasmemo` keywords are mutually exclusive.

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.14. `$muted` Keyword Registration

IMAP/JMAP keyword name: `$muted`

Purpose: Indicate to the server that the user is not interested in future replies to a particular thread.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword can cause an automatic action.

When/by whom the keyword is set/cleared: This keyword is set and cleared by the client.

Related keywords: Mutually exclusive with `$followed`. If both are specified on a thread, servers **MUST** behave as though only `$followed` were set.

Related IMAP capabilities: None

Security considerations: Muting a thread can mean a user won't be notified of a reply. If someone compromises a user's account, they may mute threads where they don't want the user to be notified of the reply, for example, when sending phishing to the user's contacts. However, there are many other ways an attacker with access to the user's mailbox can also achieve this, so this is not greatly increasing the attack surface. When restoring legitimate access to a stolen account, care must be taken to find and confirm or revert mute actions that may have been taken by the attacker.

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.15. `$new` Keyword Registration

IMAP/JMAP keyword name: `$new`

Purpose: Indicate to the client that a message should be made more prominent to the user due to a recent action.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: This keyword is set by the server. Clients can clear the keyword based on user interaction.

Related keywords: None

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: LIMITED USE

Scope: BOTH

Owner/Change controller: IESG

9.1.16. `$notify` Keyword Registration

IMAP/JMAP keyword name: `$notify`

Purpose: Indicate to the client that a notification should be presented for this message.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword can cause an automatic action.

When/by whom the keyword is set/cleared: This keyword is set by the server on delivery when a message meets certain criteria. It may be cleared by a client when the user interacts with the message.

Related keywords: None

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.1.17. `$unsubscribed` Keyword Registration

IMAP/JMAP keyword name: `$unsubscribed`

Purpose: Indicate the client has attempted to unsubscribe from the mailing list this message is from.

Private or Shared on a server: SHARED

Is it an advisory keyword or may it cause an automatic action: This keyword is advisory.

When/by whom the keyword is set/cleared: This keyword is set by the client based on user interaction.

Related keywords: None

Related IMAP capabilities: None

Security considerations: None

Published specification: RFC 9979

Intended usage: COMMON

Scope: BOTH

Owner/Change controller: IESG

9.2. IMAP Mailbox Name Attributes Registrations

The following three IMAP/JMAP mailbox name attributes have been registered in the "IMAP Mailbox Name Attributes" registry, as established by [RFC8457].

Note that none of the attributes in this section have an implied backslash. This sets them apart from those specified in Section 2 of [RFC6154].

9.2.1. `Snoozed` Mailbox Name Attribute Registration

Attribute Name: Snoozed

Description: Identifies the mailbox where temporarily snoozed messages are stored.

Reference: RFC 9979

9.2.2. `Scheduled` Mailbox Name Attribute Registration

Attribute Name: Scheduled

Description: Identifies the mailbox where messages scheduled to be sent at a later time are stored.

Reference: RFC 9979

9.2.3. `Memos` Mailbox Name Attribute Registration

Attribute Name: Memos

Description: Identifies the mailbox where user-created memo messages are stored.

Reference: RFC 9979

10. Security Considerations

The security considerations for the `$istrusted` and `$muted` keywords are described in Sections 9.1.8, 7.4, and 9.1.14, respectively.

The use and interpretation of the keywords and mailbox name attributes defined in this document depend on the client's and user's ability to trust the IMAP server. Clients should be aware that a compromised or malicious server could set these keywords incorrectly or manipulate them to mislead users. For additional security considerations related to IMAP, refer to [RFC9051].

Otherwise, this document should not affect the security of the Internet.

11. 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>.

[RFC5788] Melnikov, A. and D. Cridland, "IMAP4 Keyword Registry", RFC 5788, DOI 10.17487/RFC5788, March 2010, <https://www.rfc-editor.org/info/rfc5788>.

[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/info/rfc6154>.

[RFC8058] Levine, J. and T. Herkula, "Signaling One-Click Functionality for List Email Headers", RFC 8058, DOI 10.17487/RFC8058, January 2017, <https://www.rfc-editor.org/info/rfc8058>.

[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>.

[RFC8457] Leiba, B., Ed., "IMAP "$Important" Keyword and "\Important" Special-Use Attribute", RFC 8457, DOI 10.17487/RFC8457, September 2018, <https://www.rfc-editor.org/info/rfc8457>.

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

[RFC8621] Jenkins, N. and C. Newman, "The JSON Meta Application Protocol (JMAP) for Mail", RFC 8621, DOI 10.17487/RFC8621, August 2019, <https://www.rfc-editor.org/info/rfc8621>.

[RFC9051] Melnikov, A., Ed. and B. Leiba, Ed., "Internet Message Access Protocol (IMAP) - Version 4rev2", RFC 9051, DOI 10.17487/RFC9051, August 2021, <https://www.rfc-editor.org/info/rfc9051>.

Authors' Addresses

Neil Jenkins
Fastmail
PO Box 234, Collins St West
Melbourne VIC 8007
Australia
Email: neilj@fastmailteam.com
URI: https://www.fastmail.com

Daniel Eggert
Apple, Inc
One Apple Park Way
Cupertino, CA 95014
United States of America
Email: deggert@apple.com
URI: https://www.apple.com