Mail & Messages

- Mail & Messages

Mail & Messages

Mail & Messages

Complete mail operations: fetching messages (metadata + full bodies), labeling, marking read/unread/forwarded, deleting, creating/updating/sending drafts, importing messages, building RFC822 MIME messages, and encrypting/decrypting with PGP.

Fetching Messages

Single Message

```
msg, err := c.GetMessage(ctx, messageID)
// msg contains: ID, addressID, labelIDs, subject, sender/recipients, flags, timestamps
```

Full Message with Attachments

```
fullMsg, err := c.GetFullMessage(ctx, messageID, scheduler, storageProvider)
// fullMsg.Message + fullMsg.AttData ([][]byte of decoded attachment data)
```

Message Metadata (List)

```
// Get all message metadata
meta, err := c.GetMessageMetadata(ctx, proton.MessageFilter{
    LabelIDs: []string{"inbox-label-id"},
})

// Paginated with filters
meta, err := c.GetMessageMetadataPage(ctx, page, pageSize, filter)
```

```
// Count messages
count, err := c.CountMessages(ctx)

// Get all message IDs (auto-paginating)
allIDs, err := c.GetAllMessageIDs(ctx, afterID)
```

Drafts

```
// Create a draft
draft, err := c.CreateDraft(ctx, addrKR, proton.CreateDraftReq{
    Subject: "Hello",
    Sender:  proton.Address{ID: "address-id"},
    To:      []proton.Address{{Email: "recipient@example.com"}},
    Body:    "Message body",
    MIMEType: "text/plain",
})

// Update a draft
updated, err := c.UpdateDraft(ctx, draftID, addrKR, proton.UpdateDraftReq{
    Body: "Updated body",
    Action: proton.DraftActionReply, // reply/reply-all/forward/auto-response/read-receipt
})

// Send a draft
err := c.SendDraft(ctx, draftID, proton.SendDraftReq{
    // MessagePackage with encrypted body + per-recipient key packets
})
```

Bulk Operations

```
// Delete messages (parallel chunking, max 1000 per page)
err := c.DeleteMessage(ctx, id1, id2, id3...)

// Mark as read/unread
err := c.MarkMessagesRead(ctx, ids...)
```

```
err := c.MarkMessagesUnread(ctx, ids...)

// Mark as forwarded/unforwarded
err := c.MarkMessagesForwarded(ctx, ids...)
err := c.MarkMessagesUnForwarded(ctx, ids...)

// Label/unlabel with undo support
err := c.LabelMessages(ctx, ids, labelID)
err := c.UnlabelMessages(ctx, ids, labelID)
// Returns undo tokens for reversible actions
```

Message Import

Bulk import of raw RFC822 messages:

```
stream := c.ImportMessages(ctx, addrKR, workers, buffer, req1, req2, ...)
// req := proton.ImportReq{
//     RawMessage: []byte("Raw RFC822 message..."),
//     AddressID:  "address-id",
//     LabelIDs:   []string{"label-id"},
// }

for res := range stream {
    fmt.Println(res.ID, res.Error)
}
```

MIME Building & Encryption

```
// Reconstruct full RFC822 MIME from decrypted parts
rawMIME, err := proton.BuildRFC822(kr, msg, attData)

// Encrypt a raw RFC822 message
encrypted, err := proton.EncryptRFC822(kr, literal)
```

Key Types

Type	Description
`MessageMetadata`	ID, addressID, labelIDs, subject, sender/recipients, flags, timestamps
`Message`	Extends Metadata with raw Header, Headers, encrypted Body, MIMEType, Attachments
`FullMessage`	Message + decoded attachment data
`MessageFilter`	Filter by ID list, subject, addressID, externalID, labelID
`MessageFlag`	Bitmask: received, sent, internal, E2E, replied, forwarded, spam/DMARC/DKIM/SPF
`EncryptionScheme`	InternalScheme, EncryptedOutsideScheme, ClearScheme, PGPInlineScheme, PGPMIMEScheme
`SendPreferences`	Per-recipient encryption/signature/MIME type preferences
`MessagePackage`	Encrypted message body + per-recipient key packets
`DraftTemplate`	Subject, sender, recipients, body, MIME type for draft creation
`ImportReq` / `ImportMetadata`	Import request with raw message bytes and metadata