Overview & Quick Start

Overview & Quick Start

Go Proton API

Overview

`github.com/ProtonMail/go-proton-api` is the official Go library implementing a client for (a subset of) the Proton REST API. It covers **Mail, Contacts, Calendar, Drive Shares**, and core account operations. Built on `resty/v2` for HTTP communication and `gopenpgp/v2` for PGP encryption/decryption.

Quick Start

```
import (
    "context"
    "github.com/ProtonMail/go-proton-api"
)

// 1. Create a Manager (shared across clients)
m := proton.New()
defer m.Close()

ctx := context.Background()

// 2a. Login with username/password (full SRP flow)
c, auth, err := m.NewClientWithLogin(ctx, "user@proton.me", []byte("password"))
if err != nil {
    panic(err)
}
defer c.Close()

// Handle 2FA if needed
if auth.TwoFA.Enabled&proton.HasTOTP != 0 {
    if err := c.Auth2FA(ctx, proton.Auth2FAReq{TwoFactorCode: "123456"}); err != nil {
```

```
        panic(err)
    }
}

// 2b. Or create from existing tokens (no login needed)
c := m.NewClient("uid", "accessToken", "refreshToken")
defer c.Close()
```

Installation

```
go get github.com/ProtonMail/go-proton-api
```

Requires **Go 1.26+**.

Architecture

The library uses a **Manager → Client** hierarchy:

- **Manager** — Top-level factory object. Created via `New(opts ...Option)`. Holds a shared `*resty.Client`, status observers, error handlers, and panic recovery. Shared across all clients.
- **Client** — Per-user client bound to a specific UID. Created by the Manager via `NewClient()`, `NewClientWithLogin()`, or `NewClientWithRefresh()`. Each Client handles one Proton account.

Core Features

Feature	Description
Automatic auth refresh	401 responses trigger transparent token refresh + retry
Connection status	`Manager.AddStatusObserver()` receives `StatusUp` / `StatusDown` notifications
Error handlers	`Manager.AddErrorHandler(code, handler)` — register callbacks for specific API error codes
Request hooks	Per-client and global pre/post request middleware via resty
Parallel operations	Delete messages, import messages, and attachment downloads all use parallel execution
Undo support	Label/unlabel operations return undo tokens for reversible actions
Paging helpers	`GetAllContacts()`, `GetAllMessageIDs()` etc. auto-page through results