Authentication & Sessions

Authentication & Sessions

Authentication & Sessions

The library handles the full authentication lifecycle: SRP-based login, 2FA/FIDO2, token refresh, session management, and user creation.

Login Methods

SRP Login (Username/Password)

Full Secure Remote Password login flow -- zero-knowledge password authentication:

```
m := proton.New()
ctx := context.Background()

c, auth, err := m.NewClientWithLogin(ctx, "user@proton.me", []byte("password"))
if err != nil {
    panic(err)
}
defer c.Close()
```

2FA Support

If the account has 2FA enabled:

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

FIDO2 Support

```
if auth.TwoFA.Enabled&proton.HasFIDO2 != 0 {
    if err := c.Auth2FA(ctx, proton.Auth2FAReq{
        FIDO2Data: proton.FIDO2Req{
            Attestation: "...",
            AuthenticatorData: "...",
            Signature: "...",
            CredentialID: "...",
        },
    }); err != nil {
        panic(err)
    }
}
```

Refresh Token Login

Create a client from existing refresh token (no password needed):

```
c, _, err := m.NewClientWithRefresh(ctx, "uid", "refreshToken")
if err != nil {
    panic(err)
}
defer c.Close()
```

Direct Token Login

Create a client when all auth info is already known:

```
c := m.NewClient("uid", "accessToken", "refreshToken")
defer c.Close()
```

Session Management

```
// List active sessions
sessions, err := c.AuthSessions(ctx)

// Revoke a specific session
err := c.AuthRevoke(ctx, authUID)
```

```
// Revoke all sessions
err := c.AuthRevokeAll(ctx)

// Delete current session
err := c.AuthDelete(ctx)
```

Auth Handlers

Register callbacks to handle auth events:

```
// Save new tokens when they are refreshed
c.AddAuthHandler(func(auth proton.Auth) {
    // Save auth.AccessToken, auth.RefreshToken to keychain
})

// Handle de-authentication
c.AddDeauthHandler(func() {
    // Clear stored credentials
})
```

Key Types

Type	Description
Auth	Contains UID, AccessToken, RefreshToken, ServerProof, Scope, TwoFA info
AuthInfo	Server-provided auth metadata: salt, modulus, server ephemeral, 2FA status
TwoFAInfo / TwoFAStatus	Enum for TOTP, FIDO2, or both
FIDO2Req	FIDO2 authentication data (attestation, authenticator data, signature, credential ID)
AuthSession	Active session with client info and revocability flag
PasswordMode	One-password vs two-password mode

Security Notes

- Login uses SRP (Secure Remote Password) protocol via go-srp for zero-knowledge password authentication
- By default, the Manager verifies the server expected proof (verifyProofs flag) to detect MITM attacks
- The server sends salt/modulus/ephemeral; the client computes proofs and verifies the server proof