Real-time Events & User Account

Real-time Events

Event polling and streaming for real-time notifications via `/core/v4/events`.

Event Streaming

```
// Get the latest event ID
fromEventID, err := c.GetLatestEventID(context.Background())

// Create a new event streamer
for event := range c.NewEventStream(ctx, 20*time.Second, 20*time.Second, fromEventID) {
    fmt.Println(event.EventID)
    // Process event.User, event.Messages, event.Labels, etc.
}
```

The event stream uses a custom `NewTicker` with random jitter to avoid thundering herd.

Event Polling

```
// Fetch events starting after eventID
events, more, err := c.GetEvent(ctx, eventID)
// Returns up to 50 events per call with more indicating continuation
```

Event Types

The `Event` struct contains typed sub-events:

```
event.User          // User changes
event.UserSettings  // User settings changes
event.MailSettings  // Mail settings changes
event.Messages      // Message changes ([]MessageEvent)
event.Labels        // Label changes ([]LabelEvent)
event.Addresses     // Address changes ([]AddressEvent)
event.Notifications // Notifications ([]NotificationEvent)
event.UsedSpace     // Storage usage changes
```

Key Types

Type	Description
`Event`	Contains eventID, refresh flags, and typed sub-events
`RefreshFlag`	RefreshMail (1), RefreshAll (255)
`EventAction`	Delete, Create, Update, UpdateFlags
`MessageEvent`	EventItem (ID + Action) with full Message entity
`LabelEvent`	EventItem (ID + Action) with full Label entity
`AddressEvent`	EventItem (ID + Action) with full Address entity

User Account

User account operations via `/core/v4/users`.

Operations

```
// Fetch current user info
user, err := c.GetUser(ctx)

// With human verification token
user, err := c.GetUserWithHV(ctx, hvToken)

// Delete account (requires SRP proof re-authentication)
err := c.DeleteUser(ctx, password, proton.DeleteUserReq{
```

```
    Reason:  "no longer needed",
    Feedback: "test account",
})
```

Key Types

Type	Description
`User`	ID, name, display name, email, keys, used/max/upload space limits, credit, currency
`ProductUsedSpace`	Space breakdown by calendar, contact, drive, mail, pass (password manager)
`DeleteUserReq`	Reason, feedback, email confirmation

Feature Flags

Fetch feature flag/toggle results from the Proton API:

```
features, err := c.GetFeatures(ctx, "sticky-key-uuid")
// Returns FeatureFlagResult for a given UUID sticky key
```

Revision #1
Created 7 May 2026 11:44:36 by Clive
Updated 7 May 2026 11:44:37 by Clive