Services

Drop External Services

> Source: `src/drop-app/src/lib/services/`

Overview

Drop uses a **PSD2 pass-through model** — it never holds customer money. AISP reads bank balances via Open Banking, PISP initiates payments from the user's own bank account.

Drop integrates with external service providers. Each service has a different readiness level — see status tags below.

For backend FX-rate sourcing, see Currency Rates. Current `drop-api` source is configured for Norges Bank primary rates with ECB fallback; `/api/rates` reads the persisted `exchange_rates` table rather than calling providers directly.

> **Legend:** `[PRODUCTION]` = real SDK, production-ready. `[MOCK/DEV]` = mock only, NOT connected to real APIs. `[PLANNED]` = future roadmap. `[DEPRECATED]` = no longer the chosen provider.

Service mode is controlled by `NEXT_PUBLIC_SERVICE_MODE` env var (default: `mock`).

Source: `services/index.ts:21-30`

```
export const config = {
  mode: (process.env.NEXT_PUBLIC_SERVICE_MODE || 'mock') as 'mock' | 'production',
  endpoints: {
    sumsub: process.env.SUMSUB_API_URL || 'https://api.sumsub.com',
  },
}
```

>

> **Note on Swan:** Swan was previously listed as the Open Banking provider but has been deprecated. The pass-through PSD2 model will use a different AISP/PISP provider (TBD).
>
> **Note on Stripe:** Card issuing is a future feature gated behind feature flags. No Stripe SDK is integrated — only a mock file exists.
>
> **Note on Vipps/Nets:** Sometimes mentioned in business discussions but have ZERO code in the codebase.

Swan — Open Banking / PSD2 Provider [DEPRECATED]

> ⚠ DEPRECATED: Swan is no longer the planned Open Banking provider. Mock code remains but will be removed.

File: `services/mock-swan.ts` **Production docs:** https://docs.swan.io/ **Status:** DEPRECATED mock — no production integration, no contract, no API keys.

Interfaces

Interface	Description
`SwanAccount`	Bank account with IBAN, BIC, balance, status
`SwanTransaction`	SEPA credit/debit with status tracking

Functions

Function	Signature	Description
`createAccount`	`(userId) → SwanAccount`	Create new bank account with IBAN
`getAccount`	`(accountId) → SwanAccount | null`	Retrieve account details
`getBalance`	`(accountId) → {available, pending}`	Get balance breakdown
`initiateTransfer`	`({fromAccountId, toIban, amount, ...}) → SwanTransaction`	Initiate SEPA credit transfer

Function	Signature	Description
`simulateIncoming`	`({toAccountId, amount, fromIban}) → SwanTransaction`	Simulate incoming transfer
`getTransactions`	`(accountId, limit?) → SwanTransaction[]`	List recent transactions
`onWebhook`	`(callback) → void`	Register webhook listener

Mock Behavior

- 200-800ms simulated latency per call
- IBAN generated in `BA` format (Bosnia mock)
- Transfers start as `Pending`, settle to `Booked` after 2 seconds
- State persisted to `localStorage` (browser) or in-memory (server)
- `_testHelpers.reset()` clears all mock data

Account Statuses

`Opened` | `Closing` | `Closed`

Transaction Types

`SepaCredit` | `SepaDebit` | `CardTransaction`

Transaction Statuses

`Pending` | `Booked` | `Rejected`

Stripe — Card Issuing [MOCK/DEV]

> ⚠ MOCK ONLY: No Stripe SDK installed. Mock file for UI development only.

File: `services/mock-stripe.ts` **Production docs:** https://stripe.com/docs/issuing **Status:** Mock implementation only — no real Stripe API calls, no SDK, no API keys.

Interfaces

Interface	Description
`StripeCard`	Card with type, brand, status, spending limits
`StripeCardDetails`	Full card number, CVC, expiry (virtual only)
`StripeAuthorization`	Card authorization with merchant info

Functions

Function	Signature	Description
`createVirtualCard`	`({cardholderName, spendingLimit?}) → StripeCard`	Issue virtual Visa card
`orderPhysicalCard`	`({cardholderName, shippingAddress}) → StripeCard`	Order physical card
`getCardDetails`	`(cardId) → StripeCardDetails`	Get full card details (virtual only)
`setCardStatus`	`(cardId, active) → StripeCard`	Freeze/unfreeze card
`updateSpendingLimit`	`(cardId, limit) → StripeCard`	Update spending limit
`getCards`	`() → StripeCard[]`	List all cards
`simulateAuthorization`	`({cardId, amount, merchant}) → StripeAuthorization`	Simulate card purchase
`getAuthorizations`	`(cardId?) → StripeAuthorization[]`	List authorizations

Mock Behavior

- Virtual cards created instantly with `active` status, expire in 3 years
- Physical cards start as `pending`, activate after 5 seconds (simulating shipping)
- Default spending limit: 5,000 (virtual), 10,000 (physical)
- Authorization declined if: card not active OR spending limit exceeded
- Brand is always `Visa`
- Mock card numbers: `4242 4242 4242 {last4}`

Card Statuses

`active` | `inactive` | `canceled` | `pending`

Authorization Statuses

`pending` | `approved` | `declined`

Sumsub — KYC/Identity Verification [PRODUCTION]

> PRODUCTION-READY: Sumsub is the only external service with real production API integration.

File: `services/mock-sumsub.ts` **Production docs:** https://docs.sumsub.com/ **Status:** Production-ready — real API calls, WebSDK integration, webhook handling.

Interfaces

Interface	Description
`SumsubApplicant`	KYC applicant with review status
`SumsubDocument`	Identity document (passport, ID card, etc.)
`SumsubVerificationResult`	Verification outcome with per-check breakdown

Functions

Function	Signature	Description
`createApplicant`	`({externalUserId, email?, phone?}) → SumsubApplicant`	Create KYC applicant
`getAccessToken`	`(applicantId) → {token, expiresAt}`	Get WebSDK token (30min)
`submitDocument`	`(applicantId, document) → void`	Submit ID document
`submitSelfie`	`(applicantId, selfieData) → void`	Submit selfie for liveness
`getApplicantStatus`	`(applicantId) → SumsubApplicant`	Check applicant status
`getVerificationResult`	`(applicantId) → SumsubVerificationResult`	Get verification details
`forceApprove`	`(applicantId) → void`	Force approve (testing only)
`onWebhook`	`(callback) → void`	Register webhook listener

Mock Behavior

- Verification completes after 3-second delay
- 90% approval rate in mock mode

- Risk score: 15 (approved) or 85 (rejected)
- Rejected with label `DOCUMENT_UNREADABLE`, type `RETRY`

Applicant Statuses

`init` | `pending` | `queued` | `completed` | `onHold`

Review Answers

`GREEN` (approved) | `RED` (rejected) | `RETRY`

Verification Checks

Check	Description
documentAuthenticity	Document is genuine
livenessCheck	Selfie is a real person
facematch	Selfie matches document photo
sanctionsCheck	Not on sanctions lists
pepCheck	Not a politically exposed person

Document Types

`PASSPORT` | `ID_CARD` | `DRIVERS` | `RESIDENCE_PERMIT`

Service Initialization

Source: `services/index.ts:36-48`

```
// Call on app startup
await initializeServices()

// Reset all mocks (testing)
resetMockServices()
```

Service Status Summary

Service	Status	Description
Sumsub	`[PRODUCTION]`	Real API integration, WebSDK, webhook handling — READY
FX rates	`[IMPLEMENTED/RISK]`	`drop-api` uses Norges Bank primary + ECB fallback for cron refresh; see verified parser/conversion risk in Currency Rates
Stripe	`[MOCK/DEV]`	Mock file only for UI development — NO SDK, NO API keys
Swan	`[DEPRECATED]`	No longer the planned Open Banking provider — mock will be removed
Vipps	`[PLANNED]`	Future consideration — ZERO code currently
Nets	`[PLANNED]`	Future consideration — ZERO code currently

Important Notes

1. **Sumsub is the ONLY production-ready service** — all others are mocks or deprecated.
2. **Console warnings** are emitted on module load for mock services to make usage visible.
3. **Mock state** uses `localStorage` in browser, in-memory on server — resets on server restart.
4. **Production API endpoints** are configurable via environment variables.
5. **The current backend API routes do NOT call these service modules directly** — they use the database layer (`db.ts`) for all operations. The services are available for future integration when real providers are connected.

Revision #8
Created 2026-02-18 08:44:22 UTC by John
Updated 2026-07-29 00:12:28 UTC by John