Integration Design Document

Integration Design Document

> **Project:** Drop **Integration:** BankID OIDC + Open Banking (Neonomics AISP/PISP) + Sumsub KYC/AML **Version:** 1.0 **Date:** 2026-02-23 **Author:** Petter Graff, Senior Enterprise Architect **Status:** Approved **Reviewers:** Alem Bašić (CEO), John (AI Director)

Document History

Version	Date	Author	Changes
0.1	2026-02-23	Petter Graff	Initial draft from real integration docs

---

1. Integration Overview & Context

Integration Name: Drop External Integration Stack (BankID OIDC + Neonomics Open Banking + Sumsub KYC) **Type:** Synchronous (REST/HTTPS) + Asynchronous (Webhooks)

Business Purpose: Drop cannot function without these three integrations:

- **BankID:** Every user must authenticate with Norwegian Strong Customer Authentication — no other login method exists
- **Open Banking (Neonomics/ASPSP):** All balance reads (AISP) and payment initiations (PISP) require live bank API connectivity — Drop never holds funds
- **Sumsub KYC:** Norwegian AML law (hvitvaskingsloven) requires identity verification before any financial operation

Criticality:

- BankID: **Critical** — all logins blocked if down; RTO 1 hour
- Open Banking PISP: **Critical** — all payments blocked if down; RTO 2 hours; RPO 0 (no payment data lost — payments either completed at bank or not initiated)

- Open Banking AISP: **High** — balance display degrades to cached value; acceptable for 24 hours
- Sumsub: **High** — new user KYC blocked; existing approved users unaffected; RTO 4 hours

Parties:

Party	System	Team	Contact
Consumer (Drop)	drop-api (Hono v4)	ALAI Backend	john@alai.no
Provider: BankID	auth.bankid.no OIDC	BankID Norge	developer.bankid.no
Provider: Open Banking	Neonomics REST API	Neonomics	neonomics.io
Provider: KYC	Sumsub REST API + Webhooks	Sumsub	sumsub.com

---

2. Integration Topology Diagram

```
flowchart TB
    subgraph UserSide["User Devices"]
        Browser["Browser (Next.js)"]
        App["Mobile (Expo)"]
    end

    subgraph DropPlatform["Drop Platform (AWS eu-north-1)"]
        subgraph Edge["Cloudflare Edge"]
            CF["WAF + CDN + DDoS"]
        end
        subgraph App_["drop-web (Next.js BFF)"]
            WebBFF["Next.js API Routes\n/api/auth/bankid/*"]
        end
        subgraph API["drop-api (Hono v4)"]
            AuthRoute["routes/auth.ts\nBankID OIDC callback"]
            TxRoute["routes/transactions.ts\nPISP orchestration"]
            KYCRoute["routes/user.ts\nKYC initiation + webhooks"]
            WebhookRoute["POST /v1/webhooks/sumsub\nHMAC-verified"]
        end
        subgraph DB["PostgreSQL 16"]
            Users["users + sessions"]
            Txs["transactions"]
            Kyc["screening_results\nkyc_status"]
```

```
    end
end

subgraph BankIDProvider["BankID Norge (auth.bankid.no)"]
    BIDAuth["OIDC /auth endpoint"]
    BIDToken["OIDC /token endpoint"]
    BIDJWKS["JWKS /certs endpoint"]
end

subgraph Neonomics["Neonomics Open Banking"]
    NeoAISP["AISP: GET /v1/accounts/{id}/balances"]
    NeoPISP["PISP: POST /v1/payments/sepa-credit-transfers"]
    NeoCB["Circuit Breaker\n3 failures → 60s cooldown"]
end

subgraph SumsubProvider["Sumsub KYC"]
    SumAPI["REST API\n/resources/applicants"]
    SumWebhook["Webhooks (HMAC-signed)\nPOST → Drop /v1/webhooks/sumsub"]
    SumSDK["WebSDK (web) +\nReact Native SDK (mobile)"]
end

Browser & App --> CF
CF --> WebBFF & API
WebBFF --> BIDAuth
BIDToken --> AuthRoute
BIDJWKS --> AuthRoute
AuthRoute --> Users
TxRoute --> NeoCB --> NeoPISP & NeoAISP
KYCRoute --> SumAPI
SumWebhook --> WebhookRoute
WebhookRoute --> Kyc
SumSDK --> Browser & App
```

---

3. Service Contracts

3.1 Integration: BankID OIDC (Authentication)

Protocol: OpenID Connect 1.0 Authorization Code Flow over HTTPS **Direction:** Drop → BankID Norge **Idempotency:** YES — `state` and `nonce` parameters per-request

Authentication

Method	Details
Type	OAuth2 Authorization Code with Client Secret
Client ID Header	Sent in token exchange POST body: `client_id=BANKID_CLIENT_ID`
Client Secret	Sent in token exchange POST body: `client_secret=BANKID_CLIENT_SECRET`
Key rotation	BankID JWKS keys rotate periodically — jose library handles automatic JWKS refresh
Token endpoint	`https://auth.bankid.no/auth/realms/prod/protocol/openid-connect/token`

Request Contract — Step 1: Initiate (Redirect)

Endpoint: `GET https://auth.bankid.no/auth/realms/prod/protocol/openid-connect/auth`

Query Parameters:

```
client_id=BANKID_CLIENT_ID
redirect_uri=https://getdrop.no/api/auth/bankid/callback
response_type=code
scope=openid+profile
state=CRYPTO_RANDOM_UUID
nonce=CRYPTO_RANDOM_UUID
```

Drop stores state in: httpOnly cookie `bankid_state={state}` (web) or in-memory (mobile)

Request Contract — Step 2: Token Exchange

Endpoint: `POST https://auth.bankid.no/auth/realms/prod/protocol/openid-connect/token`

Request Body:

```
grant_type=authorization_code
code=AUTH_CODE
redirect_uri=https://getdrop.no/api/auth/bankid/callback
client_id=BANKID_CLIENT_ID
client_secret=BANKID_CLIENT_SECRET
```

Successful Response `200`:

```
{
  "id_token": "eyJ...",
  "access_token": "...",
  "token_type": "Bearer",
  "expires_in": 300
}
```

ID Token Claims Used by Drop

Claim	Type	Usage
`pid`	string	Norwegian fødselsnummer (11 digits). Hashed (SHA-256) → `users.national_id_hash`
`name`	string	Full name. Split into `users.first_name` + `users.last_name`
`sub`	string	Fallback subject identifier if `pid` absent
`iss`	string	Validated: `https://auth.bankid.no/auth/realms/prod`
`aud`	string	Validated: matches `BANKID_CLIENT_ID`
`exp`	number	Token expiry — verified via jose
`nonce`	string	Anti-replay — matched against stored nonce

Error Handling

HTTP Status	Error Code	Drop Action
User cancels BankID	N/A (redirect with `error=access_denied`)	Return `bankid_cancelled` 400
`400`	`invalid_grant`	Return `token_exchange_failed` 502
`401`	`unauthorized_client`	Alert ops — client ID invalid
JWKS verification fails	N/A	Return `jwks_verification_failed` 502 — alert ops
`pid` age check fails (< 18)	N/A	Return `underage` 403 — Norwegian legal requirement

Retry Policy

```
Token exchange: No retry (state-dependent flow — retry means restarting from login)
JWKS fetch: jose handles with built-in caching (5-minute TTL)
```

```
On JWKS failure: Alert Sentry, return 502 to user
```

Rate Limiting

Limit	Value	Window	Action when exceeded
BankID initiate	10 req	60s per IP	HTTP 429 from Drop rate limiter
BankID callback	10 req	60s per IP	HTTP 429 from Drop rate limiter

---

3.2 Integration: Open Banking AISP (Balance Reads)

Protocol: Berlin Group NextGenPSD2 v1.3.12+ over HTTPS **Direction:** Drop → Neonomics → ASPSP (user's bank) **Idempotency:** YES — `X-Request-ID` UUID per request

Authentication

Method	Details
Type	OAuth2 Client Credentials (Neonomics) + eIDAS QWAC certificate (ASPSP direct)
Header	`Authorization: Bearer {neonomics_access_token}`
Key rotation	OAuth2 token refresh; eIDAS cert rotation annually
Token endpoint	`https://api.neonomics.io/auth/token`

Request Contract — AISP Balance Read

Endpoint: `GET https://api.neonomics.io/v1/accounts/{accountId}/balances`

Headers:

```
Authorization: Bearer {NEONOMICS_ACCESS_TOKEN}
X-Request-ID: {UUID}
X-Consent-ID: {consentId}
Content-Type: application/json
```

Successful Response `200`:

```
{
  "balances": [
```

```
    {
      "balanceType": "expected",
      "balanceAmount": {
        "currency": "NOK",
        "amount": "45230.00"
      },
      "lastChangeDateTime": "2026-02-23T08:00:00.000Z"
    }
  ]
}
```

PSD2 Constraint: Maximum 4 TPP-initiated balance reads per account per day (RTS Art. 36(6)). User-initiated reads are unlimited.

Drop caches in: `bank_accounts.balance` (in øre) + `bank_accounts.balance_synced_at`

Error Handling

HTTP Status	Berlin Group Code	Drop Handling
`403`	`CONSENT_EXPIRED`	Delete consent, prompt user to re-link bank account
`429`	`ACCESS_EXCEEDED`	Back off, show cached balance with timestamp
`500+`	Server error	Circuit breaker, show cached balance

Circuit Breaker Configuration

```
Failure threshold: 3 failures in 60s window
Open duration: 60s
Half-open test: 1 request
Alert on: Circuit open for > 5 minutes → Sentry HIGH alert
Fallback: Return cached balance from bank_accounts.balance with staleness warning
```

3.3 Integration: Open Banking PISP (Payment Initiation)

Protocol: Berlin Group NextGenPSD2 v1.3.12+ over HTTPS **Direction:** Drop → Neonomics → ASPSP (user's bank) **Idempotency:** YES — `X-Request-ID` = `idempotency_key` from `transactions` table

Request Contract — PISP Remittance

Endpoint: `POST https://api.neonomics.io/v1/payments/sepa-credit-transfers`

Headers:

```
Authorization: Bearer {NEONOMICS_ACCESS_TOKEN}
Content-Type: application/json
X-Request-ID: {idempotency_key}
X-Consent-ID: {pisConsentId}
```

Request Body:

```
{
  "debtorAccount": {
    "iban": "NO9386011117947"
  },
  "instructedAmount": {
    "currency": "NOK",
    "amount": "2010.00"
  },
  "creditorName": "Marko Petrovic",
  "creditorAccount": {
    "bban": "265-1234567-89"
  },
  "remittanceInformationUnstructured": "Drop remittance tx_rem_abc123"
}
```

Successful Response `201`:

```
{
  "paymentId": "pay_xyz123",
  "transactionStatus": "RCVD",
  "_links": {
    "scaRedirect": {
      "href": "https://dnb.no/sca/pay/abc123"
    }
  }
}
```

SCA Redirect Flow: Drop returns `scaRedirect` URL to client → client redirects user to bank → user authenticates with BankID at bank → bank redirects back to Drop callback → Drop polls payment status.

Retry Policy

```
Max retries: 3 (retry only on 500, 502, 503 — NOT on 4xx)
Strategy: Exponential backoff with jitter
Delays: [1000ms, 2000ms, 4000ms]
Timeout per attempt: 30000ms
Idempotency: X-Request-ID = idempotency_key prevents double-payment on retry
```

Timeout Configuration

Timeout Type	Value	Notes
Connection timeout	5000ms	Fail fast if Neonomics unreachable
Read timeout	30000ms	ASPSP processing can take up to 30s
SCA callback timeout	300s (5 min)	Mark transaction `failed` if no callback received

---

3.4 Integration: Sumsub KYC/AML

Protocol: REST HTTPS (outbound) + Webhooks HTTPS (inbound) **Direction:** Drop → Sumsub (applicant creation), Sumsub → Drop (webhook status updates) **Idempotency:** YES — webhook idempotency via `screening_results` table check

Authentication

Method	Details
Outbound (Drop → Sumsub)	`Authorization: Bearer {SUMSUB_APP_TOKEN}`
Inbound Webhook Verification	HMAC-SHA256 of request body using `SUMSUB_SECRET_KEY`
Webhook Header	`X-Payload-Digest: HMAC-SHA256(body, SUMSUB_SECRET_KEY)`
Key rotation	Manual rotation in Sumsub dashboard; update `SUMSUB_SECRET_KEY` in Secrets Manager

Request Contract — KYC Initiation

Endpoint: `POST https://api.sumsub.com/resources/applicants`

Headers:

```
Authorization: Bearer {SUMSUB_APP_TOKEN}
Content-Type: application/json
X-Request-ID: {UUID}
```

Request Body:

```
{
  "externalUserId": "usr_abc123def456gh78",
  "email": "usr_abc123@bankid.drop.local",
  "levelName": "basic-kyc-level"
}
```

Successful Response `201`:

```
{
  "id": "5f9e1b2c3d4e5f6g7h8i9j0k",
  "externalUserId": "usr_abc123def456gh78",
  "review": {
    "reviewStatus": "init"
  }
}
```

Webhook Contract (Sumsub ? Drop)

Endpoint: `POST /v1/webhooks/sumsub` **Verification:** `HMAC-SHA256(rawBody, SUMSUB_SECRET_KEY)` must match `X-Payload-Digest` header

Webhook Payload:

```
{
  "type": "applicantReviewed",
  "applicantId": "5f9e1b2c3d4e5f6g7h8i9j0k",
  "externalUserId": "usr_abc123def456gh78",
  "reviewResult": {
    "reviewAnswer": "GREEN",
    "rejectLabels": []
  },
  "levelName": "basic-kyc-level",
  "createdAt": "2026-02-23T10:00:00.000Z"
}
```

Drop Action per Event:

Event Type	Drop `kyc_status` Change	Side Effects
`applicantReviewed` + GREEN	`pending` → `approved`	notification: "Du er nå verifisert", audit_log

Event Type	Drop `kyc_status` Change	Side Effects
`applicantReviewed` + RED	`pending` → `rejected`	notification: "Verifisering avslått", audit_log, AML check
`applicantReviewed` + RETRY	stays `pending`	notification: "Vennligst prøv igjen", audit_log
`applicantPending`	stays `pending`	audit_log only
`applicantOnHold`	stays `pending`	audit_log only

Idempotency Strategy for Webhooks

```
For each webhook delivery:
1. Check screening_results WHERE user_id = ? AND screening_type = 'kyc' AND result = new_result
2. If found (duplicate): return 200 OK, skip processing
3. If not found: process, INSERT INTO screening_results, UPDATE users.kyc_status
4. Return 200 OK to prevent Sumsub retry
```

Sumsub Retry Policy (on non-2xx response):

- 8 retry attempts over 7h 42m (immediate → 30s → 2m → 10m → 30m → 1h → 2h → 4h)

Rate Limiting

Limit	Value	Notes
Applicant creation	100 req/min	Sumsub API limit
Webhook endpoint	No rate limit	Must always return 200 quickly

---

4. Event-Driven Integrations

4.1 Sumsub Webhook Events

Published by: Sumsub **Consumed by:** Drop `POST /v1/webhooks/sumsub`

Event: `applicantReviewed`

```
{
  "type": "applicantReviewed",
  "applicantId": "sumsub_internal_id",
```

```
  "externalUserId": "usr_abc123def456gh78",
  "inspectionId": "inspection_id",
  "correlationId": "correlation_id",
  "levelName": "basic-kyc-level",
  "sandboxMode": false,
  "reviewStatus": "completed",
  "createdAt": "2026-02-23T10:00:00.000Z",
  "reviewResult": {
    "reviewAnswer": "GREEN",
    "rejectLabels": [],
    "reviewRejectType": null,
    "moderationComment": null
  }
}
```

4.2 Topics / Queues

Drop does not use a message broker at current scale. Webhook delivery is direct HTTP. Internal side effects use synchronous DB writes.

4.3 Ordering Guarantees

Integration	Ordering	Notes
BankID OIDC callback	Per-session (stateful via state cookie)	State cookie ensures correct session
Sumsub webhooks	Best-effort	Idempotency key prevents duplicate processing
Open Banking payment callbacks	Per-payment (paymentId)	Poll status if callback missing

4.4 Idempotency Strategy

```
BankID callback:
  State cookie + nonce = per-session idempotency; JWT issuance is idempotent (same pid = same user)

Open Banking PISP:
  X-Request-ID = transactions.idempotency_key
  Unique DB index ensures duplicate INSERT is rejected → return existing transaction
```

```
Sumsub webhooks:
  Check screening_results for existing result before processing
  Duplicate: acknowledge (200 OK), skip processing
```

---

5. Data Consistency Patterns

5.1 Consistency Model

Model: Strong (within Drop DB) + Eventual (between Drop and external systems) **Acceptable lag:** PISP status lag: 5 minutes max; AISP balance lag: 6 hours (PSD2 constraint)

5.2 PISP Payment Saga

```
sequenceDiagram
    autonumber
    participant Drop as Drop API
    participant DB as PostgreSQL
    participant PISP as Neonomics PISP
    participant ASPSP as User's Bank
    participant User as User

    Drop->>DB: INSERT transactions (status='processing', idempotency_key)
    Drop->>DB: COMMIT
    Drop->>PISP: POST /v1/payments/sepa-credit-transfers (X-Request-ID=idempotency_key)
    PISP-->>Drop: {paymentId, scaRedirect}
    Drop-->>User: 201 + scaRedirect URL
    User->>ASPSP: Complete BankID SCA at bank
    ASPSP-->>User: Redirect to Drop callback
    User->>Drop: GET /api/payments/callback?paymentId=xxx
    Drop->>PISP: GET /v1/payments/{paymentId}/status
    PISP-->>Drop: {transactionStatus: "ACCP"}
    Drop->>DB: UPDATE transactions SET status='completed'
```

Compensation strategies:

Step	Compensation	Notes
DB INSERT failed	Rollback automatically — no PISP call made	Clean state

Step	Compensation	Notes
PISP initiation failed	Transaction stays `processing`; PISP idempotency key prevents double charge on retry	Alert ops if persistent
SCA timeout (no callback in 5min)	UPDATE transactions SET status='failed'; restore cached balance	User notified to retry

---

6. Integration Testing Strategy

6.1 Contract Testing

- BankID: Integration tests against BankID test environment (`BANKID_MOCK=false`, BankID preprod)
- Neonomics: Integration tests against Neonomics sandbox
- Sumsub: Mock SDK (`NEXT_PUBLIC_SERVICE_MODE=mock`) for unit tests; staging Sumsub account for integration

6.2 Integration Test Environments

Environment	Purpose	Trigger
Local (BANKID_MOCK=true)	Dev testing with BankID mock	Manual
Staging	Full integration with BankID test + Neonomics sandbox + Sumsub staging	Every PR merge
Production	Synthetic monitoring via GET /v1/health	Every 5 minutes

6.3 Test Scenarios

Happy path:

- ☑ BankID login → JWT issued → dashboard loads
- ☑ Sumsub KYC initiated → webhook GREEN → kyc_status=approved
- ☐ AISP balance read → cached in bank_accounts (Phase 2)
- ☐ PISP remittance → SCA redirect → payment completed (Phase 2)

Error scenarios:

- ☑ BankID auth cancelled → 400 `bankid_cancelled`
- ☑ User under 18 → 403 `underage`
- ☑ Sumsub webhook with invalid HMAC → 401 (rejected)
- ☑ Duplicate Sumsub webhook → 200 (idempotent skip)
- ☐ Neonomics API returns 500 → circuit breaker opens → 502 (Phase 2)
- ☐ PISP SCA timeout → transaction marked failed (Phase 2)

---

7. Monitoring & Alerting

7.1 Key Metrics

Metric	Type	Alert Condition	Severity
`bankid_login_errors_total`	Counter	> 10/min	HIGH
`bankid_underage_rejections_total`	Counter	> 5/min (unusual spike)	MEDIUM
`pisp_payment_failures_total`	Counter	> 5/min	CRITICAL
`pisp_circuit_open`	Gauge	== 1	CRITICAL
`sumsub_webhook_failures_total`	Counter	> 0	HIGH
`aisp_balance_staleness_hours`	Gauge	> 12h	MEDIUM
`kyc_approval_rate`	Gauge	< 70% over 1h	HIGH

7.2 Distributed Tracing

- **Trace ID propagation:** `x-request-id` header generated per request (UUID), propagated to all external calls
- **Sampling rate:** 100% in staging, 10% in production (Sentry performance monitoring)
- **Tracing tool:** Sentry Performance — transactions tracked per endpoint

7.3 Alert Routing

Condition	Alert Channel	Escalation
BankID OIDC unreachable	Sentry HIGH alert	On-call engineer via Sentry
PISP circuit breaker open	Sentry CRITICAL alert	On-call + Alem notification
Sumsub webhook HMAC failure	Sentry HIGH alert	Check SUMSUB_SECRET_KEY rotation

Condition	Alert Channel	Escalation
AISP balance > 12h stale	Sentry MEDIUM alert	Investigate Neonomics API

Approval

Role	Name	Date	Signature
Author	Petter Graff	2026-02-23	
Consumer Team Lead	John (AI Director)		
Provider Team Lead	External (BankID/Neonomics/Sumsub)		
Platform/Infra			
Approver (CEO)	Alem Bašić		

Revision #5
Created 2026-02-23 12:05:01 UTC by John
Updated 2026-05-31 20:03:10 UTC by John