Bank Account Linking Flow

Low-Level Design: Bank Account Linking Flow

Version: 1.0 **Date:** 2026-02-21 **Author:** Banking Architecture Team **Status:** Approved **Applies to:** Drop — AISP Consent & Bank Account Linking

---

1. Overview

Bank account linking is the process where a Drop user connects their bank account via Open Banking (PSD2 AISP). This enables Drop to:

- Read the user's bank account balance (displayed on Dashboard)
- Verify sufficient funds before initiating PISP payments
- Display linked accounts in the Bank Accounts screen (`/accounts`)

The linking flow requires **AISP consent** from the user's bank (ASPSP), authenticated via **BankID SCA** at the bank. Drop stores the consent reference and caches the balance in the `bank_accounts` table.

Key principle: Drop never holds money. The `bank_accounts.balance` column is a **cached read** from the user's real bank account via AISP.

---

2. Complete Bank Linking Flow

```
sequenceDiagram
    participant U as User
    participant UI as Drop UI<br/>(/accounts)
    participant API as Drop API
    participant DB as Drop DB
    participant ASPSP as User's Bank<br/>(e.g., DNB)
```

```
    Note over U,ASPSP: Step 1: Bank Selection
    U->>UI: Tap "Koble til bank" (Link bank)
    UI->>UI: Show bank selection list<br/>(DNB, SpareBank 1, Nordea, ...)
    U->>UI: Select "DNB"

    Note over U,ASPSP: Step 2: AISP Consent Request
    UI->>API: POST /api/accounts/link<br/>{bankId: "dnb"}
    API->>API: Verify user authenticated<br/>(JWT from drop_token cookie)
    API->>ASPSP: POST /v1/consents<br/>{access: {balances: ["allAccounts"],<br/>transactions:
["allAccounts"]},<br/>recurringIndicator: true,<br/>validUntil: "2026-05-
22",<br/>frequencyPerDay: 4,<br/>combinedServiceIndicator: false}
    ASPSP-->>API: 201 Created<br/>{consentId: "cons_abc123",<br/>consentStatus:
"received",<br/>_links: {scaRedirect:<br/>"https://dnb.no/psd2/consent/authorize?id=..."}}
    API->>DB: INSERT INTO consents<br/>(consent_type: 'psd2_aisp',<br/>granted: 0,
aspsp_consent_id: cons_abc123)
    API-->>UI: {redirectUrl: "https://dnb.no/psd2/consent/authorize?id=..."}

    Note over U,ASPSP: Step 3: SCA at Bank
    UI->>U: Redirect to bank consent page
    U->>ASPSP: View consent details<br/>"Drop requests access to your<br/>account balances and
transactions"
    U->>ASPSP: Authenticate with BankID<br/>(possession + knowledge/inherence)
    ASPSP-->>U: Consent granted<br/>Redirect to Drop callback

    Note over U,ASPSP: Step 4: Callback & Account Retrieval
    U->>API: GET /api/accounts/link/callback<br/>?consentId=cons_abc123&state=xyz
    API->>API: Verify state parameter
    API->>ASPSP: GET /v1/consents/cons_abc123/status
    ASPSP-->>API: {consentStatus: "valid"}
    API->>DB: UPDATE consents SET granted = 1,<br/>granted_at = now

    API->>ASPSP: GET /v1/accounts<br/>(with consentId header)
    ASPSP-->>API: {accounts: [<br/>{resourceId: "acc_1", iban:
"NO1234567890123",<br/>currency: "NOK", name: "Brukskonto"},<br/>{resourceId: "acc_2", iban:
"NO9876543210987",<br/>currency: "NOK", name: "Sparekonto"}]}

    Note over U,ASPSP: Step 5: Balance Fetch & Storage
    loop For each account
        API->>ASPSP: GET /v1/accounts/{resourceId}/balances
```

```
        ASPSP-->>API: {balances: [{balanceType: "expected",<br/>balanceAmount: {currency:
"NOK",<br/>amount: "45230.00"}}]}
        API->>DB: INSERT INTO bank_accounts<br/>(bank_name: "DNB",<br/>account_number:
"NO12...0123",<br/>iban: "NO1234567890123",<br/>balance: 4523000,<br/>balance_synced_at:
now,<br/>is_primary: first ? 1 : 0)
    end

    API-->>UI: {success: true,<br/>accounts: [{bankName: "DNB",<br/>balance: 45230.00,
currency: "NOK"}]}
    UI-->>U: "DNB koblet til!" (DNB linked!)<br/>Show accounts with balances
```

3. Linked Account States

```
stateDiagram-v2
    [*] --> Unlinked: User has no linked bank accounts

    Unlinked --> ConsentRequested: User taps "Koble til bank"<br/>POST /v1/consents to ASPSP
    ConsentRequested --> ScaPending: ASPSP returns scaRedirect
    ScaPending --> Active: User completes BankID SCA<br/>consentStatus = "valid"
    ScaPending --> Failed: SCA timeout (5 min)
    ScaPending --> Failed: User cancels BankID
    ScaPending --> Failed: Bank rejects consent

    Active --> Active: Balance refresh<br/>(on-demand or scheduled,<br/>max 4x/day TPP-
initiated)
    Active --> SyncError: ASPSP returns error on balance read<br/>Show last cached balance
    SyncError --> Active: Next successful balance read
    Active --> ConsentExpiring: 30 days before consent expiry<br/>Notify user to renew
    ConsentExpiring --> RenewalPending: User taps "Forny tilgang"<br/>(Renew access)
    RenewalPending --> ScaPending: New consent + SCA
    ConsentExpiring --> Expired: User ignores renewal

    Active --> Unlinked: User taps "Fjern konto"<br/>(Remove account)<br/>DELETE
/v1/consents/{id}
    Active --> Suspended: ASPSP revokes consent
    Expired --> Unlinked: Consent expired,<br/>balance zeroed,<br/>notify user
    Suspended --> Unlinked: User must re-link
    Failed --> Unlinked: User can retry
```

```
    Unlinked --> [*]
```

---

4. Detailed Steps

4.1 Step 1: Bank Selection

UI: Bank Accounts screen (`/accounts`) shows a "Link bank account" button. Tapping it displays a list of supported Norwegian banks.

Supported banks (initial):

Bank	Berlin Group API	Logo
DNB	`https://api.dnb.no/psd2/`	DNB logo asset
SpareBank 1	`https://api.sparebank1.no/open-banking/`	SB1 logo asset
Nordea	`https://api.nordeaopenbanking.com/`	Nordea logo asset
Sbanken	Via SpareBank 1 API	Sbanken logo asset

4.2 Step 2: AISP Consent Creation

Berlin Group consent request:

```
{
  "access": {
    "balances": [{"iban": "allAccounts"}],
    "transactions": [{"iban": "allAccounts"}]
  },
  "recurringIndicator": true,
  "validUntil": "2026-05-22",
  "frequencyPerDay": 4,
  "combinedServiceIndicator": false
}
```

Field	Value	Reason
`balances`	`allAccounts`	User picks which accounts to display after consent

Field	Value	Reason
`transactions`	`allAccounts`	Future: transaction history from bank
`recurringIndicator`	`true`	Ongoing access (not one-time)
`validUntil`	Now + 90 days	PSD2 maximum consent duration (RTS Art. 10)
`frequencyPerDay`	`4`	Max TPP-initiated reads per day (PSD2 RTS Art. 36(6))
`combinedServiceIndicator`	`false`	AISP consent only (PISP is separate per transaction)

4.3 Step 3: SCA at Bank

The user is redirected to their bank's consent page where they:

1. See what data Drop is requesting (balances, transactions)
2. Authenticate with BankID (SCA: 2 of 3 factors)
3. Approve or deny the consent
4. Get redirected back to Drop

4.4 Step 4: Account Retrieval

After consent is confirmed, Drop calls the ASPSP's account list endpoint to discover which accounts the user has. Then it fetches the balance for each account.

4.5 Step 5: Data Storage

Each linked account is stored in the `bank_accounts` table:

Column	Value	Source
`id`	`ba_<hex16>`	Generated by `randomId("ba")`
`user_id`	Current user ID	From JWT
`bank_name`	"DNB"	From bank selection
`account_number`	Domestic format (masked in API)	From ASPSP account details
`iban`	Full IBAN	From ASPSP
`balance`	Balance in oere (e.g., 4523000 = 45,230 NOK)	From ASPSP balance endpoint
`balance_synced_at`	Current timestamp	Set on each refresh
`currency`	"NOK"	From ASPSP

Column	Value	Source
`is_primary`	1 for first account, 0 for others	Auto-set
`connected_at`	Current timestamp	Set on linking

---

5. Balance Refresh Strategy

Trigger	Frequency	ASPSP Call	UI Behavior
User opens Dashboard	On demand	GET `/v1/accounts/{id}/balances`	Show spinner, then updated balance
User pulls to refresh	On demand	GET `/v1/accounts/{id}/balances`	Pull-to-refresh animation
Background sync	Every 6 hours	GET `/v1/accounts/{id}/balances`	Silent update
Pre-payment check	Before PISP	GET `/v1/accounts/{id}/balances`	Inline balance verification
User views /accounts	On demand	GET `/v1/accounts/{id}/balances`	Show spinner per account

PSD2 constraint: TPP-initiated requests (background sync) are limited to **4 per day per account** (RTS Art. 36(6)). User-initiated requests (opening app, pull-to-refresh) are **unlimited**.

---

6. Error Handling

6.1 Error Table

Error	Stage	HTTP Status	User Message (Norwegian)	Recovery
Bank not supported	Bank selection	400	"Denne banken stottes ikke ennaa."	Show supported banks
ASPSP API unreachable	Consent creation	502	"Kunne ikke koble til banken. Prv igjen senere."	Retry after 30s
SCA timeout	SCA at bank	408	"BankID-sesjonen utlp. Prv igjen."	Restart linking flow
SCA cancelled	SCA at bank	400	"Du avbrt tilkoblingen."	Offer retry button

Error	Stage	HTTP Status	User Message (Norwegian)	Recovery
SCA rejected	SCA at bank	403	"Banken avviste tilgangen."	Contact bank support
Consent invalid	Callback	403	"Tilgangsforesprselen var ugyldig."	Restart flow
State mismatch	Callback	403	"Sikkerhetssjekk feilet. Prv igjen."	Restart flow
No accounts found	Account retrieval	404	"Fant ingen kontoer hos denne banken."	Verify correct bank selected
Balance read failed	Balance fetch	502	"Kunne ikke hente saldo. Viser sist kjente."	Show cached balance with timestamp
Consent expired	Any balance read	403	"Tilgangen til banken din er utlpt. Koble til paa nytt."	Re-link flow
Rate limit exceeded	Balance read	429	"For mange foresprsler. Viser sist kjente saldo."	Show cached balance

6.2 Fallback Behavior

When a balance read fails, Drop shows the **last cached balance** with the `balance_synced_at` timestamp:

```
DNB Brukskonto
45 230,00 kr
Sist oppdatert: 2 timer siden
```

If the consent is expired or revoked, the balance is zeroed and the user sees:

```
DNB Brukskonto
-- kr
Tilgangen er utlopt. Koble til paa nytt.
[Koble til] button
```

---

7. Consent Renewal

AISP consents have a maximum validity of 90 days (PSD2 RTS Art. 10). Drop proactively prompts renewal:

Timeline	Action
Day 0	Consent created, `validUntil` = Day 90
Day 60	Push notification: "Din banktilgang utlper snart. Forny tilgangen."
Day 80	In-app banner on Dashboard: "Tilgangen til DNB utlper om 10 dager."
Day 85	Push notification: "Tilgangen utlper om 5 dager. Forny naa."
Day 90	Consent expires. Balance zeroed. User must re-link.

Renewal flow: Same as initial linking — new consent + BankID SCA at bank. The old consent is deleted after the new one is active.

---

8. Account Unlinking

When the user taps "Fjern konto" (Remove account):

1. `DELETE /v1/consents/{consentId}` at the ASPSP (revoke consent)
2. `DELETE FROM bank_accounts WHERE id = ?` in Drop DB
3. `UPDATE consents SET withdrawn_at = now WHERE aspsp_consent_id = ?`
4. If the removed account was primary, promote another linked account to primary
5. If no accounts remain, Dashboard shows "Koble til bank" prompt

---

9. Multi-Bank Support

Users can link accounts from multiple banks. Each linked account has its own AISP consent with its own lifecycle.

Dashboard display:

```
Dine bankkontoer (Your bank accounts)

DNB Brukskonto          45 230,00 kr  (primary)
DNB Sparekonto          12 800,00 kr
Nordea Brukskonto        8 450,00 kr

Totalt                  66 480,00 kr
```

```
[Koble til ny bank]
```

Drop API: `GET /api/auth/me` returns `totalBalance` (sum of all linked account balances) and `bankAccounts[]` array.

---

10. Database Impact

10.1 Tables Affected

Table	Operation	When
`bank_accounts`	INSERT	Account linked
`bank_accounts`	UPDATE (balance, balance_synced_at)	Balance refresh
`bank_accounts`	DELETE	Account unlinked
`consents`	INSERT (consent_type: 'psd2_aisp')	Consent created
`consents`	UPDATE (granted, granted_at)	SCA completed
`consents`	UPDATE (withdrawn_at)	Consent revoked
`audit_log`	INSERT	Every linking/unlinking action
`notifications`	INSERT	Consent expiry reminders

10.2 Index Usage

Index	Used By
`idx_bank_accounts_user` on `user_id`	All account queries (scoped to user)
`idx_consents_user` on `user_id`	Consent lookups for renewal checks

---

11. Cross-References

- **Open Banking AISP/PISP:** ../integration/open-banking-aisp-pisp.md — Berlin Group API details, consent properties
- **BankID OIDC:** ../integration/bankid-oidc-integration.md — Drop authentication (separate from bank SCA)
- **Security Architecture:** ../hld/security-architecture.md — Trust boundaries, data classification
- **Remittance Flow:** flow-remittance.md — Uses linked bank account for PISP

- **Database Schema:** ../../backend/DATABASE-SCHEMA.md — `bank_accounts`, `consents` tables
- **API Reference:** ../../backend/API-REFERENCE.md — `GET /api/auth/me` returns bank accounts

---

Revision #5
Created 2026-02-21 05:58:52 UTC by John
Updated 2026-05-23 10:51:54 UTC by John