BankID OIDC Integration

BankID OIDC Integration

Version: 1.0 **Date:** 2026-02-21 **Author:** Banking Architecture Team **Status:** Approved **Applies to:** Drop — Authentication via Norwegian BankID

---

1. Overview

Drop uses **BankID OIDC** (OpenID Connect) as its sole authentication method. BankID provides Strong Customer Authentication (SCA) out of the box — combining possession (mobile device / code generator) with knowledge (personal code) or inherence (biometrics).

Email/password login has been removed. All deprecated auth endpoints return **410 Gone**.

Property	Value
Identity Provider	BankID (prod: `auth.bankid.no`)
Protocol	OpenID Connect 1.0 (Authorization Code Flow)
Auth endpoint	`https://auth.bankid.no/auth/realms/prod/protocol/openid-connect/auth`
Token endpoint	`https://auth.bankid.no/auth/realms/prod/protocol/openid-connect/token`
JWKS endpoint	`https://auth.bankid.no/auth/realms/prod/protocol/openid-connect/certs`
Issuer	`https://auth.bankid.no/auth/realms/prod`
Scopes	`openid profile`
Response type	`code` (authorization code flow)
Source code	`src/drop-api/src/lib/bankid.ts`

---

2. BankID OIDC Flow

2.1 Web Flow (Next.js BFF)

```
sequenceDiagram
    participant B as Browser
    participant N as Next.js BFF<br/>/api/auth/bankid/*
    participant BID as BankID OIDC<br/>(auth.bankid.no)

    B->>N: GET /api/auth/bankid
    Note over N: Rate limit check (10/min per IP)
    N->>N: Generate state (crypto.randomUUID)<br/>Generate nonce (crypto.randomUUID)
    N->>N: Set httpOnly cookie: bankid_state={state}
    N-->>B: {redirectUrl:
"https://auth.bankid.no/auth/...?<br/>client_id=X&redirect_uri=Y&response_type=code<br/>&scope
=openid+profile&state=Z&nonce=N"}

    B->>BID: Browser redirects to BankID authorize URL
    Note over B,BID: User authenticates with BankID<br/>(BankID app / code generator /
biometrics)
    BID-->>B: 302 redirect to /api/auth/bankid/callback<br/>?code=AUTH_CODE&state=Z

    B->>N: GET /api/auth/bankid/callback?code=AUTH_CODE&state=Z
    N->>N: Verify state matches bankid_state cookie
    N->>BID: POST /token<br/>{grant_type: authorization_code,<br/>code: AUTH_CODE,
redirect_uri: Y,<br/>client_id: X, client_secret: S}
    BID-->>N: {id_token: "eyJ...", access_token: "...",<br/>token_type: "Bearer", expires_in:
300}

    N->>N: Verify id_token signature via JWKS
    N->>N: Validate: issuer, audience, expiry, nonce
    N->>N: Extract pid (fødselsnummer) from claims
    N->>N: Parse birthdate from pid, verify age >= 18
    N->>N: findOrCreateUser(pid, name)
    N->>N: Create session (sessions table)
    N->>N: Issue Drop JWT, set httpOnly cookie (drop_token)
    N-->>B: 302 redirect to /dashboard
```

2.2 Mobile Flow (Hono API + Expo)

```
sequenceDiagram
    participant M as Mobile App<br/>(Expo)
    participant H as Hono API<br/>/v1/auth/bankid/*
    participant BID as BankID OIDC

    M->>H: GET /v1/auth/bankid/initiate?platform=mobile
    Note over H: Rate limit check (10/min per IP)
    H->>H: Generate state + nonce
    H-->>M: {redirectUrl: "https://auth.bankid.no/...",<br/>state: "Z"}

    M->>M: Store state in memory
    M->>BID: Open BankID in secure browser<br/>(expo-web-browser)
    Note over M,BID: User authenticates with BankID
    BID-->>M: Deep link: drop://auth/callback<br/>?code=AUTH_CODE&state=Z

    M->>M: Verify state matches stored state
    M->>H: POST /v1/auth/bankid/callback<br/>{code: AUTH_CODE, state: Z, platform: "mobile"}
    H->>BID: POST /token (exchange code)
    BID-->>H: {id_token, access_token}
    H->>H: Verify id_token via JWKS
    H->>H: Extract pid, verify age >= 18
    H->>H: findOrCreateUser(pid, name)
    H->>H: Create session
    H->>H: Issue Drop JWT (7-day expiry)
    H-->>M: {token: "eyJ...", data: {id, name, role}}
    M->>M: Store token in AsyncStorage
```

---

3. Token Lifecycle

3.1 Drop JWT (Issued After BankID Auth)

```
stateDiagram-v2
    [*] --> Issued: BankID auth success<br/>→ createSession() + signJWT()
    Issued --> Active: Token in use<br/>(cookie or Bearer header)
    Active --> Active: API request<br/>→ verifySession() passes
    Active --> Expired: 7d expiry reached<br/>(all clients)
    Active --> Revoked: User logs out<br/>→ revokeAllSessions()
```

```
    Active --> Revoked: Session marked revoked<br/>(admin action)
    Expired --> Refreshed: POST /auth/refresh<br/>→ new JWT + new session
    Expired --> [*]: User must re-authenticate
    Revoked --> [*]: User must re-authenticate
    Refreshed --> Active
```

3.2 JWT Payload Structure

```
interface DropJwtPayload {
  userId: string;   // "usr_a1b2c3d4e5f6g7h8"
  email: string;    // "usr_xxx@bankid.drop.local" (placeholder)
  role: string;     // "user" | "merchant"
  iat: number;      // Issued at (Unix timestamp)
  exp: number;      // Expiry (iat + 7d, all clients)
  iss?: string;     // "drop-api" (Hono only)
  aud?: string;     // "drop" (Hono only)
}
```

3.3 Token Lifetimes

Platform	Token Location	Lifetime	Refresh
Web	httpOnly cookie (`drop_token`)	24 hours	`POST /api/auth/refresh`
Mobile	Bearer header (AsyncStorage)	7 days	`POST /v1/auth/refresh`

3.4 Session Tracking

Every JWT has a corresponding record in the `sessions` table:

Column	Purpose
`id`	Session ID (`ses_<hex16>`)
`user_id`	FK to `users.id`
`token_hash`	SHA-256 hash of the JWT string
`expires_at`	Token expiry timestamp
`revoked`	`0` = active, `1` = revoked

On every authenticated request, the middleware:

1. Extracts JWT from cookie (web) or Authorization header (mobile)
2. Verifies JWT signature and expiry with `jose.jwtVerify()`
3. Hashes the JWT with SHA-256
4. Looks up the session by `token_hash`
5. Verifies `revoked = 0` and `expires_at > now`
6. Rejects the request if any check fails

---

4. BankID Claims Mapping

4.1 ID Token Claims

BankID Claim	Type	Drop Usage	Stored In
`sub`	string	Subject identifier (BankID internal)	Fallback for `pid`
`pid`	string	Norwegian fødselsnummer (11 digits)	Hashed as `users.national_id_hash` (SHA-256)
`name`	string	Full name (e.g., "Ola Nordmann")	Split into `users.first_name` + `users.last_name`
`birthdate`	string	ISO date (not always present)	Parsed from `pid` instead (more reliable)
`iss`	string	Issuer URL	Validated against expected issuer
`aud`	string	Client ID	Validated against `BANKID_CLIENT_ID`
`exp`	number	Token expiry	Validated (must be in future)
`iat`	number	Issued at	Validated (must be reasonable)
`nonce`	string	Anti-replay	Matched against value sent in auth request

4.2 PID (Fødselsnummer) Processing

The `pid` (personal identification number) is an 11-digit Norwegian national ID that encodes the birthdate:

```
Format: DDMMYYNNNCC
  DD   = Day of birth (01-31)
```

```
  MM   = Month of birth (01-12)
  YY   = Year of birth (2 digits)
  NNN  = Individual number (000-999)
  CC   = Check digits (Luhn variant)

Century derivation (from individual number):
  NNN 000-499 → born 1900-1999
  NNN 500-999 → born 2000-2099
```

Source: `bankid.ts:37-51` (`parseBirthdateFromPid`)

Processing pipeline:

1. **Parse birthdate** from pid digits → ISO date string (`YYYY-MM-DD`)
2. **Verify age >= 18** by comparing birthdate to current date (`isOver18()`)
3. **Hash pid** with SHA-256 for storage: `crypto.createHash("sha256").update(pid).digest("hex")`
4. **Never store raw pid** — only the hash exists in the database (`users.national_id_hash`)

4.3 User Deduplication

Users are identified by `national_id_hash` (SHA-256 of their fødselsnummer). This ensures:

- The same person logging in on different devices gets the same account
- Re-authentication after token expiry reconnects to the existing account
- Future Vipps Login integration can match users by the same pid hash

Source: `bankid.ts:208-249` (`findOrCreateUser`)

---

5. User Creation (Auto-Registration)

BankID login automatically creates user accounts. There is no separate registration step.

Field	Value	Source
`id`	`usr_<hex16>`	`randomId("usr")`
`email`	`usr_xxx@bankid.drop.local`	Placeholder (BankID doesn't provide email)
`password_hash`	`EIDONLY`	Sentinel — no password authentication
`auth_provider`	`bankid`	Indicates BankID-only auth

Field	Value	Source
`first_name`	First word of `name` claim	Split from BankID name
`last_name`	Remaining words of `name` claim	Split from BankID name
`date_of_birth`	Parsed from pid	`parseBirthdateFromPid(pid)`
`kyc_status`	`approved`	BankID = verified identity
`kyc_method`	`bankid`	KYC via BankID
`kyc_verified_at`	Current timestamp	Set on creation
`national_id_hash`	SHA-256(pid)	For user deduplication
`role`	`user`	Default; upgradable to `merchant`

---

6. SCA Compliance

6.1 PSD2 SCA Requirements Met by BankID

PSD2 RTS Requirement	BankID Implementation	Status
Two of three factors (knowledge, possession, inherence)	BankID app (possession) + personal code (knowledge) or biometrics (inherence)	Met
Independence of factors	Separate device (BankID app) + separate knowledge factor	Met
Dynamic linking (for PISP)	Amount and payee displayed in BankID app during payment approval	Met (ASPSP-side)
Authentication code specific to amount + payee	BankID generates unique code per transaction	Met (ASPSP-side)
Session timeout	BankID sessions expire (configurable by ASPSP)	Met
Max 5 failed attempts	BankID locks after failed attempts	Met (BankID-side)

6.2 Drop's SCA Scope

Drop performs SCA at two levels:

1. **Drop authentication (login):** BankID OIDC — satisfies SCA for account access
2. **Payment SCA (PISP):** Delegated to ASPSP — user re-authenticates with BankID at the bank for each payment (see open-banking-aisp-pisp.md)

7. Error Handling

7.1 Error Matrix

Error Scenario	HTTP Status	Error Code	User Message (Norwegian)	Recovery Action
BankID auth cancelled by user	400	`bankid_cancelled`	"Du avbrøt BankID-innlogging."	Retry login
BankID auth timeout	408	`bankid_timeout`	"BankID-sesjonen utløp. Prøv igjen."	Retry login
State mismatch (CSRF)	403	`state_mismatch`	"Sikkerhetssjekk feilet. Prøv igjen."	Restart flow
Token exchange failed	502	`token_exchange_failed`	"Kunne ikke koble til BankID. Prøv igjen."	Retry later
JWKS verification failed	502	`jwks_verification_failed`	"Teknisk feil. Prøv igjen senere."	Alert ops, retry
Invalid pid format	422	`invalid_pid`	"Ugyldig identifikasjon fra BankID."	Contact support
User under 18	403	`underage`	"Du må være minst 18 år for å bruke Drop."	No recovery — legal requirement
`BANKID_CLIENT_ID` missing	500	`config_error`	"Teknisk feil. Prøv igjen senere."	Fix server config
Session revoked	401	`session_revoked`	"Sesjonen din er utløpt. Logg inn på nytt."	Re-authenticate
JWT expired	401	`token_expired`	"Sesjonen din er utløpt. Logg inn på nytt."	Refresh or re-authenticate
Rate limited	429	`rate_limited`	"For mange forsøk. Vent litt og prøv igjen."	Wait and retry

7.2 Error Flow

```
flowchart TD
    A[BankID auth attempt] --> B{Auth successful?}
    B -->|Yes| C[Exchange code for tokens]
```

```
    B -->|No: cancelled| D[Return bankid_cancelled]
    B -->|No: timeout| E[Return bankid_timeout]

    C --> F{Token exchange OK?}
    F -->|Yes| G[Verify ID token via JWKS]
    F -->|No| H[Return token_exchange_failed<br/>Log error, alert ops]

    G --> I{JWKS valid?}
    I -->|Yes| J[Extract pid from claims]
    I -->|No| K[Return jwks_verification_failed<br/>Check JWKS URL, cert rotation]

    J --> L{Valid pid format?}
    L -->|Yes| M{Age >= 18?}
    L -->|No| N[Return invalid_pid]

    M -->|Yes| O[findOrCreateUser]
    M -->|No| P[Return underage<br/>403 Forbidden]

    O --> Q[Create session + JWT]
    Q --> R[Redirect to /dashboard]
```

---

8. Mock Mode (Development)

When `BANKID_MOCK=true`, the OIDC flow is simulated locally without contacting BankID:

Mock Code Prefix	Mock User	Birthdate	Age Check
`underage*`	"Ung Testbruker" (pid: `01011012345`)	2010-01-01	Fails (under 18)
(anything else)	"Test Bankersen" (pid: `01019012345`)	1990-01-01	Passes

Source: `bankid.ts:188-195` (`getMockUserInfo`)

Mock flow:

1. `initiateOIDC()` generates a redirect URL (same structure, but not called)
2. Frontend skips BankID redirect, posts a mock code to callback
3. `exchangeAndVerify()` detects `isMock=true` and returns mock user info
4. `findOrCreateUser()` runs normally (creates real DB records)

9. Production Migration Checklist

Step	Description	Status
1	Register BankID OIDC client at `developer.bankid.no`	Not started
2	Obtain `BANKID_CLIENT_ID` and `BANKID_CLIENT_SECRET`	Not started
3	Configure callback URLs (web + mobile deep link)	Not started
4	Set `BANKID_MOCK=false` (or remove env var)	Pending
5	Test OIDC flow in BankID preprod environment	Not started
6	Configure JWKS caching (jose handles this, verify TTL)	Pending
7	Set secure `JWT_SECRET` (min 32 chars, from vault)	Pending
8	Verify nonce validation in callback	Implemented in code
9	Test underage rejection with real BankID test user	Not started
10	Test session revocation and re-authentication	Implemented in code
11	Move to production BankID endpoints	Not started
12	Monitor JWKS key rotation (BankID rotates keys)	Pending

10. Phase 2: Vipps Login (Planned)

Vipps Login uses the same OIDC protocol. Integration plan:

1. Register Vipps Login client at `portal.vipps.no`
2. Add Vipps OIDC endpoints alongside BankID
3. Vipps also returns Norwegian pid (fødselsnummer)
4. User deduplication via `national_id_hash` — same hash regardless of whether user logs in with BankID or Vipps
5. UI: Login screen offers two buttons: "Logg inn med BankID" and "Logg inn med Vipps"

Optional: Use an OIDC aggregator like **Idura** to get a single integration point for BankID + Vipps + future providers.

---

11. Environment Variables

Required (Production)

Variable	Description	Example
`BANKID_CLIENT_ID`	OIDC client ID from BankID	`abc123-def456`
`BANKID_CLIENT_SECRET`	OIDC client secret	`secret-from-bankid`
`BANKID_CALLBACK_URL`	Web callback URL	`https://getdrop.no/api/auth/bankid/callback`
`BANKID_CALLBACK_URL_MOBILE`	Mobile deep link callback	`drop://auth/callback`
`JWT_SECRET`	Drop JWT signing secret (min 32 chars)	(from Vaultwarden)

Optional

Variable	Description	Default
`BANKID_AUTHORIZE_URL`	Override authorize endpoint	BankID prod URL
`BANKID_TOKEN_URL`	Override token endpoint	BankID prod URL
`BANKID_JWKS_URL`	Override JWKS endpoint	BankID prod URL
`BANKID_ISSUER`	Override expected issuer	BankID prod issuer
`BANKID_MOCK`	Enable mock mode (`true`/`false`)	`false`
`JWT_ALGORITHM`	JWT signing algorithm	`HS256`
`JWT_EXPIRY`	Token lifetime	`24h`

---

12. Cross-References

- **Authentication System:** ../../backend/AUTHENTICATION.md — Full auth system documentation
- **Open Banking AISP/PISP:** open-banking-aisp-pisp.md — ASPSP SCA (separate from BankID login SCA)

- **Security Architecture:** ../hld/security-architecture.md — JWT security, session management
- **Login Flow (LLD):** ../lld/flow-login-authentication.md — Step-by-step login UX
- **Database Schema:** ../../backend/DATABASE-SCHEMA.md — `users`, `sessions` tables
- **Source:** `src/drop-api/src/lib/bankid.ts` — BankID OIDC implementation

Revision #5
Created 2026-02-21 05:59:03 UTC by John
Updated 2026-05-23 10:56:59 UTC by John