Development Rules

Development Rules

Seven mandatory rules for all Tok development. These rules exist because financial data and PSD2 compliance leave no room for shortcuts.

Rule 1 — Consent Immutability

PSD2 consent records are append-only audit trails.

Never update or delete a consent record. Each state change creates a new log entry:

```
consent_created → consent_exchanged → consent_active
                                    → consent_expired (90 days)
                                    → consent_revoked (user action)
```

All consent events are logged to `LoggedAction` (append-only). This is a legal compliance requirement under PSD2.

Rule 2 — Token Encryption Mandatory

AES-256-GCM + GCP Cloud KMS for ALL OAuth tokens. No exceptions.

```
CORRECT:  Store tokens via Cloud KMS envelope encryption → encrypted_dek + iv + ciphertext in DB
WRONG:    Store raw tokens in DB or env vars
WRONG:    Store tokens in logs, files, or memory beyond request lifecycle
```

QWAC private key must also live in GCP Cloud KMS HSM — signing is done via Cloud KMS API, the key is never extracted.

Rule 3 — Bank Adapter Pattern

Every bank integration goes through the abstract `BankAdapter` interface.

```
// Correct
class BerlinGroupAdapter : BankAdapter { ... }
class BilateralAdapter   : BankAdapter { ... }

// Wrong — never call bank HTTP endpoints directly from services/routes
```

The adapter is the only layer that knows about bank-specific API formats. Services above it work only with normalized `BankTransaction` objects.

Rule 4 — Deduplication via externalId

`externalId` (bank's own transaction ID) + `bankAccountId` = unique constraint.

Duplicate imports are silently skipped — this is intentional, not an error. Never create duplicate transactions.

```
ALTER TABLE bank_transactions
ADD CONSTRAINT uq_bank_account_external
UNIQUE (bank_account_id, external_id);
```

Rule 5 — Money = NUMERIC(19,4)

Never use float or double for financial amounts.

```
// Correct
val amount: BigDecimal  // Kotlin
// Correct in DB
amount NUMERIC(19,4)

// Wrong
val amount: Double  // loses precision
val amount: Float   // loses precision
```

All amounts from bank APIs must be parsed to `BigDecimal` before storage. Amount equality comparisons use exact decimal matching.

Rule 6 — CSRF on Consent

Every OAuth consent flow must include a cryptographically random `state` parameter.

```
// Correct
val state = java.security.SecureRandom()
    .generateSeed(32)
    .toHexString()
// Store in server-side session (NOT cookie, NOT localStorage)
// Validate on callback — reject if mismatch
// One-time use — invalidate after successful exchange

// Wrong
val state = UUID.randomUUID().toString()  // too predictable
val state = "fixed-string"               // completely insecure
```

Rule 7 — 90-Day Consent Tracking

Every `BankConnection` must have automated expiry monitoring.

PSD2 (EBA RTS Art. 10) requires re-authentication every 90 days.

Mandatory implementation:

- `consentValidUntil` field on every `BankConnection`
- Daily cron: check all active connections
- 14 days before expiry: send email to org admin
- On expiry: set `consentStatus = 'expired'`, pause sync jobs
- UI: show "Bank feed paused — click to reconnect"
- One-click re-connect flow (user re-does SCA, new tokens stored)

Without this, bank feed silently breaks for ALL users when consents expire simultaneously.

Summary

Rule	Enforcement
1. Consent immutability	Code review — no UPDATE/DELETE on consent tables
2. Token encryption	No raw token strings in code/DB/logs
3. Bank adapter pattern	No direct HTTP bank calls outside adapter layer
4. Deduplication	DB unique constraint enforced
5. Money = NUMERIC(19,4)	No float/double for amounts anywhere
6. CSRF on consent	State parameter required in every consent initiation
7. 90-day tracking	Daily cron + email notification mandatory

Revision #3
Created 2026-03-04 05:07:44 UTC by John
Updated 2026-05-31 20:04:41 UTC by John