Bank Adapter Pattern

Bank Adapter Pattern

The bank adapter pattern is a core architectural principle in Tok. Every bank integration goes through an abstract `BankAdapter` interface, isolating the rest of the system from bank-specific API differences.

Why This Pattern?

Banks across Croatia, Serbia, and BiH use different API standards:

- **Croatia:** Berlin Group NextGenPSD2 (standardised, all HUB-registered banks)
- **Serbia (EU groups):** Berlin Group (UniCredit, Raiffeisen, NLB)
- **Serbia (domestic):** No central standard — bilateral per bank (AIK, OTP Serbia, Banca Intesa)
- **BiH:** Bilateral agreements only (no PSD2 mandate)

The adapter pattern hides this complexity behind one interface.

Abstract Interface

```
interface BankAdapter {
    /** Initiate OAuth consent flow — returns redirect URL */
    suspend fun initiateConsent(
        organizationId: String,
        callbackUrl: String
    ): ConsentRequest

    /** Exchange auth code for access + refresh tokens */
    suspend fun exchangeCode(code: String, state: String): OAuthTokens

    /** Refresh access token using refresh token */
```

```
    suspend fun refreshToken(refreshToken: String): OAuthTokens

    /** Fetch transactions for a date range */
    suspend fun fetchTransactions(
        accessToken: String,
        accountId: String,
        fromDate: LocalDate,
        toDate: LocalDate
    ): List<BankTransaction>

    /** Fetch account balance */
    suspend fun fetchBalance(
        accessToken: String,
        accountId: String
    ): BigDecimal

    /** Revoke consent */
    suspend fun revokeConsent(accessToken: String, consentId: String)
}
```

Implementations

BerlinGroupAdapter

Implements the **Berlin Group NextGenPSD2** standard.

Used for:

- All Croatian banks (registered with HUB — min. v1.3.8)
- EU bank groups in Serbia: UniCredit, Raiffeisen, NLB

Standard endpoints:

```
Consent URL:  {baseUrl}/v1/consents
Auth URL:     {baseUrl}/v1/oauth/authorize
Token URL:    {baseUrl}/v1/oauth/token
Accounts:     GET {baseUrl}/v1/accounts
Transactions: GET {baseUrl}/v1/accounts/{accountId}/transactions
              ?dateFrom={ISO}&dateTo={ISO}
```

Required headers:

```
Authorization: Bearer {access_token}
X-Request-ID:  {uuid}          ← unique per request
PSU-IP-Address: {user-ip}      ← required by some banks
```

Auth flow: OAuth 2.0 Authorization Code Grant + SCA redirect.

BilateralAdapter

Implements **per-bank custom REST** integrations for banks without a central standard.

Used for:

- Domestic Serbian banks (AIK, OTP Serbia, Banca Intesa Serbia)
- BiH banks under bilateral agreements
- Any bank that does not adopt Berlin Group

Each bilateral bank gets its own `BilateralAdapter` subclass with custom field mapping — the interface contract remains identical.

Bank Registry

```
val BANK_REGISTRY: Map<String, BankAdapterConfig> = mapOf(
    // Croatia (Berlin Group)
    "addiko-hr"    to BankAdapterConfig(
        adapter = "BerlinGroup",
        baseUrl = "https://oapideveloper.addiko.hr"
    ),
    "erste-hr"     to BankAdapterConfig(
        adapter = "BerlinGroup",
        baseUrl = "https://developers.erstegroup.com"
    ),
    "hpb-hr"       to BankAdapterConfig(
        adapter = "BerlinGroup",
        baseUrl = "https://openbanking.hpb.hr"
    ),
    "otp-hr"       to BankAdapterConfig(
        adapter = "BerlinGroup",
```

```
        baseUrl = "https://api.otpbanka.hr"
    ),
    "pbz-hr"       to BankAdapterConfig(
        adapter = "BerlinGroup",
        baseUrl = "https://apiportal.pbz.hr"
    ),
    "raiffeisen-hr" to BankAdapterConfig(
        adapter = "BerlinGroup",
        baseUrl = "https://sandbox.rba.hr"
    ),
    "zaba-hr"      to BankAdapterConfig(
        adapter = "BerlinGroup",
        baseUrl = "https://developer.unicredit.eu"
    ),
    // Serbia — EU groups (Berlin Group)
    "nlb-rs"       to BankAdapterConfig(
        adapter = "BerlinGroup",
        baseUrl = "https://developer.nlbkb.rs"
    ),
    "unicredit-rs" to BankAdapterConfig(
        adapter = "BerlinGroup",
        baseUrl = "https://developer.unicredit.eu"
    ),
    "raiffeisen-rs" to BankAdapterConfig(
        adapter = "BerlinGroup",
        baseUrl = "https://api.rbinternational.com"
    ),
    // Domestic Serbian banks — added as bilateral agreements are established
)
```

Transaction Normalization

Regardless of which adapter is used, all transactions are normalized to the internal `BankTransaction` format before being stored. This is the adapter's primary responsibility.

Internal format fields:

Field	Type	Source

`externalId`	String	Bank's own transaction ID (dedup key)
`bookingDate`	LocalDate	Berlin Group `bookingDate`
`valueDate`	LocalDate	Berlin Group `valueDate`
`amount`	NUMERIC(19,4)	Normalized — never float
`currency`	CurrencyCode	ISO 4217
`direction`	inbound/outbound	Derived from credit/debit indicator
`creditorIban`	String?	Berlin Group `creditorAccount.iban`
`debtorIban`	String?	Berlin Group `debtorAccount.iban`
`remittanceInfo`	String?	`remittanceInformationUnstructured`
`source`	String	`open_banking` (vs `manual` or `csv_import`)

Adding a New Bank

1. Determine which adapter applies (Berlin Group or bilateral)
2. Add entry to `BANK_REGISTRY` with `baseUrl`, `authUrl`, `scopes`
3. If bilateral — implement custom `fetchTransactions()` field mapping
4. Register sandbox credentials and test against bank sandbox
5. Add bank to UI: logo, display name, supported countries
6. Test deduplication against `externalId + bankAccountId` unique constraint

Revision #3
Created 2026-03-04 05:07:44 UTC by John
Updated 2026-05-31 20:04:40 UTC by John