# Payment Processing

# Payment Processing Architecture

**Version:** 1.0 **Date:** 2026-02-21 **Author:** Banking Architecture Team **Status:** Approved **Applies to:** Drop — Payment Initiation & Settlement

---

# 1. Overview

Drop processes two types of payments, both initiated via **PISP** (Payment Initiation Service Provider) from the user's own bank account:

1. **Remittance** — cross-border money transfers to 30+ countries (via SEPA SCT/SCTInst for EEA, SWIFT gpi for non-EEA)
2. **QR Payment** — instant domestic payments to merchants in Norway

Drop **never holds customer funds**. All payments are initiated directly from the user's bank account via PSD2 Open Banking APIs. Drop earns revenue from transaction fees (0.5% remittance, 1% QR).

| Property | Remittance | QR Payment |
|---|---|---|
| API endpoint | `POST /api/transactions/remittance` | `POST /api/transactions/qr-payment` |
| Fee | 0.5% of send amount | 1.0% of payment amount |
| Amount range | 100 - 50,000 NOK | 1 - 100,000 NOK |
| Settlement rail | SEPA SCT/SCTInst (EEA), SWIFT (non-EEA) | Domestic credit transfer (instant) |
| KYC required | Yes (`kyc_status = 'approved'`) | No (auth sufficient) |
| FX conversion | Yes (NOK to recipient currency) | No (NOK to NOK) |
| Status flow | `processing` -> `completed` / `failed` | `completed` (instant) |

---

# 2. SEPA Payment Flow (EEA Remittance)

For remittances to EEA countries (EU + Norway, Iceland, Liechtenstein), Drop uses **SEPA Credit Transfer (SCT)** or **SEPA Instant Credit Transfer (SCT Inst)**.

## 2.1 SEPA SCT Flow

```
sequenceDiagram
    participant U as User
    participant D as Drop API
    participant DB as Drop DB
    participant A as User's Bank (ASPSP)
    participant CSM as SEPA CSM<br/>(EBA CLEARING / TARGET2)
    participant RB as Recipient Bank

    U->>D: POST /api/transactions/remittance<br/>{recipientId, amount: 2000, bankAccountId}
    D->>DB: Verify: KYC approved, recipient exists,<br/>bank account exists
    D->>DB: GET exchange_rates WHERE to_currency = 'EUR'
    D->>D: Calculate fee: 2000 * 0.005 = 10 NOK<br/>Calculate receive: 2000 * 0.087 = 174 EUR<br/>Total debit: 2010 NOK

    D->>D: POST /api/transactions/disclosure<br/>(PSD2 Art. 45/46 pre-payment info)
    D-->>U: Disclosure: send 2000 NOK, fee 10 NOK,<br/>rate 0.087, receive 174 EUR,<br/>ETA 1-2 business days

    U->>D: Confirm payment
    D->>DB: Generate idempotency_key<br/>INSERT transaction (status: processing)
    D->>A: POST /v1/payments/sepa-credit-transfers<br/>{debtorAccount: {iban: user_iban},<br/>instructedAmount: {currency: NOK, amount: 2010},<br/>creditorAccount: {iban: recipient_iban},<br/>creditorName: "Recipient Name"}
    A-->>D: {paymentId, transactionStatus: RCVD,<br/>scaRedirect: "https://bank.no/sca/..."}
    D-->>U: Redirect to bank SCA

    U->>A: BankID authentication (dynamic linking:<br/>amount 2010 NOK to "Recipient Name")
    A-->>U: Redirect to Drop callback
    U->>D: Payment callback
    D->>A: GET /v1/payments/{paymentId}/status
```

```
    A-->>D: {transactionStatus: ACCP}
    D->>DB: UPDATE transaction SET status = 'completed'

    Note over A,CSM: Bank submits to SEPA CSM<br/>within cutoff time
    A->>CSM: SEPA SCT message (pacs.008)
    CSM->>RB: Route to recipient bank
    RB->>RB: Credit recipient account
    Note over CSM,RB: Settlement: T+1 business day<br/>(SCT Inst: < 10 seconds)
```

## 2.2 SEPA Specifications

| Property | SEPA SCT | SEPA SCT Inst |
|---|---|---|
| Standard | ISO 20022 pacs.008 | ISO 20022 pacs.008 |
| Max amount | 999,999,999.99 EUR | 100,000 EUR |
| Settlement time | T+1 business day | < 10 seconds (24/7/365) |
| Availability | Business days only | 24/7/365 |
| Coverage | 36 SEPA countries | Participating banks only |
| Drop usage | Default for EEA remittance | Preferred when available |
| Cut-off time | Bank-specific (typically 14:00-16:00 CET) | No cut-off |

---

# 3. Cross-Border Remittance with FX

For non-EEA corridors (Serbia, Pakistan, Turkey, etc.), Drop uses **SWIFT gpi** (Global Payments Innovation) or correspondent banking networks.

## 3.1 Cross-Border Flow with FX Conversion

```
sequenceDiagram
    participant U as User
    participant D as Drop API
    participant FX as FX Rate Provider
    participant A as User's Bank (ASPSP)
    participant CB as Correspondent Bank
    participant RB as Recipient Bank<br/>(e.g., Banca Intesa, Serbia)
```

```
    U->>D: POST /api/transactions/remittance<br/>{recipientId: rec_1, amount: 2000}
    D->>D: Lookup recipient: Marko Petrovic,<br/>Serbia, RSD, Banca Intesa

    D->>FX: GET current NOK/RSD rate
    FX-->>D: Rate: 10.17 (1 NOK = 10.17 RSD)

    D->>D: Calculate:<br/>Send: 2000 NOK<br/>Fee: 2000 * 0.005 = 10 NOK<br/>Receive: 2000 *
10.17 = 20,340 RSD<br/>Total debit: 2010 NOK

    D-->>U: Disclosure (PSD2 Art. 45):<br/>You send: 2,000 NOK<br/>Fee: 10 NOK
(0.5%)<br/>Rate: 1 NOK = 10.17 RSD<br/>Recipient receives: 20,340 RSD<br/>Total cost: 2,010
NOK<br/>ETA: 2-4 business days

    U->>D: Confirm payment
    D->>D: Lock FX rate for 15 minutes<br/>(rate_locked_at = now, rate_expires_at = now + 15m)
    D->>D: Generate idempotency_key<br/>INSERT transaction (status: processing)

    D->>A: POST /v1/payments/cross-border-credit-transfers<br/>{debtorAccount: {iban},
instructedAmount: {NOK, 2010},<br/>creditorAccount: {bban:
recipient_bank_account},<br/>creditorName: "Marko Petrovic",<br/>creditorAgent: {bic:
DBDBRSBG}}
    A-->>D: {paymentId, scaRedirect}
    D-->>U: Redirect to bank SCA

    U->>A: BankID authentication
    A-->>U: Redirect to Drop callback
    D->>A: GET /v1/payments/{paymentId}/status
    A-->>D: {transactionStatus: ACCP}
    D->>D: UPDATE transaction status = 'completed'

    Note over A,CB: SWIFT gpi transfer<br/>UETR tracking ID assigned
    A->>CB: MT103 / pacs.008 (NOK)
    CB->>CB: FX conversion NOK to RSD<br/>(at correspondent bank rate)
    CB->>RB: Credit in RSD
    RB->>RB: Credit Marko's account<br/>20,340 RSD received
```

# 3.2 Supported Corridors

| Corridor | Currency | Exchange Rate (NOK to) | Rail | Estimated Delivery |
|---|---|---|---|---|
| Norway to Serbia | RSD | 10.17 | SWIFT gpi | 2-4 business days |
| Norway to Bosnia | BAM | 0.17 | SWIFT gpi | 2-4 business days |
| Norway to Poland | PLN | 0.374 | SEPA SCT (EEA) | 1-2 business days |
| Norway to Pakistan | PKR | 26.5 | SWIFT gpi | 2-4 business days |
| Norway to Turkey | TRY | 3.39 | SWIFT gpi | 2-4 business days |
| Norway to EU (EUR) | EUR | 0.087 | SEPA SCT/SCTInst | 1-2 days / instant |

**Source:** `exchange_rates` table, seeded in `db.ts:234-237`

---

# 4. FX Rate Management

## 4.1 Rate Sourcing

| Phase | Source | Refresh | Markup |
|---|---|---|---|
| MVP (current) | Static seed data in `exchange_rates` table | Manual update | None (display rate = mid-market) |
| Phase 2 | ECB reference rates + commercial FX provider | Every 15 minutes | 0.1-0.3% spread |
| Phase 3 | Real-time feed from FX partner (e.g., Wise, CurrencyCloud) | Real-time (streaming) | Configurable per corridor |

## 4.2 Rate Lock Window

When a user initiates a remittance, the FX rate is **locked for 15 minutes**:

1. User sees rate on the disclosure screen
2. Rate is locked when user confirms (before SCA)
3. If SCA completes within 15 minutes, the locked rate applies
4. If SCA times out, the rate expires and must be re-quoted

This protects both the user (no surprise rate changes during authentication) and Drop (limited exposure to rate movement).

## 4.3 Rate Storage

| Column | Table | Description |
|---|---|---|
| `exchange_rates.rate` | `exchange_rates` | Current mid-market rate (NOK to target) |
| `exchange_rates.updated_at` | `exchange_rates` | Last rate update timestamp |
| `transactions.exchange_rate` | `transactions` | Rate locked at transaction time |
| `transactions.send_amount` | `transactions` | Amount in NOK (stored in oere) |
| `transactions.receive_amount` | `transactions` | Amount in target currency (stored in subunits) |

---

# 5. Fee Calculation Model

## 5.1 Fee Structure

| Transaction Type | Fee Rate | Min Fee | Max Fee | Applied To |
|---|---|---|---|---|
| Remittance | 0.5% | 10 NOK | 500 NOK | Send amount (before FX) |
| QR Payment | 1.0% | 1 NOK | 1,000 NOK | Payment amount |
| AISP balance read | Free | - | - | No charge |

## 5.2 Fee Calculation

**Remittance example (2,000 NOK to Serbia):**

| Line Item | Calculation | Amount |
|---|---|---|
| Send amount | User input | 2,000.00 NOK |
| Fee (0.5%) | 2,000 * 0.005 | 10.00 NOK |
| Total debit | Send + Fee | 2,010.00 NOK |
| Exchange rate | From `exchange_rates` table | 10.17 RSD/NOK |
| Receive amount | 2,000 * 10.17 | 20,340.00 RSD |

**QR Payment example (149 NOK at merchant):**

| Line Item | Calculation | Amount |
|---|---|---|
| Payment amount | From QR scan | 149.00 NOK |
| Fee (1.0%) | 149 * 0.01 | 1.49 NOK |

| Line Item | Calculation | Amount |
|---|---|---|
| Total debit | Payment + Fee | 150.49 NOK |
| Merchant receives | Payment - merchant fee | 147.51 NOK |

## 5.3 Fee Code References

| Endpoint | Fee Logic | Source |
|---|---|---|
| `POST /api/transactions/remittance` | `fee = amount * 0.005` | `transactions/remittance/route.ts` |
| `POST /api/transactions/qr-payment` | `fee = amount * 0.01` | `transactions/qr-payment/route.ts` |
| `POST /api/transactions/disclosure` | Returns fee + FX pre-payment | `transactions/disclosure/route.ts` |
| `GET /api/rates/[currency]` | Returns `fee: 0.005` (informational) | `rates/[currency]/route.ts` |

## 5.4 Revenue Model Comparison

| Provider | Remittance Fee | QR/In-Store Fee |
|---|---|---|
| **Drop** | **0.5%** | **1.0%** |
| Western Union | 5-10% | N/A |
| Wise | 0.7-1.5% | N/A |
| Vipps | N/A | 1.75-2.75% |

---

# 6. Settlement & Reconciliation

## 6.1 Settlement Flow

Drop does not settle payments itself — the ASPSP (user's bank) handles settlement via interbank rails. Drop's role is to **initiate** and **track** payments.

| Step | Actor | Action |
|---|---|---|
| 1. Initiation | Drop | POST PISP request to ASPSP |
| 2. SCA | User + ASPSP | User authenticates at bank |
| 3. Acceptance | ASPSP | Bank accepts payment instruction |
| 4. Clearing | CSM (SEPA) / SWIFT | Message routed to recipient bank |
| 5. Settlement | Central bank / CSM | Funds transferred between banks |

| Step | Actor | Action |
|---|---|---|
| 6. Credit | Recipient bank | Recipient account credited |
| 7. Confirmation | Drop | Poll payment status, update transaction |

# 6.2 Reconciliation Process

```
flowchart TD
    A[Scheduled reconciliation job<br/>runs every hour] --> B{Fetch transactions<br/>WHERE status = 'processing'<br/>AND created_at older than 1h}
    B --> C[For each pending transaction]
    C --> D[GET /v1/payments/paymentId/status<br/>from ASPSP]
    D --> E{ASPSP status?}

    E -->|ACSC / ACCP| F[UPDATE status = 'completed'<br/>SET completed_at = now]
    E -->|RJCT| G[UPDATE status = 'failed'<br/>Log rejection reason]
    E -->|PDNG / ACTC| H[Keep as 'processing'<br/>Check again next cycle]
    E -->|API error| I[Log error, retry next cycle<br/>Circuit breaker if repeated]

    F --> J[Create notification:<br/>'Overfoering fullfoert']
    G --> K[Create notification:<br/>'Overfoering feilet'<br/>+ refund logic]

    K --> L{Funds already debited?}
    L -->|Yes| M[Initiate refund via ASPSP<br/>or manual intervention]
    L -->|No| N[No action needed<br/>Payment was never executed]
```

# 6.3 ASPSP Transaction Statuses (Berlin Group)

| Status Code | Meaning | Drop Action |
|---|---|---|
| `RCVD` | Received (payment accepted for processing) | Transaction created, status = `processing` |
| `PDNG` | Pending (awaiting SCA or bank processing) | Keep as `processing` |
| `ACTC` | Accepted Technical (technical validation passed) | Keep as `processing` |
| `ACCP` | Accepted Customer Profile (customer checks passed) | Keep as `processing` |
| `ACSC` | Accepted Settlement Completed | Update to `completed` |

| Status Code | Meaning | Drop Action |
|---|---|---|
| `ACSP` | Accepted Settlement In Process | Keep as `processing` |
| `RJCT` | Rejected | Update to `failed` |
| `CANC` | Cancelled | Update to `failed` |

---

# 7. Idempotency & Retry Strategy

## 7.1 Idempotency

The `transactions` table has a unique index on `idempotency_key` (`idx_tx_idempotency`):

```
CREATE UNIQUE INDEX IF NOT EXISTS idx_tx_idempotency
  ON transactions(idempotency_key)
  WHERE idempotency_key IS NOT NULL;
```

**Key generation:** `{userId}:{recipientId|merchantId}:{amount}:{timestamp_minute}`

**Flow:**

1. Before creating a transaction, check if `idempotency_key` already exists
2. If exists, return the existing transaction (no duplicate)
3. If not, insert new transaction with the key
4. Pass the same key as `X-Request-ID` to the ASPSP

## 7.2 Retry Strategy

| Failure Type | Retry? | Strategy |
|---|---|---|
| Network timeout to ASPSP | Yes | Exponential backoff: 1s, 2s, 4s (max 3 retries) |
| ASPSP returns 5xx | Yes | Exponential backoff with jitter, max 3 retries |
| ASPSP returns 4xx | No | Log error, fail immediately (client error) |
| SCA timeout | No | Mark as failed, user must restart |
| Duplicate detected | No | Return existing transaction |
| FX rate expired | No | Re-quote rate, user must re-confirm |

---

# 8. Pre-Payment Disclosure (PSD2 Art. 45/46)

Before initiating any payment, Drop must provide the user with clear information about costs and delivery. The `POST /api/transactions/disclosure` endpoint generates this.

## 8.1 Disclosure Content

| Information Item | PSD2 Article | Drop Implementation |
|---|---|---|
| Total amount debited | Art. 45(1)(a) | `totalCost` = amount + fee |
| Fee amount and percentage | Art. 45(1)(b) | `fee`, `feePercentage` |
| Exchange rate applied | Art. 45(1)(c) | `exchangeRate` |
| Amount received by recipient | Art. 45(1)(d) | `receiveAmount` in `receiveCurrency` |
| Estimated delivery time | Art. 45(1)(e) | `estimatedDelivery` |
| Currency of debit | Art. 45(1)(f) | Send currency (NOK) |
| Currency of credit | Art. 45(1)(g) | Receive currency |

## 8.2 Disclosure API Response

```
{
  "amount": 2000,
  "fee": 10,
  "feePercentage": 0.5,
  "exchangeRate": 10.17,
  "receiveAmount": 20340,
  "receiveCurrency": "RSD",
  "estimatedDelivery": "2-4 business days",
  "totalCost": 2010
}
```

## 8.3 Delivery Time Estimates

| Transaction Type | Corridor | Estimate |
|---|---|---|
| QR Payment | Domestic (Norway) | "Instant" |
| Remittance | EEA (SEPA) | "1-2 business days" |

| Transaction Type | Corridor | Estimate |
|---|---|---|
| Remittance | Non-EEA (SWIFT) | "2-4 business days" |

---

# 9. Transaction Integrity

## 9.1 Atomic Operations

All financial operations use database transactions to ensure atomicity. The `transaction()` function in `db.ts:123-179` wraps operations in `BEGIN`/`COMMIT` blocks with automatic `ROLLBACK` on error.

Key integrity checks:

- `WHERE balance >= ?` prevents overdraw
- PostgreSQL MVCC + `READ COMMITTED` isolation (default) prevents dirty reads; use `SERIALIZABLE` for phantom read protection
- Fee calculated and included in the single atomic debit

## 9.2 Consistency Guarantees

| Guarantee | Mechanism |
|---|---|
| No double-spend | `WHERE balance >= ?` in UPDATE + idempotency_key |
| No partial transactions | PostgreSQL `BEGIN`/`COMMIT` |
| No phantom reads | PostgreSQL MVCC snapshot isolation; use SERIALIZABLE isolation for full phantom read protection |
| No duplicate payments | Unique index on `idempotency_key` |
| No stale balances | `balance_synced_at` tracking + pre-payment AISP refresh |

---

# 10. Monitoring & Alerts

## 10.1 Key Metrics

| Metric | Threshold | Alert |
|---|---|---|
| Transaction success rate | < 95% over 1 hour | Critical alert |
| Average settlement time (SEPA) | > 48 hours | Warning |

| Metric | Threshold | Alert |
|---|---|---|
| Average settlement time (SWIFT) | > 5 business days | Warning |
| Failed transaction rate | > 5% | Warning |
| Reconciliation mismatches | Any | Immediate alert |
| FX rate staleness | > 1 hour since last update | Warning |

## 10.2 Audit Trail

All payment operations are logged in the `audit_log` table:

| Action | Logged Data |
|---|---|
| `payment.initiated` | Transaction ID, amount, recipient, bank account |
| `payment.sca_completed` | Transaction ID, SCA method |
| `payment.completed` | Transaction ID, ASPSP status, settlement reference |
| `payment.failed` | Transaction ID, failure reason, ASPSP error |
| `payment.refund` | Original transaction ID, refund amount |

---

# 11. Cross-References

- **Open Banking AISP/PISP:** open-banking-aisp-pisp.md — Berlin Group API integration, consent lifecycle
- **BankID OIDC:** bankid-oidc-integration.md — Authentication (not payment SCA)
- **Security Architecture:** ../hld/security-architecture.md — Fraud detection, AML screening
- **Remittance Flow (LLD):** ../lld/flow-remittance.md — Step-by-step remittance UX
- **Database Schema:** ../../backend/DATABASE-SCHEMA.md — `transactions`, `exchange_rates`, `bank_accounts` tables
- **API Reference:** ../../backend/API-REFERENCE.md — Transaction, disclosure, and rate endpoints
- **Compliance:** ../../security/COMPLIANCE.md — PSD2, AML readiness

---

Revision #5
Created 2026-02-21 05:59:04 UTC by John
Updated 2026-05-23 10:57:04 UTC by John