drop-fx-transparency-spec

Drop FX Rate Transparency & Fee Breakdown Specification

Task: MC #1193 **Created:** 2026-02-17 **Author:** John (Architect Agent) **Status:** DRAFT — Awaiting Alem approval **Product:** Drop — Fintech Payment App (Pass-through PSD2 PISP model)

Executive Summary

This specification defines the architecture for **real-time exchange rate transparency and fee breakdown** for Drop's remittance payment service. Drop operates as a PSD2 PISP (Payment Initiation Service Provider) — we initiate payments from users' bank accounts but never hold customer money.

Regulatory Requirement: PSD2 Directive 2015/2366/EU Article 45 mandates that payment service providers disclose:

1. All charges payable by the payment service user with a breakdown
2. The actual or reference exchange rate to be applied to the payment transaction
3. Maximum execution time for the payment service

Current State: Drop uses mocked/seeded exchange rates from database (`exchange_rates` table). Rates refresh hourly from `EXCHANGE_RATE_API_URL` (env var, not set). Fee calculation is hardcoded at 0.5% for remittances.

Goal: Live FX rates (Norges Bank official data), transparent fee breakdown shown BEFORE user confirms transfer, full PSD2 compliance.

---

1. Business Context

1.1 Norwegian Context

- **Base Currency:** NOK (Norwegian Krone)
- **Target Users:** ALL residents in Norway (NOT limited to diaspora)
- **Corridors:** 30+ countries (seeded in `recipients` table — RSD, EUR, USD, GBP, SEK, DKK, PLN, etc.)
- **Regulatory Body:** Finanstilsynet (Norwegian Financial Supervisory Authority)
- **Reference Rate:** Norges Bank publishes daily exchange rates at ~16:00 CET for 40+ currencies

1.2 Competitive Landscape

Wise (market leader):

- Uses mid-market rate with 0.5-1% transparent fee
- Full breakdown shown BEFORE transfer: amount + fee + FX rate + total + "You send X, they get Y"
- Calculator on homepage (no login required)

Remitly:

- Variable FX markup 0.5-3% (not transparent, hidden in rate)
- Fee varies by speed (Express vs Economy), payment method, new vs returning user
- Less transparent than Wise — users complain about hidden fees

Drop's Positioning:

- **Transparent like Wise** — show mid-market reference rate + markup + fee separately
- **Norwegian-first** — NOK base, Norges Bank as reference
- **Cheaper than traditional banks** (DNB, Nordea charge 2-5%)

1.3 Pass-Through Model Implications

Drop does NOT:

- Hold customer money
- Perform FX conversion itself (that's the bank's job via PISP)
- Need forex license (we're PISP, not EMI)

Drop DOES:

- Show user the expected FX rate BEFORE transfer
- Charge a transparent service fee (0.5% for remittance)
- Display what recipient will receive (calculated estimate)
- Send payment instruction to user's bank via Open Banking

Key Point: FX rate shown to user is **reference/estimate**. Actual conversion happens at user's bank. Drop must disclose this clearly (PSD2 Article 45).

---

2. Current Implementation Analysis

2.1 What Exists (Good Foundation)

Database Schema:

```
-- exchange_rates table (seeded with 30+ currencies)
CREATE TABLE exchange_rates (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  from_currency TEXT NOT NULL,
  to_currency TEXT NOT NULL,
  rate REAL NOT NULL,
  updated_at TEXT NOT NULL DEFAULT (datetime('now')),
  UNIQUE(from_currency, to_currency)
);
```

Service Layer: `/src/lib/services/rates.ts`

- `getExchangeRates()` — returns all rates from DB
- `getRate(from, to)` — returns single rate
- `refreshRatesIfStale()` — fetches from `EXCHANGE_RATE_API_URL` if last update > 1 hour
- Graceful degradation — falls back to cached rates on API failure

API Endpoints:

- `GET /api/rates` — returns all rates
- `GET /api/rates/[currency]` — returns single rate
- `POST /api/transactions/disclosure` — calculates fee + FX preview (BEFORE user confirms)

UI Component: `/src/components/pre-payment-disclosure.tsx`

- Shows breakdown: amount + fee + FX rate + receive amount + total + ETA
- Modal dialog — user must confirm before transaction submits
- Norwegian language, good UX (matches Wise pattern)

Fee Calculation: Hardcoded in `/src/app/api/transactions/remittance/route.ts`

```
const feePercent = 0.005; // 0.5%
const fee = Math.round(amount * feePercent * 100) / 100;
```

2.2 What's Missing (Critical Gaps)

 Live FX Rate Source: `EXCHANGE_RATE_API_URL` not configured, no real provider **Norges Bank Integration:** Not using official Norwegian reference rates **FX Markup Transparency:** No distinction between mid-market rate and Drop's rate **Fee Configuration:** Fees hardcoded, no corridor-specific fees, no volume tiers **Rate Locking:** No guarantee user gets the rate they saw (can change between preview and execution) **Stale Rate Detection:** 1-hour threshold too long for high-volume corridors (EUR, USD) **Rate Drift Monitoring:** No alerts when rates deviate significantly from Norges Bank reference **Historical Rate Tracking:** No log of which rate was used for completed transactions **PSD2 Disclosure Language:** Component exists but needs exact regulatory wording

---

3. Data Sources

3.1 Norges Bank API (Primary Reference)

Official Source: Norges Bank Data Warehouse

API Base URL: `https://data.norges-bank.no/api/data/EXR/`

Example Request:

```
# Daily exchange rates for EUR, USD, GBP against NOK
# B = Business day frequency, SP = Spot (daily reference rate)
curl "https://data.norges-bank.no/api/data/EXR/B..NOK.SP?startPeriod=2026-02-
17&endPeriod=2026-02-17&format=sdmx-json&locale=en"
```

Key Features:

- **Free:** No API key required, fully open
- **Authoritative:** Official central bank rates used by Norwegian businesses
- **Daily Updates:** Published ~16:00 CET
- **40+ Currencies:** Covers all major Drop corridors
- **SDMX Standard:** Industry-standard format for statistical data

Limitations:

- **Daily Frequency:** Not real-time (updated once per business day)
- **No Weekends:** Last Friday rate used on weekends
- **Spot Rate Only:** No forward rates, no historical intraday

Cost: $0 (completely free)

3.2 Commercial FX Provider (Real-Time Fallback)

For high-volume corridors (EUR, USD, GBP) where users expect near-real-time rates:

Option A: ExchangeRate-API.com

- **Free Tier:** 1,500 requests/month, hourly updates
- **Paid Tier:** $9/month for 100k requests, 10-min updates
- **Currencies:** 160+
- **Format:** Simple JSON: `{ "NOK": { "EUR": 0.0867 } }`
- **Reliability:** 99.9% uptime SLA (paid tier)

Recommendation: Start with **Norges Bank (free) + ExchangeRate-API.com free tier** (1,500 requests/month = 50/day, enough for MVP). Migrate to paid tier when transaction volume > 50/day.

Hybrid Strategy:

1. **Norges Bank:** Daily reference rate for all corridors (free, authoritative)
2. **ExchangeRate-API:** Real-time updates for EUR, USD, GBP (high volume)
3. **Fallback:** Cached DB rates (stale threshold: 4 hours for major corridors, 24 hours for minor)

Cost Estimate: $0-9/month (depending on traffic)

---

4. Database Schema

4.1 Enhanced `exchange_rates` Table

```
-- Add columns to existing table
ALTER TABLE exchange_rates ADD COLUMN source TEXT DEFAULT 'seed';
  -- 'norges_bank', 'exchangerate_api', 'fixer', 'seed'
```

```
ALTER TABLE exchange_rates ADD COLUMN rate_type TEXT DEFAULT 'spot';
  -- 'spot', 'mid_market', 'buy', 'sell'

ALTER TABLE exchange_rates ADD COLUMN markup REAL DEFAULT 0.0;
  -- Drop's markup percentage (0.0 = no markup, 0.005 = 0.5%)

ALTER TABLE exchange_rates ADD COLUMN external_id TEXT;
  -- Provider's ID for this rate (if applicable)

ALTER TABLE exchange_rates ADD COLUMN is_stale INTEGER DEFAULT 0;
  -- 0 = fresh, 1 = stale (triggers refresh)

ALTER TABLE exchange_rates ADD COLUMN last_refresh_attempt TEXT;
  -- Timestamp of last API fetch attempt (for monitoring)

CREATE INDEX idx_rates_stale ON exchange_rates(is_stale, updated_at);
CREATE INDEX idx_rates_source ON exchange_rates(source);
```

4.2 New Table: `fx_rate_history`

Track which rate was shown to user AND which rate was actually used by bank:

```
CREATE TABLE fx_rate_history (
  id TEXT PRIMARY KEY,
  transaction_id TEXT REFERENCES transactions(id) ON DELETE CASCADE,
  from_currency TEXT NOT NULL,
  to_currency TEXT NOT NULL,
  shown_rate REAL NOT NULL,       -- Rate displayed to user at disclosure
  shown_markup REAL NOT NULL,     -- Markup at disclosure time
  shown_source TEXT NOT NULL,     -- 'norges_bank', 'exchangerate_api', etc.
  actual_rate REAL,               -- Actual bank rate (if known from bank statement)
  rate_locked_at TEXT,            -- Timestamp when rate was locked (if locking enabled)
  rate_locked_until TEXT,         -- Expiry time for locked rate
  created_at TEXT DEFAULT (datetime('now')),
  updated_at TEXT DEFAULT (datetime('now'))
);

CREATE INDEX idx_fx_history_tx ON fx_rate_history(transaction_id);
CREATE INDEX idx_fx_history_locked ON fx_rate_history(rate_locked_at);
```

Purpose:

- **Compliance:** PSD2 requires we log what rate we showed user
- **Transparency:** User can compare promised rate vs actual rate
- **Dispute Resolution:** Evidence for customer complaints ("you said 10.5, I got 10.3")
- **Rate Locking (Future):** Store locked rate + expiry

4.3 New Table: `fee_configs`

Make fees configurable instead of hardcoded:

```
CREATE TABLE fee_configs (
  id TEXT PRIMARY KEY,
  corridor TEXT NOT NULL UNIQUE, -- 'NOK-RSD', 'NOK-EUR', '*' (default)
  fee_type TEXT NOT NULL CHECK(fee_type IN ('percentage', 'flat', 'tiered')),
  fee_percentage REAL,           -- For percentage type (0.005 = 0.5%)
  fee_flat REAL,                 -- For flat type (25 NOK)
  fee_tiers TEXT,                -- JSON for tiered:
[{"max":1000,"rate":0.01},{"max":null,"rate":0.005}]
  min_fee REAL DEFAULT 0,        -- Minimum fee (NOK)
  max_fee REAL,                  -- Maximum fee cap (NOK, NULL = no cap)
  effective_from TEXT NOT NULL DEFAULT (datetime('now')),
  effective_until TEXT,          -- NULL = indefinite
  created_at TEXT DEFAULT (datetime('now')),
  updated_at TEXT DEFAULT (datetime('now'))
);

-- Seed default config
INSERT INTO fee_configs (id, corridor, fee_type, fee_percentage, min_fee)
VALUES ('fee_default', '*', 'percentage', 0.005, 10.0); -- 0.5%, min 10 NOK

-- Corridor-specific example (cheaper EUR corridor)
INSERT INTO fee_configs (id, corridor, fee_type, fee_percentage, min_fee)
VALUES ('fee_eur', 'NOK-EUR', 'percentage', 0.003, 5.0); -- 0.3%, min 5 NOK
```

Benefits:

- **A/B Testing:** Change fees without code deploy
- **Promotional Pricing:** Set temporary fee discounts (effective_from/until)
- **Corridor Optimization:** Cheaper fees for high-volume corridors (EUR, USD)
- **Tiered Pricing:** Lower fee percentage for large transfers (e.g., >10k NOK)

4.4 New Table: `fx_rate_alerts`

Monitor rate drift and API failures:

```
CREATE TABLE fx_rate_alerts (
  id TEXT PRIMARY KEY,
  alert_type TEXT NOT NULL CHECK(alert_type IN ('stale_rate', 'rate_drift', 'api_failure',
'missing_rate')),
  severity TEXT NOT NULL CHECK(severity IN ('low', 'medium', 'high', 'critical')),
  from_currency TEXT NOT NULL,
  to_currency TEXT,              -- NULL for API failure alerts
  details TEXT,                  -- JSON: {"expected":10.5,"actual":11.2,"drift_pct":6.7}
  resolved INTEGER DEFAULT 0,    -- 0 = open, 1 = resolved
  created_at TEXT DEFAULT (datetime('now')),
  resolved_at TEXT
);

CREATE INDEX idx_fx_alerts_unresolved ON fx_rate_alerts(alert_type, resolved, created_at);
```

Alert Examples:

- `stale_rate`: EUR/NOK not updated in 6 hours (should refresh hourly)
- `rate_drift`: Norges Bank shows 11.5, ExchangeRate-API shows 11.9 (3.5% drift — investigate)
- `api_failure`: ExchangeRate-API returned 503 for 3 consecutive attempts
- `missing_rate`: User tried to transfer to THB, no rate in DB

---

5. API Endpoints

5.1 Enhanced `GET /api/rates`

Current: Returns all rates from DB **New:** Add metadata, source attribution, staleness indicator

Request:

```
GET /api/rates?base=NOK&symbols=EUR,USD,RSD
```

Response:

```
{
  "base": "NOK",
  "rates": {
    "EUR": {
      "rate": 0.0867,
      "markup": 0.005,
      "effectiveRate": 0.0871,
      "source": "norges_bank",
      "updatedAt": "2026-02-17T16:00:00Z",
      "isStale": false
    },
    "USD": {
      "rate": 0.0923,
      "markup": 0.005,
      "effectiveRate": 0.0928,
      "source": "exchangerate_api",
      "updatedAt": "2026-02-17T14:30:00Z",
      "isStale": false
    }
  },
  "updatedAt": "2026-02-17T14:30:00Z"
}
```


Fields:

- `rate`: Mid-market reference rate (from Norges Bank or provider)
- `markup`: Drop's markup percentage
- `effectiveRate`: `rate * (1 + markup)` — what user actually pays
- `source`: Which API provided this rate
- `isStale`: True if rate hasn't refreshed within threshold

Caching: Cache-Control: public, max-age=300 (5 minutes)

5.2 Enhanced `POST /api/transactions/disclosure`

Current: Calculates fee + FX preview **New:** Return full PSD2-compliant disclosure, log to `fx_rate_history`

Request:

```
{
  "type": "remittance",
  "amount": 5000,
  "recipientId": "rec_abc123"
}
```

Response:

```
{
  "sendAmount": 5000,
  "sendCurrency": "NOK",
  "fee": 25,
  "feePercentage": 0.5,
  "totalCost": 5025,
  "exchangeRate": {
    "reference": 10.1700,
    "source": "Norges Bank (16:00 CET)",
    "markup": 0.5,
    "effectiveRate": 10.2209,
    "updatedAt": "2026-02-17T16:00:00Z"
  },
  "receiveAmount": 51104.50,
  "receiveCurrency": "RSD",
  "estimatedDelivery": "1-2 business days",
  "rateValidUntil": "2026-02-17T17:00:00Z",
  "psd2Disclosure": {
    "en": "This is an estimate. Your bank will apply its own exchange rate at the time of transfer. The final amount received may differ.",
    "no": "Dette er et estimat. Din bank vil bruke sin egen valutakurs ved overføring. Endelig beløp mottatt kan avvike."
  },
  "disclosureId": "disc_xyz789"
}
```

---

6. Rate Refresh Strategy

6.1 Refresh Schedule

Corridor	Source	Refresh Frequency	Stale Threshold	Fallback
NOK-EUR	ExchangeRate-API	10 min	1 hour	Norges Bank daily
NOK-USD	ExchangeRate-API	10 min	1 hour	Norges Bank daily
NOK-GBP	ExchangeRate-API	10 min	1 hour	Norges Bank daily
NOK-RSD	Norges Bank	Daily (16:00 CET)	24 hours	Cached DB
NOK-SEK	Norges Bank	Daily (16:00 CET)	24 hours	Cached DB
All Others	Norges Bank	Daily (16:00 CET)	48 hours	Cached DB

Rationale:

- **High-Volume Corridors (EUR, USD, GBP):** Users expect near-real-time rates. 10-min refresh acceptable.
- **Regional Corridors (RSD, SEK, PLN):** Daily Norges Bank rate sufficient (low volatility).
- **Long-Tail Corridors:** 48-hour stale threshold (rarely used, rate changes minimal).

6.2 Refresh Triggers

Automatic:

1. **Cron Job:** Every 10 min, check stale rates and refresh high-volume corridors
2. **API Request:** When user requests `/api/transactions/disclosure`, check if corridor rate is stale → refresh synchronously (max 5s timeout)
3. **Daily Batch:** At 16:30 CET (30 min after Norges Bank publishes), fetch all 40 currencies

Manual:

- Admin endpoint: `POST /api/admin/fx/refresh` (force refresh all rates)

---

7. Fee Calculation Engine

7.1 Fee Structure

Fee Types:

1. **Percentage:** `amount * fee_percentage` (most common)
2. **Flat:** Fixed amount (e.g., 25 NOK for all transfers)
3. **Tiered:** Different percentage based on amount brackets

Example Tiered Fee (NOK-RSD):

```
[
  { "max": 1000, "rate": 0.01 },   // 0-1000 NOK: 1%
  { "max": 5000, "rate": 0.007 },  // 1001-5000 NOK: 0.7%
  { "max": null, "rate": 0.005 }   // >5000 NOK: 0.5%
]
```

Implementation:

```
// /lib/services/fees.ts
export interface FeeConfig {
  corridor: string;
  feeType: 'percentage' | 'flat' | 'tiered';
  feePercentage?: number;
  feeFlat?: number;
  feeTiers?: { max: number | null; rate: number }[];
  minFee: number;
  maxFee?: number;
}

export async function calculateFee(
  amount: number,
  fromCurrency: string,
  toCurrency: string
): Promise<{ fee: number; config: FeeConfig }> {
  const corridor = `${fromCurrency}-${toCurrency}`;

  // 1. Get corridor-specific config (or default '*')
  let config = await getOne<FeeConfig>(
    `SELECT * FROM fee_configs
     WHERE corridor = ?
       AND (effective_from <= datetime('now'))
       AND (effective_until IS NULL OR effective_until > datetime('now'))
     ORDER BY effective_from DESC LIMIT 1`,
    [corridor]
  );

  if (!config) {
    config = await getOne<FeeConfig>(
      `SELECT * FROM fee_configs WHERE corridor = '*' LIMIT 1`,
      []
```

```
    );
  }

  if (!config) {
    throw new Error('No fee configuration found');
  }

  // 2. Calculate fee based on type
  let fee = 0;

  if (config.feeType === 'percentage') {
    fee = amount * (config.feePercentage || 0);
  } else if (config.feeType === 'flat') {
    fee = config.feeFlat || 0;
  } else if (config.feeType === 'tiered') {
    const tiers = JSON.parse(config.feeTiers || '[]');
    for (const tier of tiers) {
      if (tier.max === null || amount <= tier.max) {
        fee = amount * tier.rate;
        break;
      }
    }
  }

  // 3. Apply min/max caps
  if (config.minFee && fee < config.minFee) {
    fee = config.minFee;
  }
  if (config.maxFee && fee > config.maxFee) {
    fee = config.maxFee;
  }

  // 4. Round to 2 decimals
  fee = Math.round(fee * 100) / 100;

  return { fee, config };
}
```

8. UI Component Spec

8.1 Pre-Payment Disclosure (Enhanced)

Location: `/src/components/pre-payment-disclosure.tsx`

Required Changes:

```
interface PrePaymentDisclosureProps {
  amount: number;
  fee: number;
  feeConfig: FeeConfig; // NEW: show fee structure
  exchangeRate: {
    reference: number;
    source: string; // "Norges Bank (16:00 CET)"
    markup: number;
    effectiveRate: number;
    updatedAt: string;
  };
  receiveAmount: number;
  receiveCurrency: string;
  estimatedDelivery: string;
  psd2Disclosure: { no: string; en: string }; // NEW: regulatory text
  rateValidUntil: string; // NEW: countdown timer
  onConfirm: () => void;
  onCancel: () => void;
}
```

Layout Additions:

1. **Exchange Rate Breakdown Section:**

```
<div className="bg-[#F8FAFC] rounded-xl p-4 space-y-2">
  <div className="flex items-center justify-between">
    <span className="text-xs text-[#64748B]">Referansekurs (Norges Bank)</span>
    <span className="text-sm font-medium text-[#1E293B]">1 NOK = 10.1700 RSD</span>
  </div>
  <div className="flex items-center justify-between">
    <span className="text-xs text-[#64748B]">Drop's påslag (0.5%)</span>
    <span className="text-sm font-medium text-[#1E293B]">+ 0.0509 RSD</span>
```

```
  </div>
  <div className="pt-2 border-t border-[#E2E8F0] flex items-center justify-between">
    <span className="text-sm font-bold text-[#0F172A]">Effektiv kurs</span>
    <span className="text-sm font-bold text-[#0B6E35]">1 NOK = 10.2209 RSD</span>
  </div>
</div>
```

2. **PSD2 Regulatory Disclosure:**

```
<div className="p-3 bg-[#FFFBEB] border border-[#FCD34D] rounded-xl">
  <p className="text-xs text-[#92400E] leading-relaxed">
    <strong>Viktig informasjon:</strong> Dette er et estimat basert på dagens valutakurs.
    Din bank vil bruke sin egen valutakurs ved overføring. Endelig beløp mottatt kan avvike.
  </p>
</div>
```

---

9. PSD2 Compliance Checklist

9.1 Article 45: Information Before Payment Execution

Requirements (from PSD2 Directive):

Maximum execution time:

- Drop: "1-2 business days" (EEA) or "2-4 business days" (non-EEA)
- Shown in disclosure modal + transaction receipt

All charges payable with breakdown:

- Drop: "Gebyr (0.5%): 25 NOK" + "Total kostnad: 5025 NOK"
- Breakdown: base amount + fee = total

Actual or reference exchange rate:

- Drop: Shows Norges Bank reference rate + markup + effective rate
- Disclosure: "Referansekurs (Norges Bank): 1 NOK = 10.1700 RSD, Drop's påslag (0.5%): + 0.0509 RSD, Effektiv kurs: 1 NOK = 10.2209 RSD"

9.2 Disclosure Language (Norwegian + English)

PSD2-Compliant Text:

Norwegian (Primary):

```
VIKTIG INFORMASJON OM VALUTAVEKSLING

Dette er et estimat basert på dagens valutakurs fra Norges Bank (oppdatert [TIMESTAMP]).

Din bank vil bruke sin egen valutakurs ved gjennomføring av betalingen.
Det endelige beløpet mottaker får kan avvike fra dette estimatet.

Drop legger til et påslag på [X]% på referansekursen. Dette er inkludert i "Effektiv kurs"
ovenfor.

Gebyret på [FEE] NOK er fast og vil ikke endres.

Ved å bekrefte godtar du at totalkostnaden på [TOTAL] NOK trekkes fra din bankkonto.
```

English (Secondary):

```
IMPORTANT INFORMATION ABOUT CURRENCY CONVERSION

This is an estimate based on today's exchange rate from Norges Bank (updated [TIMESTAMP]).

Your bank will apply its own exchange rate when executing the payment.
The final amount received may differ from this estimate.

Drop applies a markup of [X]% on the reference rate. This is included in the "Effective rate"
above.

The fee of [FEE] NOK is fixed and will not change.

By confirming, you agree that the total cost of [TOTAL] NOK will be debited from your bank
account.
```

---

10. Monitoring & Alerting

10.1 Metrics to Track

Metric	Threshold	Alert If	Action
Stale rate count	0	> 5	Investigate API provider
Rate drift (vs Norges Bank)	<1%	>3%	Check provider, notify admin
API failure rate	<1%	>5%	Switch to fallback source
Rate refresh latency	<5s	>10s	Optimize API calls
Fee revenue (daily)	N/A	Sudden drop >50%	Check fee config changes

10.2 Slack Alerts

Webhook URL: `SLACK_WEBHOOK_URL` env var

Alert Conditions:

```
// /lib/services/fx-alerts.ts
export async function checkRateDrift(
  from: string,
  to: string,
  norgesBankRate: number,
  providerRate: number
) {
  const driftPercent = Math.abs((providerRate - norgesBankRate) / norgesBankRate) * 100;

  if (driftPercent > 3) {
    const alertId = randomId('alert');
    await run(`
      INSERT INTO fx_rate_alerts (
        id, alert_type, severity, from_currency, to_currency, details
      ) VALUES (?, 'rate_drift', 'high', ?, ?, ?)
    `, [
      alertId,
      from,
      to,
      JSON.stringify({ norgesBankRate, providerRate, driftPercent })
    ]);

    await sendSlackAlert({
```

```
      severity: 'high',
      title: `Rate drift detected: ${from}-${to}`,
      message: `Norges Bank: ${norgesBankRate}, Provider: ${providerRate}
(${driftPercent.toFixed(2)}% drift)`,
      alertId,
    });
  }
}
```

---

11. Implementation Phases

Phase 1: Foundation (Week 1) — Core Infrastructure

Deliverables:

- Enhanced `exchange_rates` table schema (source, markup, is_stale columns)
- `fee_configs` table + seeded data
- `fx_rate_history` table
- `fx_rate_alerts` table
- Fee calculation engine (`/lib/services/fees.ts`)
- Enhanced rate service with staleness detection
- Unit tests for fee calculation + rate retrieval

Acceptance Criteria:

- `calculateFee()` correctly handles percentage, flat, tiered fees
- `getRate()` detects stale rates and flags them
- Database migrations run successfully on SQLite + PostgreSQL

Effort: 3 days (1 builder agent)

Phase 2: Norges Bank Integration (Week 1) — Primary Data Source

Deliverables:

- Norges Bank API client (`/lib/providers/norges-bank.ts`)

- SDMX-JSON parser for Norges Bank response format
- Daily refresh cron job (16:30 CET)
- Seed all 40 currencies from Norges Bank
- Integration tests (mock Norges Bank API responses)

Acceptance Criteria:

- All 30+ Drop corridors have rates from Norges Bank
- Rates update automatically every business day at 16:30 CET
- Parser handles SDMX-JSON format correctly
- Graceful handling of Norges Bank API downtime (use cached rates)

Effort: 2 days (1 builder agent)

Phase 3: Commercial Provider Integration (Week 2) — Real-Time Fallback

Deliverables:

- ExchangeRate-API client (`/lib/providers/exchangerate-api.ts`)
- Hybrid refresh strategy (Norges Bank daily + ExchangeRate-API 10-min for EUR/USD/GBP)
- 10-minute cron job for high-volume corridors
- Rate drift detection (compare Norges Bank vs ExchangeRate-API)
- Admin alert on >3% drift

Acceptance Criteria:

- EUR, USD, GBP rates refresh every 10 minutes
- Other corridors use Norges Bank daily rate
- Drift alerts appear in `fx_rate_alerts` table
- Slack webhook fires for high drift (>3%)

Effort: 2 days (1 builder agent)

Cost: $0 (free tier: 1,500 requests/month = 50/day, sufficient for MVP)

Phase 4: Enhanced Disclosure UI (Week 2) — Frontend

Deliverables:

- Enhanced `pre-payment-disclosure.tsx` component

    - Rate breakdown section (reference + markup + effective)
    - Rate validity countdown timer
    - PSD2 regulatory text
    - Fee breakdown (tiered if applicable)
- Homepage rate calculator widget
- Updated `/api/transactions/disclosure` endpoint (return full breakdown)

Acceptance Criteria:

- Disclosure modal shows: reference rate, markup, effective rate, source, last updated
- Countdown timer shows "Rate valid for X minutes"
- PSD2 text in Norwegian + English
- Calculator works without login (public API route)

Effort: 3 days (1 builder + 1 designer agent)

Phase 5: Admin Tools & Monitoring (Week 3) — Ops Dashboard

Deliverables:

- `/admin/fx-monitoring` dashboard page
- `GET /api/admin/fx/alerts` endpoint
- `POST /api/admin/fx/refresh` manual refresh endpoint
- Slack webhook integration for alerts
- Rate drift monitoring (background job every 10 min)
- Stale rate detection (background job every 10 min)

Acceptance Criteria:

- Dashboard shows all rates, staleness, drift, alerts
- Manual refresh button works (triggers immediate API fetch)
- Slack alerts received for high/critical issues
- Unresolved alerts badge in admin nav

Effort: 2 days (1 builder agent)

Phase 6: Rate Locking (Future — Deferred) — Advanced Feature

Deliverables:

- `POST /api/fx/lock-rate` endpoint
- Lock expiry background job (expire after 10 min)
- UI: "Lock this rate" button in disclosure modal (for transfers >10k NOK)
- Backend: Verify `lockId` during transaction submission

Acceptance Criteria:

- User can lock rate for 10 minutes
- Locked rate guaranteed during checkout
- Expired locks show error message
- Lock usage logged in `fx_rate_history`

Effort: 2 days (1 builder agent)

Deferral Reason: Not MVP-critical. Can ship without rate locking. Add when user feedback indicates need.

---

12. Cost Analysis

12.1 Infrastructure Costs

Component	Provider	Cost	Notes
Norges Bank API	Norges Bank	$0	Free, open API
ExchangeRate-API (Free Tier)	ExchangeRate-API.com	$0	1,500 requests/month (50/day)
ExchangeRate-API (Paid Tier)	ExchangeRate-API.com	$9/month	100k requests (when traffic grows)
Cron Jobs (Vercel)	Vercel	$0	Included in Hobby/Pro plan
Slack Webhook	Slack	$0	Free tier sufficient

Total Monthly Cost (MVP): $0 **Total Monthly Cost (Production >50 txs/day):** $9

12.2 Revenue Impact

Fee Revenue Projection:

Scenario	Avg Transfer	Fee %	Txs/Month	Monthly Revenue
MVP (Beta)	2,000 NOK	0.5%	100	1,000 NOK ($95)
Growth	3,000 NOK	0.5%	500	7,500 NOK ($710)

Scenario	Avg Transfer	Fee %	Txs/Month	Monthly Revenue
Scale	4,000 NOK	0.5%	2,000	40,000 NOK ($3,800)

Break-Even: 10 transactions/month covers $9 API cost (at 0.5% fee on 2,000 NOK avg)

---

13. Acceptance Criteria

13.1 Functional Requirements

- ☐ **Live Rates:** EUR, USD, GBP refresh every 10 minutes from ExchangeRate-API
- ☐ **Official Rates:** All other corridors use Norges Bank daily rate (16:00 CET)
- ☐ **Fee Transparency:** User sees breakdown: amount + fee (%) + total BEFORE confirming
- ☐ **FX Transparency:** User sees: reference rate (Norges Bank) + markup + effective rate
- ☐ **Source Attribution:** Disclosure shows "Rate from Norges Bank, updated X minutes ago"
- ☐ **Rate Validity:** User told "Rate valid for 10 minutes" with countdown timer
- ☐ **PSD2 Compliance:** Disclosure text matches Article 45 requirements (Norwegian + English)
- ☐ **Configurable Fees:** Admin can change fees via `fee_configs` table (no code deploy)
- ☐ **Corridor-Specific Fees:** Different fees for different corridors (e.g., EUR cheaper than RSD)

13.2 Non-Functional Requirements

- ☐ **Performance:** Disclosure endpoint responds in <2s (including rate fetch)
- ☐ **Availability:** Graceful degradation if API fails (use cached rates + show staleness warning)
- ☐ **Accuracy:** Rate shown to user logged in `fx_rate_history` for audit
- ☐ **Monitoring:** Slack alerts for rate drift >3%, API failures, stale rates >6h
- ☐ **Security:** API keys stored in env vars (never hardcoded), admin endpoints require auth

13.3 Compliance Requirements

- ☐ **PSD2 Article 45:** All charges + exchange rate disclosed BEFORE payment execution
- ☐ **PSD2 Record-Keeping:** `fx_rate_history` retained for 5+ years (never auto-deleted)

☐ **Finanstilsynet Compliance:** Norges Bank used as official reference (Norwegian requirement)

☐ **Consumer Protection:** Clear language ("This is an estimate, your bank may use different rate")

---

14. Open Questions (For Alem)

Q1: Rate Locking Priority

Question: Should we implement rate locking in Phase 1 (MVP) or defer to Phase 2?

Recommendation: Defer. Rate locking adds complexity and is not required for PSD2 compliance. Ship MVP faster, validate user demand first.

Q2: Commercial FX Provider Choice

Question: Which paid FX API should we use?

Recommendation: ExchangeRate-API.com — Best price/performance ratio ($9/month), simplest integration, Norwegian NOK supported as base currency.

Q3: Fee Structure

Question: What fee structure should we launch with?

Recommendation: Flat 0.5% for MVP, then A/B test tiered pricing after 100 transactions. Simple beats clever for launch.

Q4: Homepage Calculator

Question: Should we show fee breakdown in homepage calculator (public, no login)?

Recommendation: Yes. Full transparency, matches Wise UX. Show: "Total cost: 5025 NOK (includes 25 NOK fee)" even before login.

---

15. Next Steps

1. **Review this spec** with Alem (approve/request changes)
2. **Answer Open Questions** (Q1-Q4 above)
3. **Prioritize phases** (which to implement first?)
4. **Assign Phase 1 to builder agent** (schema changes + fee calculation engine)
5. **Validate after Phase 1** (validator agent checks DB schema + tests)
6. **Iterate through Phases 2-5** (one phase at a time, validate each)

Estimated Timeline:

- **Phase 1:** 3 days (schema + fee engine)
- **Phase 2:** 2 days (Norges Bank integration)
- **Phase 3:** 2 days (ExchangeRate-API integration)
- **Phase 4:** 3 days (UI + disclosure enhancements)
- **Phase 5:** 2 days (admin dashboard + monitoring)

Total: 12 days (~2.5 weeks) for full PSD2-compliant FX transparency system

---

Sources

- Norges Bank Exchange Rates
- Norges Bank Data Warehouse
- PSD2 Directive 2015/2366/EU
- PSD2 Article 45 Information Requirements
- Wise Remitly Fee Comparison
- ExchangeRate-API Documentation
- Remitly vs Wise Transparency Analysis

End of Specification

Revision #4
Created 2026-02-18 08:44:44 UTC by John
Updated 2026-05-31 20:02:14 UTC by John