Runbook: Neonomics Outage

Runbook: Neonomics Open Banking Outage

Service: Neonomics Open Banking Aggregator **Severity:** CRITICAL (blocks AISP balance fetch and PISP payments) **MTTR Target:** <20 minutes **Owner:** John (AI Director)

Overview

Neonomics is Drop's Open Banking aggregator for Norwegian banks. It provides:

- **AISP (Account Information):** Fetch user's bank account balance via PSD2 consent
- **PISP (Payment Initiation):** Initiate payments from user's bank account
- **Bank connectivity:** Single API to connect to all Norwegian banks (DNB, Nordea, SpareBank 1, etc.)

Impact: If Neonomics is down, Drop cannot:

- Show bank balances
- Initiate remittance payments
- Process QR payments

This is a **critical** outage affecting core functionality.

Symptoms

Users report core features not working:

- Cannot see bank balance (shows "unavailable")
- Cannot initiate payments (error at payment step)
- Bank connection fails ("Cannot connect to bank")
- Error: "Open Banking service unavailable"

User impact: Cannot use core Drop features (balance, payments).

Diagnosis

1. Check Neonomics Service Status

External status:

```
# Neonomics has no public status page
# Test via API health check
curl -X GET https://api.neonomics.io/health \
  -H "Authorization: Bearer <api-key>" \
  -v

# Expected: HTTP 200
# If 500/503: Neonomics outage
# If timeout: Network or Neonomics connectivity issue
```

Check specific bank connectivity:

```
# List banks and their status
curl -X GET https://api.neonomics.io/banks \
  -H "Authorization: Bearer <api-key>" \
  | jq '.[] | select(.country == "NO") | {name, status, lastChecked}'

# Look for:
# - "status": "degraded" or "offline"
# - Specific bank down (e.g., DNB) vs all banks
```

2. Check Drop Logs

```
# CloudWatch Logs (production)
aws logs filter-log-events \
  --log-group-name /aws/apprunner/drop-production \
  --filter-pattern "neonomics" \
  --start-time $(date -u -d '15 minutes ago' +%s)000 \
  --region eu-west-1
```

```
# Look for:
# - "Neonomics API timeout"
# - "Neonomics 503 Service Unavailable"
# - "Bank API unavailable: DNB"
# - "Payment initiation failed: NEONOMICS_TIMEOUT"
```

3. Determine Scope of Outage

Is it all banks or specific banks?

```
# Count recent failures by bank
aws logs filter-log-events \
  --log-group-name /aws/apprunner/drop-production \
  --filter-pattern "Neonomics.*failed" \
  --start-time $(date -u -d '30 minutes ago' +%s)000 \
  | jq '.events[].message' \
  | grep -o '"bank":"[^"]*"' \
  | sort | uniq -c | sort -rn

# Example output:
# 45 "bank":"DNB"        ← DNB-specific issue
# 2 "bank":"Nordea"      ← Nordea working mostly
# 1 "bank":"SpareBank1"  ← SpareBank1 working
```

Is it AISP, PISP, or both?

```
# Check failure type
aws logs filter-log-events \
  --log-group-name /aws/apprunner/drop-production \
  --filter-pattern "Neonomics" \
  --start-time $(date -u -d '15 minutes ago' +%s)000 \
  | jq '.events[].message' \
  | grep -E "aisp|pisp" \
  | sort | uniq -c

# Example:
# 30 "service":"aisp"  ← AISP failing
# 45 "service":"pisp"  ← PISP failing
```

```
# If both high: full Neonomics outage
```

4. Test AISP and PISP Flows

Test AISP (balance fetch):

```
# Staging environment
TOKEN=$(curl -X POST https://drop-staging.fly.dev/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"test@example.com","password":"test1234"}' \
  | jq -r '.data.token')

curl -X GET https://drop-staging.fly.dev/api/accounts/balance \
  -H "Authorization: Bearer $TOKEN" \
  -v

# Expected: HTTP 200, balance data
# If 500: AISP broken
```

Test PISP (payment initiation):

```
curl -X POST https://drop-staging.fly.dev/api/transactions/remittance \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "recipientId": "rec_test123",
    "amount": 100,
    "currency": "NOK"
  }' \
  -v

# Expected: HTTP 200, payment initiated
# If 500: PISP broken
```

5. Check Neonomics API Credentials

```
# Verify API key is valid
bw get item "Neonomics API" --session $BW_SESSION
```

```
# Check App Runner environment variables
aws apprunner describe-service \
  --service-arn <ARN> \
  --region eu-west-1 \
  | jq
'.Service.SourceConfiguration.ImageRepository.ImageConfiguration.RuntimeEnvironmentVariables'
\
  | grep NEONOMICS

# Expected:
# NEONOMICS_API_KEY: <exists>
# NEONOMICS_ENVIRONMENT: production
```

Common Causes & Solutions

Cause 1: Neonomics Full Outage (All Banks)

Probability: 10% (rare but critical)

Symptoms:

- ALL banks fail (DNB, Nordea, SpareBank 1, etc.)
- All AISP and PISP requests timeout or return 503
- Neonomics API health check fails

Solution:

1. **Verify full outage:**
   ```
   # Test multiple endpoints
   curl -X GET https://api.neonomics.io/health -v
   curl -X GET https://api.neonomics.io/banks -H "Authorization: Bearer <key>" -v

   # If both fail: confirmed full outage
   ```
2. **Contact Neonomics support URGENTLY:**
   - Email: support@neonomics.io
   - Slack: #neonomics-support (if available)
   - Phone: +47 XXXX XXXX (check Neonomics Dashboard)
3. **Communicate to users (Norwegian):**

```
Emne: Betalingstjenester midlertidig utilgjengelige

Hei,

Vi opplever for øyeblikket tekniske problemer med vår betalingsleverandør.
Dette påvirker:
- Visning av saldo
- Nye betalinger

Vi jobber med å gjenopprette tjenesten så raskt som mulig.
Estimert løsning: [X minutter/timer]

Mvh,
Drop
```

4. **Enable degraded mode:**

```
# Show cached balances, disable new payments
aws apprunner update-service --service-arn <ARN> \
  --instance-configuration "EnvironmentVariables={
    AISP_MODE=cached,
    PISP_MODE=disabled,
    NEONOMICS_FALLBACK=true
  }"
```

5. **Show maintenance banner in app:**

```
⚠ Betalinger midlertidig utilgjengelig
Vi opplever tekniske problemer. Saldo vises med forsinkelse.
Betalinger er deaktivert midlertidig.
```

6. **Monitor Neonomics status:**
   - Check API health every 5 minutes
   - When API returns 200: test AISP/PISP flows
   - Re-enable features gradually
7. **Re-enable live mode when resolved:**

```
aws apprunner update-service --service-arn <ARN> \
  --instance-configuration "EnvironmentVariables={
    AISP_MODE=live,
    PISP_MODE=live,
    NEONOMICS_FALLBACK=false
```

```
  }"
```

ETA: Depends on Neonomics (typically <2 hours for major incidents)

Cause 2: Specific Bank API Down

Probability: 25% (one bank's API temporarily unavailable)

Symptoms:

- Only users of specific bank (e.g., DNB) affected
- Other banks work fine (Nordea, SpareBank 1)
- Logs show: "Bank API timeout: DNB"

Common reasons:

- Bank's API maintenance (often 02:00-06:00 CET)
- Bank's API outage
- Bank rate limiting Neonomics
- Bank API certificate expired

Solution:

1. **Identify affected bank:**

```
# Count failures by bank
aws logs filter-log-events \
  --log-group-name /aws/apprunner/drop-production \
  --filter-pattern "Bank API" \
  --start-time $(date -u -d '30 minutes ago' +%s)000 \
  | jq '.events[].message' \
  | grep -o '"bank":"[^"]*"' \
  | sort | uniq -c | sort -rn
```

2. **Check bank status:**
   - **DNB:** https://www.dnb.no/drift
   - **Nordea:** https://www.nordea.no/info/driftsmeldinger
   - **SpareBank 1:** https://www.sparebank1.no/driftsmeldinger
   - Norwegian banks often announce maintenance
3. **Contact Neonomics to verify:**
   - Neonomics may already know about bank API issues
   - Ask for ETA on bank connectivity restoration

4. **Notify affected users (bank-specific):**

```
-- Find users with affected bank
SELECT user_id, email
FROM bank_accounts
JOIN users ON users.id = bank_accounts.user_id
WHERE bank_name = 'DNB';
```

   Email (Norwegian):

```
Emne: Problemer med [Bank] tilkobling

Hei,

Vi opplever for øyeblikket problemer med tilkoblingen til [Bank].
Dette skyldes tekniske problemer hos banken.

Andre banker virker som normalt.
Hvis du har konto i en annen bank, kan du bruke den i mellomtiden.

Estimert løsning: [X minutter/timer]

Mvh,
Drop
```

5. **Graceful degradation (bank-specific):**

```
// src/lib/neonomics-client.ts
async function fetchBalance(userId: string, bankId: string) {
  try {
    return await neonomicsAPI.getBalance(userId, bankId);
  } catch (error) {
    if (error.code === 'BANK_API_TIMEOUT' && error.bank === 'DNB') {
      // Return cached balance for DNB users
      const cached = await getCachedBalance(userId);
      return {
        balance: cached?.balance || null,
        currency: 'NOK',
        lastUpdated: cached?.timestamp,
        warning: 'DNB opplever tekniske problemer. Saldo kan være utdatert.'
      };
    }
```

```
    throw error;
  }
}
```

ETA: Depends on bank (typically <2 hours for maintenance, <4 hours for incidents)

Cause 3: Neonomics API Rate Limiting

Probability: 15% (during peak hours or viral growth)

Symptoms:

- Logs show: HTTP 429 "Too Many Requests"
- Intermittent failures (some requests succeed, others fail)
- Rate limit headers in logs

Solution:

1. **Check rate limit headers:**

```
aws logs filter-log-events \
  --log-group-name /aws/apprunner/drop-production \
  --filter-pattern "X-RateLimit" \
  --start-time $(date -u -d '10 minutes ago' +%s)000 \
  | jq -r '.events[].message' \
  | grep -E "X-RateLimit-(Limit|Remaining|Reset)"
```

2. **Implement request throttling:**

```
// src/lib/neonomics-client.ts
import PQueue from 'p-queue';

const queue = new PQueue({
  concurrency: 10,      // Max 10 concurrent requests
  interval: 1000,        // Per second
  intervalCap: 50        // Max 50 requests per second
});

export async function callNeonomics(endpoint: string, options: any) {
  return queue.add(() =>
    fetch(`https://api.neonomics.io${endpoint}`, {
```

```
      ...options,
      headers: {
        'Authorization': `Bearer ${process.env.NEONOMICS_API_KEY}`,
        ...options.headers,
      },
    })
  );
}
```

3. **Aggressive caching during rate limit:**

```
// Cache balance for 5 minutes during rate limit (vs 1 minute normally)
const CACHE_TTL_NORMAL = 60;      // 1 minute
const CACHE_TTL_RATE_LIMIT = 300; // 5 minutes

async function getBalanceWithCache(userId: string) {
  const cached = await redis.get(`balance:${userId}`);
  if (cached) return JSON.parse(cached);

  try {
    const balance = await neonomicsAPI.getBalance(userId);
    await redis.setex(`balance:${userId}`, CACHE_TTL_NORMAL,
JSON.stringify(balance));
    return balance;
  } catch (error) {
    if (error.status === 429) {
      // Extend cache during rate limit
      if (cached) {
        await redis.expire(`balance:${userId}`, CACHE_TTL_RATE_LIMIT);
        return JSON.parse(cached);
      }
    }
    throw error;
  }
}
```

4. **Contact Neonomics to increase rate limit:**
   - Email: support@neonomics.io
   - Provide traffic stats (requests/day, peak times)
   - Request higher API quota

ETA: 5 minutes (automatic throttling), 1-2 days (if quota increase needed)

Cause 4: Invalid or Expired API Credentials

Probability: 5% (after credential rotation or account issue)

Symptoms:

- Logs show: "401 Unauthorized" or "403 Forbidden"
- All Neonomics API calls fail immediately
- API health check returns 401

Solution:

1. **Verify Neonomics API credentials:**

```
bw get item "Neonomics API" --session $BW_SESSION


# Check:
# - API key is correct
# - Not expired
# - Correct environment (production vs sandbox)
```

2. **Regenerate API key (if needed):**
   - Login to Neonomics Dashboard (if available)
   - Navigate to Settings → API Keys
   - Generate new API key
   - Copy to Vaultwarden
3. **Update App Runner environment variables:**

```
aws apprunner update-service --service-arn <ARN> \
  --instance-configuration "EnvironmentVariables={
    NEONOMICS_API_KEY=<new-key>,
    NEONOMICS_ENVIRONMENT=production
  }"
```

4. **Trigger deployment:**

```
aws apprunner start-deployment --service-arn <ARN> --region eu-west-1
```

5. **Test after deployment:**

```
curl -X GET https://getdrop.no/api/accounts/balance \
  -H "Authorization: Bearer <test-user-token>" \
  -v
```

```
# Expected: HTTP 200, balance data
```

ETA: 10 minutes

Cause 5: PSD2 Consent Expired (AISP Only)

Probability: 20% (affects AISP, not PISP)

Symptoms:

- Only AISP (balance fetch) fails
- PISP (payments) still works
- Logs show: "CONSENT_EXPIRED" or "CONSENT_INVALID"
- Specific users affected (not all)

Note: This is actually a user-level issue, not a Neonomics outage. See `aisp-balance-failure.md` runbook for full details.

Quick solution:

1. **Identify users with expired consent:**

   ```
   SELECT user_id, email, bank_name, consent_expires_at
   FROM bank_accounts
   JOIN users ON users.id = bank_accounts.user_id
   WHERE consent_expires_at < datetime('now');
   ```

2. **Notify users to re-authorize (Norwegian):**

   ```
   Push notification:
   Banktilkobling utløpt – Trykk her for å fornye
   ```

3. **User re-authorizes via BankID + bank consent flow**

ETA: Immediate (user action required)

Cause 6: Network or Firewall Issues

Probability: 5% (AWS security group misconfiguration)

Symptoms:

- Logs show: "Connection timeout" or "ECONNREFUSED"
- Neonomics API requests never reach destination
- Works locally but fails in production

Solution:

1. **Check outbound connectivity:**

```
# App Runner egress is unrestricted by default
# If using VPC connector, check security group
aws ec2 describe-security-groups \
  --group-ids <vpc-connector-sg> \
  --region eu-west-1 \
  | jq '.SecurityGroups[].IpPermissionsEgress'
```

2. **Test DNS resolution:**

```
nslookup api.neonomics.io

# Should resolve to Neonomics IPs
# If NXDOMAIN: DNS issue
```

3. **Check AWS service health:**

```
# Check App Runner service events
aws apprunner list-operations \
  --service-arn <ARN> \
  --region eu-west-1
```

4. **Whitelist Neonomics IPs (if using strict firewall):**
   - Contact Neonomics for IP ranges
   - Add to security group outbound rules (port 443)

ETA: 15 minutes (if quick fix), 1 hour (if requires networking changes)

Emergency Workarounds

Option 1: Cached Balance + Disable Payments

Use case: Neonomics down >30 minutes, no ETA

Steps:

1. Enable cached balance mode:

```
aws apprunner update-service --service-arn <ARN> \
  --instance-configuration "EnvironmentVariables={
    AISP_MODE=cached,
    AISP_CACHE_TTL=3600,
    PISP_MODE=disabled
  }"
```

2. Show warning banner in app:

```
⚠ Betalinger midlertidig utilgjengelige
Saldo vises med forsinkelse (opptil 1 time).
Nye betalinger er deaktivert til tjenesten er tilbake.
```

3. Allow read-only features:
   - Users can see cached balance
   - Users can see transaction history
   - Cannot initiate new payments
4. **Re-enable when Neonomics is back:**

```
aws apprunner update-service --service-arn <ARN> \
  --instance-configuration "EnvironmentVariables={
    AISP_MODE=live,
    PISP_MODE=live
  }"
```

Risk: Stale balance data. Users may think they have more/less money than reality.

Option 2: Queue Payments for Later Processing

Use case: PISP down, users need to make urgent payments

Steps:

1. Queue payment requests:

```
// src/app/api/transactions/remittance/route.ts
export async function POST(request: Request) {
  const paymentData = await request.json();

  try {
```

```
    return await neonomicsAPI.initiatePayment(paymentData);
  } catch (error) {
    if (error.code === 'NEONOMICS_UNAVAILABLE') {
      // Queue for later
      await db.insert('pending_payments', {
        user_id: userId,
        payment_data: paymentData,
        status: 'queued',
        created_at: new Date(),
      });

      return {
        success: true,
        message: 'Betaling satt i kø. Vil bli behandlet innen 2 timer.',
      };
    }
    throw error;
  }
}
```

2. Process queue when Neonomics is back:

```
node ~/ALAI/products/Drop/scripts/process-pending-payments.js
```

3. Notify users when payment completes:

```
Din betaling er behandlet
Betalingen på [amount] til [recipient] er fullført.
```

Risk: Delayed payments. User may expect instant transfer.

Monitoring & Alerts

Metrics to Track

- **Neonomics API success rate:** Should be >99%
- **Neonomics API latency:** p50 <2s, p95 <5s, p99 <10s
- **Bank-specific failure rate:** Track DNB, Nordea, SpareBank 1 separately

Alert Rules

```
// src/lib/neonomics-monitor.ts
export async function trackNeonomicsFailure(service: 'aisp' | 'pisp', error: any) {
  const failureRate = await calculateFailureRate('neonomics', 'last_5_minutes');

  if (failureRate > 0.1) { // 10% failure rate
    await sendAlert({
      severity: 'critical',
      title: 'Neonomics API failure rate high',
      message: `${(failureRate * 100).toFixed(1)}% of Neonomics calls failing`,
      service,
    });
  }
}
```

Post-Incident Actions

1. **Process queued operations:**

   ```
   SELECT * FROM pending_payments WHERE status = 'queued';
   -- Retry all pending payments
   ```

2. **Document incident:**

   ```
   touch ~/ALAI/products/Drop/comms/incidents/$(date +%Y-%m-%d)-neonomics-outage.md
   ```

3. **Review SLA with Neonomics:**
   - Check if outage violated SLA
   - Request compensation/credits
   - Discuss redundancy options
4. **Improve resilience:**
   - Add Neonomics health check (synthetic test every 5 min)
   - Implement circuit breaker for Neonomics API
   - Consider multi-provider strategy (backup Open Banking aggregator)

Escalation

Time	Action
0 min	John starts diagnosis
10 min	If full Neonomics outage confirmed, notify Alem
20 min	If not resolved, enable degraded mode (cached balance, disable payments)
30 min	Contact Neonomics support via phone if no response
1 hour	Public communication to all users
2 hours	Assess alternative Open Banking providers (emergency only)

Contacts

- **Neonomics Support:** support@neonomics.io
- **Neonomics Slack:** #neonomics-support (if available)
- **Neonomics Phone:** +47 XXXX XXXX (check Neonomics Dashboard)
- **Internal:** Alem (CEO, final decision on fallback modes)

Related Documentation

- `docs/architecture/open-banking.md` — Neonomics AISP/PISP flow
- `src/lib/neonomics-client.ts` — Neonomics API client
- `docs/compliance/psd2-requirements.md` — PSD2 regulatory requirements
- `support/runbooks/aisp-balance-failure.md` — AISP-specific failures
- `support/runbooks/pisp-payment-failure.md` — PISP-specific failures
- Vaultwarden item: "Neonomics API" — Credentials

Last Updated: 2026-02-22 **Next Review:** Before Phase 2 (Banking Integration)

Revision #3
Created 2026-03-06 15:44:54 UTC by John
Updated 2026-05-25 07:27:40 UTC by John