Runbook: Swan API Outage

Runbook: Swan BaaS API Outage

Service: Swan Banking-as-a-Service **Severity:** CRITICAL (blocks accounts, cards, payments if Swan is primary provider) **MTTR Target:** <15 minutes **Owner:** John (AI Director)

Overview

Swan provides core banking infrastructure for Drop. Depending on Drop's architecture phase, Swan may handle:

- **Account creation** (virtual IBAN accounts for users)
- **Card issuance** (virtual/physical debit cards)
- **Payment processing** (domestic/international transfers)
- **Balance management** (wallet balances, not Open Banking)

Impact: If Swan is the primary BaaS provider, an outage affects ALL core banking operations.

Symptoms

Users report critical failures:

- Cannot create new account
- Cannot view wallet balance (if using Swan wallets)
- Card payments fail or decline
- Error: "Banking service unavailable"
- Dashboard shows "System error" for account-related features

User impact: Complete inability to use banking features (depending on Drop's reliance on Swan).

Diagnosis

1. Check Swan Status Page

External status:

```
# Swan official status page
open https://status.swan.io

# Check for:
# - Incident reported
# - Degraded performance
# - Scheduled maintenance
```

2. Test Swan API

Health check:

```
# GraphQL health query
curl https://api.swan.io/graphql \
  -H "Authorization: Bearer <api-key>" \
  -H "Content-Type: application/json" \
  -d '{"query": "{viewer{id}}"}' \
  -v

# Expected: HTTP 200, {"data": {"viewer": {"id": "..."}}}
# If 500/503: Swan API down
# If 401: Credential issue
# If timeout: Network or Swan connectivity issue
```

Test account creation:

```
# Attempt to create test account
curl https://api.swan.io/graphql \
  -H "Authorization: Bearer <api-key>" \
  -H "Content-Type: application/json" \
  -d '{
    "query": "mutation { createAccount(input: {name: \"Test Account\"}) { id } }"
```

```
  }' \
  -v

# Expected: HTTP 200 with account ID
# If error: Check response for Swan error codes
```

3. Check Drop Logs

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

# Look for:
# - "Swan API timeout"
# - "Swan GraphQL error: INTERNAL_SERVER_ERROR"
# - "Swan 503 Service Unavailable"
# - "Swan rate limit exceeded"
```

4. Check Swan API Credentials

```
# Verify Swan API key is valid
bw get item "Swan 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 SWAN

# Expected:
# SWAN_API_KEY: <exists>
# SWAN_ENVIRONMENT: production (or sandbox)
```

```
# SWAN_PARTNER_ID: <partner-id>
```

5. Check Recent Swan API Changes

Review Swan changelog:

```
# Swan may deprecate API endpoints or change schemas
# Check Swan developer portal for breaking changes
open https://docs.swan.io/changelog

# Review recent GraphQL schema changes
# Verify Drop uses supported API versions
```

Common Causes & Solutions

Cause 1: Swan Service Outage (External)

Probability: 5% (Swan is highly reliable, but incidents happen)

Symptoms:

- Swan status page reports incident
- All Swan API calls fail with 500/503
- No error in Drop code/config
- Social media mentions Swan issues

Solution:

1. **Verify outage scope:**
   - Check Swan status page
   - Test API from different networks (rule out local network issue)
   - Contact Swan support for ETA
2. **Communicate to users (Norwegian):**

```
Emne: Bankfunksjoner midlertidig utilgjengelig

Hei,

Vår bankinfrastruktur-leverandør (Swan) opplever tekniske problemer.
```

```
Dette påvirker:
- Kontoopprettelse
- Korttransaksjoner
- Overføringer

Vi overvåker situasjonen og forventer at tjenesten er tilbake innen [X
minutter/timer].

Mvh,
Drop
```

3. **Enable degraded mode:**

```
# Disable features that depend on Swan
aws apprunner update-service --service-arn <ARN> \
  --instance-configuration "EnvironmentVariables={
    FEATURE_ACCOUNTS=disabled,
    FEATURE_CARDS=disabled,
    SWAN_MODE=degraded
  }"

# Show maintenance banner in app
```

4. **Monitor Swan status:**
   - Subscribe to Swan status updates (RSS/email)
   - Check every 10 minutes for resolution
   - Test API as soon as Swan reports "Resolved"
5. **Re-enable features when Swan is back:**

```
aws apprunner update-service --service-arn <ARN> \
  --instance-configuration "EnvironmentVariables={
    FEATURE_ACCOUNTS=enabled,
    FEATURE_CARDS=enabled,
    SWAN_MODE=live
  }"
```

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

Cause 2: Invalid or Expired API Credentials

Probability: 15% (after credential rotation or Swan account changes)

Symptoms:

- Logs show: "401 Unauthorized" or "Forbidden"
- All Swan API requests fail immediately
- Swan API test returns authentication error

Solution:

1. **Verify Swan API credentials:**

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


# Check:
# - API key is not expired
# - API key has correct permissions (accounts, cards, payments)
# - Partner ID is correct
```

2. **Regenerate API key (if needed):**
   - Login to Swan Dashboard: https://dashboard.swan.io
   - Navigate to Settings → API Keys
   - Revoke old key, generate new key
   - Copy new key to Vaultwarden
3. **Update App Runner environment variables:**

```
aws apprunner update-service --service-arn <ARN> \
  --source-configuration "ImageRepository={...}" \
  --instance-configuration "EnvironmentVariables={
    SWAN_API_KEY=<new-key>,
    SWAN_PARTNER_ID=<partner-id>
  }"
```

4. **Trigger deployment:**

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

5. **Test after deployment (3-5 min):**

```
curl https://getdrop.no/api/accounts/create \
  -H "Authorization: Bearer <test-user-token>" \
  -H "Content-Type: application/json" \
  -d '{"accountType": "personal"}' \
  -v


# Expected: HTTP 200, account created
```

ETA: 10 minutes

Cause 3: Swan API Rate Limiting

Probability: 10% (during high-traffic events or viral growth)

Symptoms:

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

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 Swan
```

2. **Implement request queuing:**

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

const queue = new PQueue({
  concurrency: 5,     // Max 5 concurrent Swan requests
  interval: 1000,      // Per second
  intervalCap: 20      // Max 20 requests per second
});

export async function swanGraphQL(query: string, variables?: any) {
  return queue.add(() =>
    fetch('https://api.swan.io/graphql', {
      method: 'POST',
      headers: {
        'Authorization': `Bearer ${process.env.SWAN_API_KEY}`,
        'Content-Type': 'application/json',
```

```
      },
      body: JSON.stringify({ query, variables }),
    })
  );
}
```

3. **Exponential backoff on retry:**

```
async function retrySwan(operation: () => Promise<any>, attempt = 1) {
  try {
    return await operation();
  } catch (error) {
    if (error.status === 429 && attempt <= 3) {
      const delay = 1000 * Math.pow(2, attempt); // 2s, 4s, 8s
      await sleep(delay);
      return retrySwan(operation, attempt + 1);
    }
    throw error;
  }
}
```

4. **Contact Swan to increase rate limit:**
   - Email Swan support with traffic stats
   - Provide justification: user growth, peak times
   - Request higher API quota

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

Cause 4: Swan GraphQL Schema Change (Breaking)

Probability: 5% (Swan updates API, breaks Drop integration)

Symptoms:

- Logs show: "GraphQL validation error"
- Specific queries fail: "Field 'X' doesn't exist on type 'Y'"
- Swan API works for some operations, fails for others

Solution:

1. **Check Swan changelog:**

```
# Review recent API changes
open https://docs.swan.io/changelog

# Look for:
# - Deprecated fields
# - Required fields added
# - Type changes
```

2. **Identify breaking changes:**

```
# Compare current Drop queries to Swan schema
# Example: account creation query
grep -r "createAccount" src/lib/swan-client.ts

# Cross-reference with Swan GraphQL schema
# https://api.swan.io/graphql (GraphQL Playground)
```

3. **Update Drop GraphQL queries:**

```
// Before (deprecated)
mutation {
  createAccount(input: { name: "User Account" }) {
    id
    balance  // □ Deprecated field
  }
}

// After (updated)
mutation {
  createAccount(input: { name: "User Account" }) {
    id
    balances {  // □ New field structure
      available
      currency
    }
  }
}
```

4. **Test updated queries:**

```
# Test in Swan GraphQL Playground first
# Then deploy to staging
# Verify all Swan-dependent features work
```

5. **Deploy fix:**

```
git add src/lib/swan-client.ts
git commit -m "Fix: Update Swan GraphQL queries to match latest schema"
git push origin main

# CI/CD triggers deployment
```

ETA: 30 minutes (if simple field change), 2 hours (if major refactor needed)

Cause 5: Network or Firewall Issues

Probability: 5% (AWS security group misconfiguration)

Symptoms:

- Logs show: "Connection timeout" or "ECONNREFUSED"
- Swan 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.swan.io

# Should resolve to Swan 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 \
  | jq '.OperationSummaryList[0]'
```

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

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

Cause 6: Swan Account Suspended or Payment Overdue

Probability: 2% (billing issue or compliance violation)

Symptoms:

- All Swan API calls fail with "Account suspended"
- Swan Dashboard shows billing alert
- Email from Swan about overdue payment or compliance issue

Solution:

1. **Check Swan Dashboard:**
   - Login: https://dashboard.swan.io
   - Look for alerts: billing, compliance, KYC
2. **Resolve billing issue:**
   - If overdue payment: pay immediately via Swan Dashboard
   - If billing method expired: update payment method
   - Contact Swan billing: billing@swan.io
3. **Resolve compliance issue:**
   - Swan requires KYC for partner accounts
   - Upload missing documents (company registration, director ID, etc.)
   - Respond to Swan compliance team ASAP
4. **Request urgent reactivation:**
   - Email Swan support: support@swan.io
   - Subject: "URGENT: Account reactivation needed - [Partner ID]"
   - Explain impact (users affected)
   - Provide evidence of issue resolution

ETA: 15 minutes (if billing), 24 hours (if compliance review needed)

Emergency Workarounds

Option 1: Degraded Mode (Disable Swan Features)

Use case: Swan down >30 minutes, no ETA, users need core app functionality

Steps:

1. Disable Swan-dependent features:

```
aws apprunner update-service --service-arn <ARN> \
  --instance-configuration "EnvironmentVariables={
    FEATURE_ACCOUNTS=disabled,
    FEATURE_CARDS=disabled,
    FEATURE_SWAN_WALLETS=disabled
  }"
```

2. Show banner in app:

```
⚠ Noen funksjoner er midlertidig utilgjengelige
Kontoopprettelse og korttransaksjoner er ikke tilgjengelig for øyeblikket.
Andre funksjoner virker som normalt.
```

3. Allow core features to work:
   - BankID login: (not Swan-dependent)
   - Open Banking balance: (uses Neonomics, not Swan)
   - PISP payments: (uses Neonomics, not Swan)
   - Swan accounts: (disabled)
4. **Re-enable when Swan is back:**

```
aws apprunner update-service --service-arn <ARN> \
  --instance-configuration "EnvironmentVariables={
    FEATURE_ACCOUNTS=enabled,
    FEATURE_CARDS=enabled,
    FEATURE_SWAN_WALLETS=enabled
  }"
```

Risk: Users cannot create accounts or use cards during outage.

Option 2: Queue Swan Operations for Later

Use case: Swan down, users need to create accounts but can wait

Steps:

1. Queue account creation requests:

```
// src/app/api/accounts/create/route.ts
export async function POST(request: Request) {
  const { accountType } = await request.json();

  try {
    return await swanClient.createAccount(accountType);
  } catch (error) {
    if (error.code === 'SWAN_UNAVAILABLE') {
      // Queue for later processing
      await db.insert('pending_accounts', {
        user_id: userId,
        account_type: accountType,
        status: 'queued',
        created_at: new Date(),
      });

      return {
        success: true,
        message: 'Account creation queued, will complete within 1 hour',
      };
    }
    throw error;
  }
}
```

2. Process queue when Swan is back:

```
# Run cron job to process pending accounts
node ~/ALAI/products/Drop/scripts/process-pending-accounts.js
```

3. Notify users when account is ready:

```
Din konto er klar!
Takk for tålmodigheten. Du kan nå bruke alle funksjoner i Drop.
```

Risk: Delayed user experience. Users may expect instant account creation.

Monitoring & Alerts

Metrics to Track

- **Swan API success rate:** Should be >99%
- **Swan API latency:** p50 <500ms, p95 <2s, p99 <5s
- **Swan error rate by operation:** Track createAccount, issueCard, makePayment separately

Alert Rules

```
// src/lib/swan-monitor.ts
export async function trackSwanFailure(operation: string, error: any) {
  const failureRate = await calculateSwanFailureRate('last_5_minutes');

  if (failureRate > 0.05) { // 5% failure rate
    await sendAlert({
      severity: 'critical',
      title: 'Swan API failure rate high',
      message: `${(failureRate * 100).toFixed(1)}% of Swan calls failing`,
      operation,
    });
  }
}
```

Post-Incident Actions

1. **Process queued operations:**

```
SELECT * FROM pending_accounts WHERE status = 'queued';
-- Retry all pending account creations
```

2. **Document incident:**

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

3. **Review SLA with Swan:**
   - Check if outage violated SLA
   - Request compensation/credits
   - Discuss failover options
4. **Improve resilience:**
   - Add Swan health check (every 5 min)
   - Implement circuit breaker for Swan API
   - Consider multi-provider strategy (backup BaaS)

Escalation

Time	Action
0 min	John starts diagnosis
5 min	If Swan status page shows incident, notify Alem
15 min	If not resolved, enable degraded mode
30 min	Contact Swan support via phone if no ETA
1 hour	Public communication to users

Contacts

- **Swan Support:** support@swan.io
- **Swan Phone:** +33 X XXXX XXXX (check Swan Dashboard for number)
- **Swan Status:** https://status.swan.io
- **Internal:** Alem (CEO, final decision on feature disabling)

Related Documentation

- `docs/architecture/banking.md` — Swan BaaS integration
- `src/lib/swan-client.ts` — Swan GraphQL client

- `docs/compliance/swan-requirements.md` — Swan partner KYC/compliance
- Vaultwarden item: "Swan API" — Credentials

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

Revision #6
Created 2026-02-23 11:29:22 UTC by John
Updated 2026-05-25 07:27:48 UTC by John