Sentry Setup

Drop Sentry Setup

Last updated: 2026-02-20 **Source:** `src/drop-app/src/lib/sentry.ts`, `src/drop-app/src/lib/sentry-server.ts`, `src/drop-api/src/lib/sentry.ts`, `src/drop-app/.env.example`

Overview

Drop uses Sentry for error tracking and performance monitoring across three components:

1. **drop-app (client-side)** - Browser errors via `@sentry/browser`
2. **drop-app (server-side)** - Next.js middleware/API errors via custom envelope API
3. **drop-api** - Backend API errors via `@sentry/node`

All three components share the same DSN and gracefully degrade to console-only logging when Sentry is not configured.

Sentry Account Setup

1. Create Free Sentry Account

1. Visit sentry.io and sign up (free tier: 5,000 errors/month)
2. Confirm email and log in

2. Create Projects

Create **two separate projects** (one for app, one for API):

Project 1: drop-app

1. Click **Projects → Create Project**
2. Platform: **Next.js**

3. Project name: `drop-app`
4. Team: Default team (or create `drop-team`)
5. Alert frequency: **On every new issue**
6. Click **Create Project**
7. Copy the DSN (format: `https://examplePublicKey@o0.ingest.sentry.io/0`)

Project 2: drop-api

1. Repeat steps above with platform **Node.js**
2. Project name: `drop-api`
3. Copy the DSN (different from drop-app)

IMPORTANT: Use separate projects to keep frontend and backend errors isolated.

Environment Variables Configuration

drop-app (.env.local)

Add these variables to `src/drop-app/.env.local`:

```
# --- Sentry (Error Tracking) ---
# Client-side error tracking (browser)
NEXT_PUBLIC_SENTRY_DSN=https://YOUR_PUBLIC_KEY@o0.ingest.sentry.io/YOUR_PROJECT_ID

# Server-side error tracking (middleware/API routes)
# NOTE: drop-app server uses custom envelope API (no @sentry/nextjs due to Turbopack
incompatibility)
# Both client and server use the SAME DSN (NEXT_PUBLIC_SENTRY_DSN)

# Optional: Performance monitoring sample rate (0.0 to 1.0, default: 0.1 = 10%)
NEXT_PUBLIC_SENTRY_TRACES_SAMPLE_RATE=0.1

# Optional: For source map uploads (requires auth token from Sentry → Settings → Auth Tokens)
SENTRY_ORG=your-org-slug
SENTRY_PROJECT=drop-app
SENTRY_AUTH_TOKEN=your-auth-token
```

drop-api (.env)

Add these variables to `src/drop-api/.env`:

```
# --- Sentry (Error Tracking) ---
SENTRY_DSN=https://YOUR_PUBLIC_KEY@o0.ingest.sentry.io/YOUR_API_PROJECT_ID

# Optional: Performance monitoring sample rate (0.0 to 1.0, default: 0.1 = 10%)
SENTRY_TRACES_SAMPLE_RATE=0.1

# Optional: For source map uploads
SENTRY_ORG=your-org-slug
SENTRY_PROJECT=drop-api
SENTRY_AUTH_TOKEN=your-auth-token
```

Where to find these values:

- **DSN:** Project Settings → Client Keys (DSN)
- **Org slug:** Settings → Organization → General Settings → Organization Slug
- **Project name:** Project Settings → General → Project Name
- **Auth token:** Settings → Auth Tokens → Create New Token (scopes: `project:releases`, `project:write`)

Verification

Test Client-Side Error Capture (drop-app)

1. Start the app: `npm run dev` (in `src/drop-app/`)
2. Open browser console: `http://localhost:3000`
3. Trigger test error via console:

   ```
   throw new Error("Sentry test error - client-side");
   ```

4. Check Sentry dashboard: **Projects → drop-app → Issues**
5. You should see the test error appear within 10 seconds

Expected behavior:

- Error logged to browser console: `[Sentry] Error captured: Error: Sentry test error - client-side`
- Error appears in Sentry dashboard with stack trace, breadcrumbs, and browser context

Test Server-Side Error Capture (drop-app)

1. Create test API route: `src/drop-app/src/app/api/sentry-test/route.ts`

```
import { NextResponse } from 'next/server';
import { captureServerError } from '@/lib/sentry-server';

export async function GET() {
  try {
    throw new Error('Sentry test error - server-side');
  } catch (error) {
    captureServerError(error as Error, { tags: { test: 'true' } });
    return NextResponse.json({ error: 'Test error sent to Sentry' }, { status: 500
});
  }
}
```

2. Visit: `http://localhost:3000/api/sentry-test`
3. Check server console: `[Sentry Server] Error captured: Error: Sentry test error - server-side`
4. Check Sentry dashboard: **Projects → drop-app → Issues**

Test API Error Capture (drop-api)

1. Start the API: `npm run dev` (in `src/drop-api/`)
2. Trigger test error via curl:

```
curl http://localhost:4000/api/sentry-test
```

3. OR create test endpoint in `src/drop-api/src/routes/test.ts`:

```
import { Router } from 'express';
import { captureError } from '../lib/sentry.js';

const router = Router();

router.get('/sentry-test', (req, res) => {
  try {
    throw new Error('Sentry test error - API');
  } catch (error) {
    captureError(error as Error, { tags: { test: 'true' } });
    res.status(500).json({ error: 'Test error sent to Sentry' });
  }
});
```

```
export default router;
```

4. Check Sentry dashboard: **Projects → drop-api → Issues**

Source Map Upload Setup

Source maps allow Sentry to show readable stack traces instead of minified code.

1. Install Sentry CLI

```
# macOS (Homebrew)
brew install getsentry/tools/sentry-cli

# Or via npm (global)
npm install -g @sentry/cli
```

2. Configure Sentry CLI

Create `.sentryclirc` in project root:

```
[defaults]
url=https://sentry.io/
org=your-org-slug
project=drop-app

[auth]
token=your-auth-token
```

IMPORTANT: Add `.sentryclirc` to `.gitignore` (contains auth token).

3. Add Build Script (drop-app)

Update `src/drop-app/package.json`:

```
{
  "scripts": {
```

```
    "build": "next build",
    "build:sentry": "next build && sentry-cli sourcemaps upload --validate .next/static"
  }
}
```

4. Test Source Map Upload

```
cd src/drop-app
npm run build:sentry
```

Expected output:

```
> Analyzing source maps for sentry
> Uploading source maps to Sentry
✓ Successfully uploaded source maps
```

5. CI/CD Integration

For automated uploads in CI/CD, add these secrets to your deployment platform:

Vercel/Railway/Fly.io:

- `SENTRY_ORG`
- `SENTRY_PROJECT`
- `SENTRY_AUTH_TOKEN`

Then update build command:

```
npm run build && sentry-cli sourcemaps upload --validate .next/static
```

Alert Rules Configuration

Recommended Alert Rules

1. New Issue Alert (drop-app)

1. Go to **Projects → drop-app → Settings → Alerts**
2. Click **Create Alert Rule**
3. Configure:

- **Conditions:** When a new issue is created
- **Filters:** Environment = production
- **Actions:**
  - Send notification to: Slack channel `#drop-alerts`
  - Send email to: alem@alai.no

4. Save rule

2. High Error Rate Alert (drop-app)

1. Create new alert rule
2. Configure:
   - **Conditions:** Number of events in an issue is more than 100 in 1 hour
   - **Filters:** Environment = production, Level = error
   - **Actions:**
     - Send notification to: Slack channel `#drop-alerts`
     - Send email to: alem@alai.no
3. Save rule

3. Critical Error Alert (drop-api)

1. Go to **Projects → drop-api → Settings → Alerts**
2. Create alert rule:
   - **Conditions:** When a new issue is created AND Level = fatal
   - **Filters:** Environment = production
   - **Actions:**
     - Send notification to: Slack channel `#drop-critical`
     - Send email to: alem@alai.no
3. Save rule

4. Performance Degradation Alert (drop-app)

1. Create alert rule:
   - **Conditions:** Average transaction duration is above 2000ms for 5 minutes
   - **Filters:** Environment = production, Transaction = /api/transactions/*
   - **Actions:**
     - Send notification to: Slack channel `#drop-performance`
2. Save rule

Slack Integration (Optional)

1. Go to **Settings → Integrations → Slack**
2. Click **Add Workspace**
3. Authorize Sentry to access your Slack workspace
4. Select channels: `#drop-alerts`, `#drop-critical`, `#drop-performance`
5. Test integration by triggering a test error

PII Scrubbing

All three Sentry integrations automatically scrub sensitive data before sending events:

Scrubbed fields:

- `password`
- `pin`
- `cardNumber`
- `cvv`
- `fødselsnummer`
- `authorization` headers
- `cookie` headers

Implementation:

- **drop-app (client):** `src/drop-app/src/lib/sentry.ts` (lines 51-76)
- **drop-app (server):** Custom envelope API (no PII in server-side events)
- **drop-api:** `src/drop-api/src/lib/sentry.ts` (lines 48-139)

Verification:

1. Trigger error with sensitive data:

```
try {
  throw new Error('Login failed for user with password=secret123');
} catch (error) {
  captureError(error, { extra: { cardNumber: '1234567890123456' } });
}
```

2. Check Sentry event:
   - Message should show: `Login failed for user with password=[REDACTED]`
   - Extra context should show: `cardNumber: [REDACTED]`

Environment-Specific Configuration

Development

- **DSN:** Optional (errors log to console only if not set)
- **Sample rate:** 1.0 (capture all errors for debugging)

- **Source maps:** Not required (local stack traces are readable)

```
# .env.local (development)
NEXT_PUBLIC_SENTRY_DSN=  # Leave empty to disable Sentry in dev
```

Staging

- **DSN:** Required (test Sentry integration before production)
- **Sample rate:** 0.5 (capture 50% of transactions)
- **Source maps:** Enabled (verify uploads work)

```
# .env.staging
NEXT_PUBLIC_SENTRY_DSN=https://YOUR_KEY@sentry.io/YOUR_PROJECT_ID
NEXT_PUBLIC_SENTRY_TRACES_SAMPLE_RATE=0.5
SENTRY_AUTH_TOKEN=your-auth-token
```

Production

- **DSN:** Required (critical for production monitoring)
- **Sample rate:** 0.1 (capture 10% of transactions to stay within free tier)
- **Source maps:** Enabled (required for readable stack traces)

```
# .env.production
NEXT_PUBLIC_SENTRY_DSN=https://YOUR_KEY@sentry.io/YOUR_PROJECT_ID
NEXT_PUBLIC_SENTRY_TRACES_SAMPLE_RATE=0.1
SENTRY_AUTH_TOKEN=your-auth-token
```

Troubleshooting

No errors appearing in Sentry dashboard

Check 1: DSN configured?

```
# drop-app
echo $NEXT_PUBLIC_SENTRY_DSN

# drop-api
```

```
echo $SENTRY_DSN
```

Check 2: Console output?

- Errors should ALWAYS log to console, even if Sentry upload fails
- Look for: `[Sentry] Error captured: ...`

Check 3: Network errors?

- Open browser DevTools → Network tab
- Filter by `sentry.io`
- Check for failed requests (should see POST to `https://o0.ingest.sentry.io/api/.../envelope/`)

Check 4: Environment mismatch?

- Sentry filters events by environment (`production`, `development`, `staging`)
- Verify `NEXT_PUBLIC_APP_ENV` or `NODE_ENV` matches your Sentry project filters

Source maps not working (minified stack traces)

Check 1: Source maps uploaded?

```
cd src/drop-app
sentry-cli releases list
```

Check 2: Release version matches?

- Sentry matches source maps by release version
- Verify `package.json` version matches uploaded release

Check 3: Upload command ran?

```
# Manually test upload
sentry-cli sourcemaps upload --validate .next/static
```

PII still appearing in events

Check 1: Verify beforeSend hook

- Inspect `src/lib/sentry.ts` (client) or `src/lib/sentry.ts` (API)
- Confirm `beforeSend` function is scrubbing sensitive keys

Check 2: Add custom scrubbing

- If new sensitive fields appear, add them to scrubbing list:

```
const sensitiveKeys = ["password", "pin", "yourNewField"];
```

Cost Management

Sentry Free Tier:

- 5,000 errors per month
- 10,000 performance units per month
- 1 GB attachments
- 30 days retention

Staying within free tier:

1. **Lower sample rate:** Set `SENTRY_TRACES_SAMPLE_RATE=0.1` (10%)
2. **Filter noisy errors:** Use `beforeSend` to ignore expected errors (e.g., 404s)
3. **Set up quotas:** Sentry → Settings → Quotas → Set monthly limits

Example: Ignore 404 errors

```
beforeSend(event, hint) {
  // Ignore 404 errors
  if (event.request?.url?.includes('/api/') &&
hint?.originalException?.message?.includes('404')) {
    return null; // Don't send to Sentry
  }
  return event;
}
```

Security Considerations

1. **Auth token storage:**
   - NEVER commit `.sentryclirc` to git
   - Store `SENTRY_AUTH_TOKEN` in CI/CD secrets, not `.env` files
2. **DSN exposure:**
   - `NEXT_PUBLIC_SENTRY_DSN` is exposed to client-side code (safe - it's public)
   - Sentry rate-limits abuse via DSN quotas
3. **PII scrubbing:**

- Always verify PII scrubbing works before deploying to production
- Test with real-world data patterns (Norwegian fødselsnummer, BankID tokens)

4. **Access control:**
   - Limit Sentry dashboard access to authorized team members only
   - Use Sentry Teams to restrict project access

References

- **Sentry Docs:** https://docs.sentry.io/platforms/javascript/guides/nextjs/
- **Sentry CLI:** https://docs.sentry.io/product/cli/
- **Source Maps:** https://docs.sentry.io/platforms/javascript/sourcemaps/
- **PII Scrubbing:** https://docs.sentry.io/platforms/javascript/data-management/sensitive-data/
- **Alert Rules:** https://docs.sentry.io/product/alerts/

Next Steps

1. Create Sentry account and projects (drop-app, drop-api)
2. Add DSN to `.env.local` (development) and `.env.production` (production)
3. Test error capture in all three components
4. Configure alert rules (new issues, high error rate, critical errors)
5. Set up source map uploads for production builds
6. Integrate Slack notifications (optional)
7. Monitor error dashboard daily during initial deployment

Revision #7
Created 2026-02-23 11:28:55 UTC by John
Updated 2026-05-23 10:58:26 UTC by John