

Sentry Monitoring Fix - Summary

Issues Identified

1. Client-Side Initialization Problem

- The client config was logging “[Sentry] Not running in browser environment”
- The config was checking `process.env.SENTRY_DSN` which isn’t available at runtime in the browser
- Client-side code needs to use `NEXT_PUBLIC_SENTRY_DSN` for runtime access

2. Connection Timeout Errors

- Logs showing `ETIMEDOUT` and `ENETUNREACH` errors when Sentry tries to send events
- These errors were cluttering logs but not preventing Sentry from working
- Sentry should queue events and retry, but we need to suppress these error logs

3. No Test Mechanism

- No way to easily verify if Sentry is capturing errors
- Need test endpoints to trigger intentional errors

Fixes Implemented

1. Fixed Client-Side Configuration (`sentry.client.config.ts`)

Changes:

- Changed from `process.env.SENTRY_DSN` to `process.env.NEXT_PUBLIC_SENTRY_DSN` (required for client runtime)
- Added proper browser environment check: `typeof window !== 'undefined'`
- Added try-catch wrapper around initialization
- Added `beforeSend` filter to suppress connection timeout errors
- Improved logging to differentiate between initialization states

Key Points:

- Client-side configs MUST use `NEXT_PUBLIC_` prefixed environment variables
- Without `NEXT_PUBLIC_`, the variable is only available at build time, not runtime
- This was the primary reason client-side Sentry wasn’t initializing

2. Enhanced Server-Side Configuration (`sentry.server.config.ts`)

Changes:

- Added filter in `beforeSend` to suppress connection timeout errors
- Prevents `ETIMEDOUT` and `ENETUNREACH` errors from being sent to Sentry
- These network errors are not actionable and clutter the dashboard

3. Created Test Endpoints

Server-Side Test: `/api/test-sentry-server-error`

- Throws intentional server-side error
- Adds context and breadcrumbs for debugging
- Returns JSON with instructions to check Sentry dashboard

Client-Side Test: /api/test-sentry-client-error

- Returns HTML page with interactive test buttons
- Throws client-side JavaScript errors
- Can send test messages to Sentry
- Provides visual feedback and instructions

Testing Instructions

Test Server-Side Error Tracking

1. Visit the server test endpoint:

<https://getcarelinkai.com/api/test-sentry-server-error>

2. Expected Response:

- You'll see a JSON response with error details
- Response will include link to Sentry dashboard

3. Check Sentry Dashboard:

- Go to: <https://sentry.io/organizations/carelinkai/issues/>
- Within 1-5 minutes, you should see the error appear
- Error message will start with “ TEST ERROR: Sentry server-side monitoring test”

Test Client-Side Error Tracking

1. Visit the client test page:

<https://getcarelinkai.com/api/test-sentry-client-error>

2. Click “Throw Test Error” button

- This will throw an error in the browser
- You should see a success message

3. Check Browser Console:

- Open browser DevTools (F12)
- Look for Sentry initialization message
- Should see: “ Sentry is loaded and ready”

4. Check Sentry Dashboard:

- Go to: <https://sentry.io/organizations/carelinkai/issues/>
- Within 1-5 minutes, you should see the error appear
- Error message will start with “ TEST ERROR: Sentry client-side monitoring test”

Verification Checklist

After deployment, verify the following:

- [] Visit homepage and check browser console for “[Sentry] Client-side initialization successful”
- [] Check server logs for “[Sentry] Server-side initialization successful”
- [] Check server logs for “[Sentry] Edge initialization successful”
- [] Verify no “[Sentry] Not running in browser environment” errors
- [] Test server-side error endpoint and verify error appears in Sentry
- [] Test client-side error page and verify error appears in Sentry
- [] Verify connection timeout errors are no longer cluttering logs



Expected Sentry Dashboard Activity

After testing, you should see in your Sentry dashboard:

1. Server-Side Test Error

- Error: “ TEST ERROR: Sentry server-side monitoring test...”
- Contains context about the test
- Includes breadcrumbs showing test flow

2. Client-Side Test Error

- Error: “ TEST ERROR: Sentry client-side monitoring test...”
- Contains browser context (user agent, page URL)
- Includes breadcrumbs showing user interaction



Technical Details

Environment Variables Required

- `NEXT_PUBLIC_SENTRY_DSN` : Must be set for both build-time and runtime
- Format: `https://[key]@[org].ingest.us.sentry.io/[project]`
- Current value: `https://d649b9c85c145427fcf-b62cecdeaa2d9e@o4510110703216128.ingest.us.sentry.io/4510154420089472`

Sentry Integration

- Uses `@sentry/nextjs` package
- Configured via `next.config.js` with `withSentryConfig`
- Tunnel route: `/monitoring` (helps bypass ad-blockers)
- Source maps uploaded automatically during build

Error Filtering

Both client and server configs now filter out:

- Connection timeout errors (`ETIMEDOUT`)
- Network unreachable errors (`ENETUNREACH`)
- Prisma client initialization errors (development only)



Known Issues

Connection Timeouts

- The connection timeout errors (`ETIMEDOUT` , `ENETUNREACH`) may still occur
- This appears to be a Render network/firewall issue
- Sentry queues events and retries, so errors should eventually be sent
- The tunnel route (`/monitoring`) helps bypass this by proxying through Next.js server
- These errors are now filtered from being sent to Sentry dashboard

Render Environment

- Ensure `NEXT_PUBLIC_SENTRY_DSN` is set in Render environment variables
- Without the `NEXT_PUBLIC_` prefix, client-side tracking won't work
- Server-side will work with just `SENTRY_DSN` , but using `NEXT_PUBLIC_SENTRY_DSN` works for both



Files Modified

1. `sentry.client.config.ts` - Fixed client-side initialization
2. `sentry.server.config.ts` - Added error filtering
3. `src/app/api/test-sentry-server-error/route.ts` - New test endpoint
4. `src/app/api/test-sentry-client-error/route.ts` - New test endpoint



Success Criteria

Sentry monitoring is working correctly when:

1. Both client and server initialization messages appear in logs
2. Test errors appear in Sentry dashboard within 5 minutes
3. Real application errors are captured and reported
4. No “[Sentry] Not running in browser environment” errors
5. Connection timeout errors are suppressed from dashboard



Additional Resources

- Sentry Next.js Docs: <https://docs.sentry.io/platforms/javascript/guides/nextjs/>
- Sentry Dashboard: <https://sentry.io/organizations/carelinkai/>
- Next.js Environment Variables: <https://nextjs.org/docs/app/building-your-application/configuring/environment-variables>