CspReportController

September 4, 2026

CspReportController

Kind: Controller

Source: `atloria-monorepo/apps/api/src/csp-report/csp-report.controller.ts`

Collector for browser Content-Security-Policy violation reports.

The web app ships a `Content-Security-Policy-Report-Only` header (see next.config.js) whose
`report-uri` points here. Every time the report-only policy WOULD have blocked something,
the browser POSTs a report to this endpoint. We log each distinct (directive, blocked-uri)
combination ONCE so we can enumerate exactly what an ENFORCED CSP would break — turning
"I'm afraid enforcing will break the app" into a concrete, evidence-based list.

Public + best-effort by design: browsers send these reports unauthenticated and cannot set
custom headers, so there is no guard. The global per-IP ThrottlerGuard bounds abuse, and the
in-memory de-dupe cap bounds memory. Nothing here trusts or acts on the payload.

`CspReportController` receives browser-generated Content Security Policy violation reports sent to the application's `Content-Security-Policy-Report-Only` `report-uri`. It records each distinct `(violated directive, blocked URI)` combination once, providing evidence of resources that would fail under an enforced CSP without trusting or acting on report payloads. The endpoint is intentionally public and relies on global IP throttling plus bounded in-memory de-duplication to limit abuse and memory usage.

Diagram

```
graph LR
  Browser[Browser] -->|CSP Report-Only violation POST| Controller[CspReportController]
  NextConfig[next.config.js CSP header] -->|report-uri| Controller
  Controller -->|Extract directive + blocked URI| Deduper[In-memory de-duplication]
  Deduper -->|First occurrence only| Logger[Application logger]
  Guard[Global ThrottlerGuard] -->|Rate limits by IP| Controller
```

Usage

```
// The browser posts CSP reports automatically when the Report-Only policy
// would have blocked a resource. For example:
//
// Content-Security-Policy-Report-Only:
//   default-src 'self'; report-uri /csp-report

await fetch('/csp-report', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/csp-report',
  },
  body: JSON.stringify({
    'csp-report': {
      'document-uri': 'https://app.example.com/dashboard',
      'violated-directive': 'script-src',
      'blocked-uri': 'https://cdn.example.org/widget.js',
      'original-policy': "default-src 'self'; report-uri /csp-report",
    },
  }),
});

// In normal operation, do not call this endpoint from application code.
// Browsers submit reports automatically based on the CSP header.
```

AI Coding Instructions

- Keep this endpoint unauthenticated and free of custom-header requirements; browser CSP reporting cannot reliably provide either.
- Treat all report fields as untrusted diagnostic input only—never use them for authorization, redirects, persistence decisions, or downstream requests.
- Preserve de-duplication by `(directive, blocked-uri)` and keep the cache bounded so malformed or high-cardinality reports cannot cause unbounded memory growth.
- Ensure the route remains covered by the global per-IP `ThrottlerGuard`; do not add expensive processing before rate limiting.

- When changing CSP directives in `next.config.js`, use the logged report combinations to validate what an enforced policy would block.

Relationships

- MODULE_DECLARES → `report`

Referenced By

- `CspReportModule` (MODULE_DECLARES)