> Menu

```
App Router > ... > File Conventions > instrumentation.js
```

instrumentation.js

The instrumentation.js|ts file is used to integrate observability tools into your application, allowing you to track the performance and behavior, and to debug issues in production.

To use it, place the file in the **root** of your application or inside a src folder if using one.

Enabling Instrumentation

Instrumentation is currently an experimental feature, to use the instrumentation.js file, you must explicitly opt-in by defining experimental.instrumentationHook = true; in your next.config.js:

```
import type { NextConfig } from 'next'

const nextConfig: NextConfig = {
    experimental: {
        instrumentationHook: true,
    },
    }

export default nextConfig
```

Exports

register (required)

The file exports a register function that is called **once** when a new Next.js server instance is initiated. register can be an async function.



```
import { registerOTel } from '@vercel/otel'

export function register() {
   registerOTel('next-app')
}
```

onRequestError (optional)

```
This API is available in Next.js canary.
```

You can optionally export an onRequestError function to track and send server errors to an observability tool.

```
Ts instrumentation.ts
                                                                         TypeScript ~
                                                                                       \Box
     import { type Instrumentation } from 'next'
 1
 2
 3
    export const onRequestError: Instrumentation.onRequestError = (
      err,
 5
       request,
      context
 7
     ) => {
 8
       fetch('https://.../write-log', {
         method: 'POST',
10
         body: JSON.stringify({
11
           message: err.message,
12
           request,
13
           context,
14
         }),
```

Parameters

The function accepts three parameters: [error], [request], and [context].

```
Types
    export function onRequestError(
 2
       error: { digest: string } & Error,
       request: {
 3
 4
         path: string // resource path, e.g. /blog?name=foo
        method: string // request method. e.g. GET, POST, etc
 5
        headers: { [key: string]: string }
 6
 7
       },
 8
      context: {
        routerKind: 'Pages Router' | 'App Router' // the router type
 9
        routePath: string // the route file path, e.g. /app/blog/[dynamic]
10
         routeType: 'render' | 'route' | 'action' | 'middleware' // the context in which
11
        renderSource:
12
           'react-server-components'
13
           | 'react-server-components-payload'
14
15
           'server-rendering'
         revalidateReason: 'on-demand' | 'stale' | undefined // undefined is a normal re
16
         renderType: 'dynamic' | 'dynamic-resume' // 'dynamic-resume' for PPR
17
18
       }
19
    )
```

- error: The caught error itself (type is always Error), and a digest property which is the unique ID of the error.
- request: Read-only request information associated with the error.
- context: The context in which the error occurred. This can be the type of router (App or Pages Router), and/or (Server Components ('render'), Route Handlers ('route'), Server Actions ('action'), or Middleware ('middleware')).

Version History

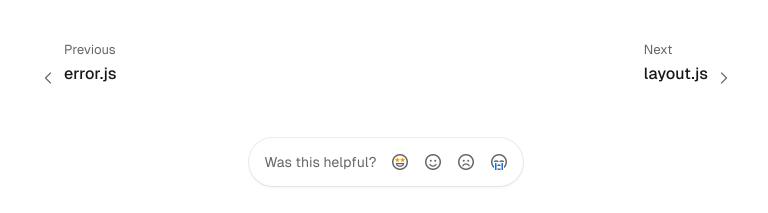
Version	Changes			
v15.0.0	onRequestError introduced			
v14.0.4	Turbopack support for instrumentation			
v13.2.0	instrumentation introduced as an experimental feature			

Learn more about Instrumentation

App Router > ... > Optimizing

Instrumentation

Learn how to use instrumentation to run code at server startup in your Next.js app



▲ Vercel	Resources	More	About Vercel	Legal
	Docs	Next.js Commerce	Next.js + Vercel	Privacy Policy

Learn Contact Sales Open Source Software

Showcase GitHub GitHub

Blog Releases X

Analytics Telemetry

Next.js Conf Governance

Previews

Subscribe to our newsletter

Stay updated on new releases and features, guides, and case studies.

you@domain.com

Subscribe

© 2024 Vercel, Inc.









