

[> 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`:

TS next.config.ts

TypeScript ▾



```
1 import type { NextConfig } from 'next'
2
3 const nextConfig: NextConfig = {
4   experimental: {
5     instrumentationHook: true,
6   },
7 }
8
9 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.



```
1 import { registerOTel } from '@vercel/otel'
2
3 export function register() {
4   registerOTel('next-app')
5 }
```

`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.

`instrumentation.ts`

TypeScript ▾



```
1 import { type Instrumentation } from 'next'
2
3 export const onRequestError: Instrumentation.onRequestError = (
4   err,
5   request,
6   context
7 ) => {
8   fetch('https://.../write-log', {
9     method: 'POST',
10    body: JSON.stringify({
11      message: err.message,
12      request,
13      context,
14    }),
15  })
```

```
15     headers: {
16       'Content-Type': 'application/json',
17     },
18   })
19 }
```

Parameters

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

Types

```
1  export function onRequestError(
2    error: { digest: string } & Error,
3    request: {
4      path: string // resource path, e.g. /blog?name=foo
5      method: string // request method. e.g. GET, POST, etc
6      headers: { [key: string]: string }
7    },
8    context: {
9      routerKind: 'Pages Router' | 'App Router' // the router type
10     routePath: string // the route file path, e.g. /app/blog/[dynamic]
11     routeType: 'render' | 'route' | 'action' | 'middleware' // the context in which
12     renderSource:
13       | 'react-server-components'
14       | 'react-server-components-payload'
15       | 'server-rendering'
16     revalidateReason: 'on-demand' | 'stale' | undefined // undefined is a normal r
17     renderType: 'dynamic' | 'dynamic-resume' // 'dynamic-resume' for PPR
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	<code>onRequestError</code> introduced
v14.0.4	Turbopack support for <code>instrumentation</code>
v13.2.0	<code>instrumentation</code> 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

Previous
< **error.js**

Next
layout.js >

Was this helpful?    

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.

