StreamableFile

August 19, 2026

StreamableFile

Kind: Class

Source: `packages/common/file-stream/streamable-file.ts`

Part of: Common

`StreamableFile` wraps a `Readable` stream or in-memory buffer so it can be returned as a streamed HTTP response. It centralizes response metadata such as content type, disposition, and length, while allowing custom stream error handling and logging.

Methods

Method	Signature	Returns
`getStream`	`getStream()`	`Readable`
`getHeaders`	`getHeaders()`	`void`
`setErrorHandler`	`setErrorHandler(handler: (err: Error, response: StreamableHandlerResponse) => void)`	`void`
`setErrorLogger`	`setErrorLogger(handler: (err: Error) => void)`	`void`

Properties

Property	Type
`logger`	`any`
`handleError`	`( err: Error, response: StreamableHandlerResponse, ) => void`
`logError`	`(err: Error) => void`

Where it refuses work

- `StreamableFile` stops the work with an early return when `res.destroyed`.

Diagram

```
graph LR
  A[File Buffer or Readable Stream] --> B[StreamableFile]
  B --> C[getStream()]
  B --> D[getHeaders()]
  B --> E[setErrorHandler()]
  B --> F[setErrorLogger()]
  C --> G[HTTP Response Stream]
  D --> G
  E --> H[Custom Error Response]
  F --> I[Application Logger]
```

Usage

```
import { Controller, Get, StreamableFile } from '@nestjs/common';
import { createReadStream } from 'node:fs';
import { join } from 'node:path';

@Controller('reports')
export class ReportsController {
  @Get('download')
  download(): StreamableFile {
    const filePath = join(process.cwd(), 'files', 'monthly-report.pdf');
    const stream = createReadStream(filePath);

    return new StreamableFile(stream, {
      type: 'application/pdf',
      disposition: 'attachment; filename="monthly-report.pdf"',
    })
      .setErrorLogger((error) => {
        console.error('Unable to stream report:', error);
      })
      .setErrorHandler((error, response) => {
        response.statusCode = 500;
        response.end('The report could not be downloaded.');
      });
  }
}
```

AI Coding Instructions

- Pass a `Readable` stream for large files to avoid loading the entire file into memory; use a buffer only for small generated content.
- Set `type` and `disposition` options when returning downloadable files so clients receive the correct MIME type and filename.

- Return the `StreamableFile` instance directly from supported controller routes rather than manually piping its stream unless custom response handling is required.
- Configure `setErrorLogger()` and `setErrorHandler()` when stream failures need application-specific logging or HTTP error responses.
- Ensure file streams are created from validated paths and that stream errors are handled, especially for files that may be missing or inaccessible.

How it works

`StreamableFile` is an exported class that packages either byte data or a readable/pipe-capable object with optional HTTP file-response metadata. It is re-exported from the common package. packages/common/file-stream/streamable-file.ts:13 packages/common/index.ts:9-12

- Its declared constructor overloads accept a `Uint8Array` or Node `Readable`, plus optional `StreamableFileOptions`. At runtime, it also accepts any object whose `pipe` property is a function. packages/common/file-stream/streamable-file.ts:37-50 packages/common/utils/shared.utils.ts:43-44
- For a `Uint8Array` —including a `Buffer` —it creates a new `Readable`, pushes the byte array, then pushes `null` to end that stream. If `options.length` is nullish, construction writes the byte-array length into the supplied options object. packages/common/file-stream/streamable-file.ts:43-47
- For an input with a functional `pipe`, it retains that input as the stream rather than wrapping it. packages/common/file-stream/streamable-file.ts:48-50
- Input that is neither a `Uint8Array` nor pipe-capable is not rejected by this constructor, and no stream is assigned; consequently, `getStream()` returns `undefined` in the tested case despite its declared `Readable` return type. Consumers that need a stream must therefore pass byte data or a pipe-capable object. packages/common/file-stream/streamable-file.ts:43-55 packages/common/test/file-stream/streamable-file.spec.ts:22-27
- `getStream()` returns the stored stream reference. packages/common/file-stream/streamable-file.ts:53-55

`StreamableFileOptions` contains `type`, `disposition`, and `length`, corresponding to `Content-Type`, `Content-Disposition`, and `Content-Length` response headers. `disposition` may be a string or string array. packages/common/file-stream/interfaces/streamable-options.interface.ts:8-21 `getHeaders()` returns these three values, defaulting `type` to `application/octet-stream` and the other two to `undefined`. packages/common/file-stream/streamable-file.ts:57-68

When an Express response body is a `StreamableFile`, the adapter sets those headers only if the response does not already contain them, then pipes `getStream()` into the response. String-array header values are joined with commas before Express sets them. packages/platform-express/adapters/express-adapter.ts:110-118 packages/platform-express/adapters/express-adapter.ts:501-522 The Fastify adapter similarly assigns absent headers and sends the stored stream. packages/platform-fastify/adapters/fastify-adapter.ts:449-482

The class exposes replaceable stream-error callbacks:

- The default `errorHandler` returns without action when `response.destroyed` is true. If headers were already sent, it calls `response.end()`. Otherwise, it sets `response.statusCode` to `HttpStatus.BAD_REQUEST` (`400`) and sends `err.message`. packages/common/file-stream/streamable-file.ts:17-31 packages/common/enums/http-status.enum.ts:24-27
- `setErrorHandler(handler)` replaces that callback and returns the same `StreamableFile` instance for chaining. packages/common/file-stream/streamable-file.ts:70-82
- The default `errorLogger` calls `Logger('StreamableFile').error(err)`. `setErrorLogger(handler)` replaces it and also returns the instance. packages/common/file-stream/streamable-file.ts:15 packages/common/file-stream/streamable-file.ts:33-35 packages/common/file-stream/streamable-file.ts:84-91
- In the Express adapter, a source-stream `error` invokes `errorHandler(err, response)`, while an error emitted by the result of `stream.pipe(response)` invokes `errorLogger(err)`. packages/platform-express/adapters/express-adapter.ts:112-118

Used by

4 references from 4 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (4)

- `AppController` — `integration/send-files/src/app.controller.ts`:6
- `AppService` — `integration/send-files/src/app.service.ts`:9
- `ExpressAdapter` — `packages/platform-express/adapters/express-adapter.ts`:51
- `FastifyAdapter` — `packages/platform-fastify/adapters/fastify-adapter.ts`:124