Websockets

July 31, 2026

Websockets

What it is responsible for

The Websockets subsystem manages websocket-oriented work entering through `BaseWsInstance`, `AbstractWsAdapter`, and `MESSAGE_MAPPING_METADATA`. Its documented repository context describes Nest as a framework for building Node.js server-side applications; within that context, this subsystem owns the named websocket entry points rather than an independently documented external dependency. `MESSAGE_MAPPING_METADATA` marks a metadata-based boundary for work entering the subsystem, while the two class-level entry points establish the visible instance and adapter sides of that boundary. The available evidence does not describe transport behavior, connection lifecycle rules, message formats, routing semantics, or external integration details, so those should not be assumed from these names alone.

What it refuses

It rejects activation when `!canActivate`, with `WsException`. It rejects exception-filter input when `!Array.isArray(filters)`, with `InvalidExceptionFilterException`. It rejects a socket port when `!Number.isInteger(port)`, with `InvalidSocketPortException`. No literal exception text is present in the supplied evidence; the exception symbols and predicate conditions are the developers' stated rejection information.

Notable members

- `BaseWsInstance` is a work-entry symbol and carries the instance-facing side of the subsystem's visible boundary.
- `AbstractWsAdapter` is a work-entry symbol and identifies the abstract adapter-facing side of that boundary.
- `MESSAGE_MAPPING_METADATA` is a work-entry constant that identifies the metadata boundary associated with incoming mapped work in this named subsystem.

87 entities in `packages/websockets`. Nothing else in this repository depends on it.

What it is made of

Its 87 entities sit in 36 files under `packages/websockets` : 29 doc comments, 15 classes, 14 constants, 14 functions and 15 more.
`constants.ts` holds 12 of them — more than any other file here.
`WebSocketsController` declares 10 methods, the widest surface here.

Where work enters

- `Readme.md` — `packages/websockets/Readme.md` :1
- `BaseWsInstance` — `packages/websockets/adapters/ws-adapter.ts` :8
- `AbstractWsAdapter` — `packages/websockets/adapters/ws-adapter.ts` :13
- `MESSAGE_MAPPING_METADATA` — `packages/websockets/constants.ts` :3

How it refuses and fails

10 of its components record a refusal or a failure handler.
9 of them refuse work outright, under a condition written into the component itself.
Their `catch` blocks handle a failure that already happened in 1 place.

How this code is named

These conventions cover most of the codebase. Learning them is faster than reading an index —
each one lets you find any member of its family without looking it up.

Pattern	Where	Count	Examples
`*.interface.ts`	across the repository	9	`ws-response.interface.ts` , `nest-gateway.interface.ts` , `on-gateway-init.interface.ts` , `gateway-metadata.interface.ts`
`*.decorator.ts`	`packages/websockets/decorators/`	6	`ack.decorator.ts` , `message-body.decorator.ts` , `gateway-server.decorator.ts` , `socket-gateway.decorator.ts`