



WEBSOCKETS

# AbstractWsAdapter

Kind: Class

VERSION

**47**

UPDATED

**7 August 2026**

DOCUMENT

**reference-subsystem-packages-websockets/class-abstractwsadapter**

## CONTENTS

|          |            |          |          |                        |          |
|----------|------------|----------|----------|------------------------|----------|
| <b>1</b> | Methods    | <b>2</b> | <b>4</b> | Usage                  | <b>3</b> |
| <b>2</b> | Properties | <b>2</b> | <b>5</b> | AI Coding Instructions | <b>4</b> |
| <b>3</b> | Diagram    | <b>3</b> | <b>6</b> | Used by                | <b>4</b> |

**Kind:** Class

**Source:** `packages/websockets/adapters/ws-adapter.ts`

**Part of:** [Websockets](#)

`AbstractWsAdapter` is the base class for WebSocket transport adapters in NestJS. It provides shared lifecycle behavior for client connection/disconnection events and server cleanup, while concrete adapters implement server creation and message-handler binding for a specific WebSocket library.

**Implements:** `WebSocketAdapter`

## 1 Methods

| Method                            | Signature                                                                                                                           | Returns              |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| <code>bindClientConnect</code>    | <code>bindClientConnect(server: TServer, callback: Function)</code>                                                                 | <code>void</code>    |
| <code>bindClientDisconnect</code> | <code>bindClientDisconnect(client: TClient, callback: Function)</code>                                                              | <code>void</code>    |
| <code>close</code>                | <code>close(server: TServer)</code>                                                                                                 | <code>void</code>    |
| <code>dispose</code>              | <code>dispose()</code>                                                                                                              | <code>void</code>    |
| <code>create</code>               | <code>create(port: number, options: TOptions)</code>                                                                                | <code>TServer</code> |
| <code>bindMessageHandlers</code>  | <code>bindMessageHandlers(client: TClient, handlers: WsMessageHandler[], transform: (data: any) =&gt; Observable&lt;any&gt;)</code> | <code>void</code>    |

## 2 Properties

| Property                | Type             |
|-------------------------|------------------|
| <code>httpServer</code> | <code>any</code> |

## 3 Diagram

```
graph LR
  App[Nest Application] --> Adapter[AbstractWsAdapter]
  Adapter --> Create[create()]
  Adapter --> Connect[bindClientConnect()]
  Adapter --> Disconnect[bindClientDisconnect()]
  Adapter --> Messages[bindMessageHandlers()]
  Adapter --> Close[close() / dispose()]
  Create --> Server[WebSocket Server]
  Server --> Client[WebSocket Client]
  Client --> Messages
```

## 4 Usage

```
import { AbstractWsAdapter } from '@nestjs/websockets';

class CustomWsAdapter extends AbstractWsAdapter<MyServer, MyClient> {
  create(port: number): MyServer {
    return createMyWebSocketServer(port);
  }

  bindMessageHandlers(client: MyClient, handlers, transform) {
    client.on('message', async rawMessage => {
      const message = JSON.parse(rawMessage.toString());

      for (const handler of handlers) {
        if (handler.message === message.event) {
          const response = await transform(
            handler.callback(message.data, client),
          );

          client.send(JSON.stringify(response));
        }
      }
    });
  }
}

// main.ts
async function bootstrap() {
  const app = await NestFactory.create(AppModule);

  app.useWebSocketAdapter(new CustomWsAdapter(app));
  await app.listen(3000);
}
```

## 5 AI Coding Instructions

- Extend `AbstractWsAdapter` rather than instantiating it directly; implement `create()` and `bindMessageHandlers()` for the chosen WebSocket library.
- Use `bindClientConnect()` and `bindClientDisconnect()` to attach lifecycle callbacks instead of duplicating connection event wiring.
- Ensure incoming messages are parsed and routed to the matching Nest gateway handler in `bindMessageHandlers()`.
- Always support adapter cleanup through `close()` and `dispose()`, especially when the underlying server keeps open client connections.
- Register the concrete adapter with `app.useWebSocketAdapter()` during application bootstrap.

## 6 Used by

2 references from 2 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

### Imported by (2)

- `IoAdapter` — `packages/platform-socket.io/adapters/io-adapter.ts`:14
- `WsAdapter` — `packages/platform-ws/adapters/ws-adapter.ts`:39