SocketModule

August 17, 2026

SocketModule

Kind: Class

Source: `packages/websockets/socket-module.ts`

Part of: Websockets

`SocketModule` coordinates WebSocket gateway registration and connection lifecycle management. It registers available gateways, connects them to the server, and provides a unified `close()` method for shutting down active socket resources.

Methods

Method	Signature	Returns
`register`	`register(container: NestContainer, applicationConfig: ApplicationConfig, graphInspector: GraphInspector, appOptions: TAppOptions, httpServer: THttpServer)`	`void`
`connectAllGateways`	`connectAllGateways(providers: Map<InjectionToken, InstanceWrapper<Injectable>>, moduleName: string)`	`void`
`connectGatewayToServer`	`connectGatewayToServer(wrapper: InstanceWrapper<Injectable>, moduleName: string)`	`void`
`close`	`close()`	`Promise<any>`

Where it refuses work

- `SocketModule` stops the work with an early return when `!metadataKeys.includes(GATEWAY_METADATA)`.
- `SocketModule` stops the work with an early return when `!this.applicationConfig`.

- `SocketModule` stops the work with an early return when `!adapter`.

Diagram

```
graph LR
  App[Application Bootstrap] --> Module[SocketModule]
  Module --> Register[register()]
  Register --> Gateways[WebSocket Gateways]
  Module --> ConnectAll[connectAllGateways()]
  ConnectAll --> ConnectGateway[connectGatewayToServer()]
  ConnectGateway --> Server[WebSocket Server]
  Module --> Close[close()]
  Close --> Gateways
```

Usage

```
import { SocketModule } from './packages/websockets/socket-module';

// Create the module using the dependencies required by your application.
const socketModule = new SocketModule(/* gateway and server dependencies */);

// Register configured gateways and establish their server connections.
socketModule.register();
socketModule.connectAllGateways();

// During application shutdown, close socket connections cleanly.
async function shutdown() {
  await socketModule.close();
}
```

AI Coding Instructions

- Call `register()` before attempting to connect gateways so gateway definitions and handlers are available.
- Use `connectAllGateways()` for standard application startup; use `connectGatewayToServer()` when connecting an individual gateway as part of custom orchestration.
- Always await `close()` during shutdown to release WebSocket connections and related resources cleanly.
- Keep gateway-specific behavior inside gateway implementations; `SocketModule` should remain responsible for registration and connection lifecycle coordination.

- Ensure server dependencies are initialized before invoking gateway connection methods.

How it works

`SocketModule` is the WebSocket gateway lifecycle coordinator. It scans Nest container providers for classes marked with the `websockets:is_gateway` metadata key, initializes a WebSocket adapter when it finds the first such gateway, and delegates gateway attachment to `WebSocketsController`. packages/websockets/socket-module.ts:68-95 The `@WebSocketGateway()` decorator writes that marker together with gateway port and options metadata. packages/websockets/decorators/socket-gateway.decorator.ts:20-29

- `register(container, applicationConfig, graphInspector, appOptions, httpServer?)` stores the application configuration, application options, and optional HTTP server; creates a WebSocket execution-context creator; creates `SocketServerProvider` and `WebSocketsController` instances; then visits the providers in every container module. packages/websockets/socket-module.ts:39-66
- During that scan, it ignores missing wrappers and wrappers flagged as `isNotMetatype`; every remaining wrapper is checked for the gateway metadata key on its metatype. Non-gateway providers are skipped. packages/websockets/socket-module.ts:68-85
- The first detected gateway initializes the adapter. If `ApplicationConfig` already has an IO adapter, the module sets that adapter's `forceCloseConnections` property from application options. Otherwise, it dynamically loads `@nestjs/platform-socket.io`, constructs its `IoAdapter` with the optional HTTP server, sets the same property, and saves the adapter in `ApplicationConfig`. packages/websockets/socket-module.ts:116-136 If the default adapter package cannot be loaded, `loadAdapter` logs an error and calls `process.exit(1)`. packages/core/helpers/load-adapter.ts:11-21
- For each marked gateway, it calls `WebSocketsController.connectGatewayToServer()` with the gateway instance, its class, module name, and wrapper ID. packages/websockets/socket-module.ts:86-95 That controller reads the gateway's port and options metadata and throws `InvalidSocketPortException` when the port is not an integer. packages/websockets/web-sockets-controller.ts:45-64
- The context creator assembled by the module includes WebSocket exception filters, pipes, guards, and interceptors. packages/websockets/socket-module.ts:138-149 The controller wraps discovered message handlers through

that context creator, records WebSocket entrypoint definitions in the graph inspector, and, unless preview mode is enabled, creates or finds a socket server, assigns it to server-decorated gateway properties, and subscribes handlers. packages/websockets/web-sockets-controller.ts:73-104

- Server instances are tracked by socket configuration. The server provider reuses an existing host with the same port and path, creates a new adapter server when none exists, and creates separately tracked namespace hosts when a namespace is configured. packages/websockets/socket-server-provider.ts:14-32 packages/websockets/socket-server-provider.ts:34-73
- `close()` is a no-op when registration has not set `applicationConfig`, or when that configuration has no adapter. Otherwise, it calls `adapter.close(server)` concurrently for every tracked host whose `server` value is truthy, awaits `adapter.dispose()`, and clears the tracked hosts. packages/websockets/socket-module.ts:97-114
- Nest applications invoke `register()` while registering modules, passing the container, application config, graph inspector, app options, and HTTP server. packages/core/nest-application.ts:140-161 packages/core/nest-application.ts:164-176 Application disposal awaits `SocketModule.close()` before closing the HTTP adapter. packages/core/nest-application.ts:98-101

Relationships

- IMPORTS → `NestApplicationOptions`
- IMPORTS → `InjectionToken`
- IMPORTS → `Injectable`
- IMPORTS → `NestApplicationContextOptions`
- IMPORTS → `ApplicationConfig`
- IMPORTS → `GuardsConsumer`
- IMPORTS → `GuardsContextCreator`
- IMPORTS → `loadAdapter`
- IMPORTS → `NestContainer`
- IMPORTS → `InstanceWrapper`
- IMPORTS → `GraphInspector`
- IMPORTS → `InterceptorsConsumer`
- IMPORTS → `InterceptorsContextCreator`
- IMPORTS → `PipesConsumer`
- IMPORTS → `PipesContextCreator`

- IMPORTS → `-nestjs-platform-socket-io`

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)

- `NestApplication` — `packages/core/nest-application.ts`:54
- `NestMicroservice` — `packages/microservices/nest-microservice.ts`:35