WebSocketsController

August 18, 2026

WebSocketsController

Kind: Class

Source: `packages/websockets/web-sockets-controller.ts`

Part of: Websockets

`WebSocketsController` is the framework-level coordinator that discovers WebSocket gateway entrypoints and connects them to the configured socket server. It subscribes gateway lifecycle hooks (`init` , `connection` , `disconnect`) and message handlers, then normalizes handler results into observables for transport delivery.

Methods

Method	Signature	Returns
`connectGatewayToServer`	`connectGatewayToServer(instance: NestGateway, metatype: Type	Function, moduleKey: string, instanceWrapperId: string)`
`subscribeToServerEvents`	`subscribeToServerEvents(instance: NestGateway, options: T, port: number, moduleKey: string, instanceWrapperId: string)`	`void`
`subscribeEvents`	`subscribeEvents(instance: NestGateway, subscribersMap: MessageMappingProperties[], observableServer: ServerAndEventStreamsHost)`	`void`
`getConnectionHandler`	`getConnectionHandler(context: WebSocketsController, instance: NestGateway, subscribersMap: MessageMappingProperties[], disconnect: Subject<any>, connection: Subject<any>)`	`void`
`subscribeInitEvent`	`subscribeInitEvent(instance: NestGateway, event: Subject<any>)`	`void`

Method	Signature	Returns
`subscribeConnectionEvent`	`subscribeConnectionEvent(instance: NestGateway, event: Subject<any>)`	`void`
`subscribeDisconnectEvent`	`subscribeDisconnectEvent(instance: NestGateway, event: Subject<any>)`	`void`
`subscribeMessages`	`subscribeMessages(subscribersMap: MessageMappingProperties[], client: T, instance: NestGateway)`	`void`
`pickResult`	`pickResult(deferredResult: Promise<any>)`	`Promise<Observable<any>>`
`inspectEntrypointDefinitions`	`inspectEntrypointDefinitions(instance: NestGateway, port: number, messageHandlers: MessageMappingProperties[], instanceWrapperId: string)`	`void`

Where it refuses work

- `WebSocketsController` stops the work with `InvalidSocketPortException` when `!Number.isInteger(port)`.
- `WebSocketsController` stops the work with an early return when `this.appOptions.preview`.
- `WebSocketsController` stops the work with an early return when `isObservable(result)`.
- `WebSocketsController` stops the work with an early return when `result instanceof Promise`.
- `WebSocketsController` stops the work with an early return when `!gatewayClassName`.

Diagram

```
graph LR
  A[Gateway Provider] --> B[WebSocketsController]
  B --> C[inspectEntrypointDefinitions]
  B --> D[connectGatewayToServer]
  D --> E[Socket Server]
  B --> F[subscribeToServerEvents]
  F --> G[subscribeInitEvent]
  F --> H[subscribeConnectionEvent]
  F --> I[subscribeDisconnectEvent]
  F --> J[subscribeMessages]
  J --> K[getConnectionHandler]
  K --> L[pickResult]
  L --> M[Observable Response]
  E --> N[Connected Clients]
```

Usage

```
import {
  SubscribeMessage,
  WebSocketGateway,
  WebSocketServer,
} from '@nestjs/websockets';
import { Server, Socket } from 'socket.io';

@WebSocketGateway({ namespace: '/chat' })
export class ChatGateway {
  @WebSocketServer()
  server: Server;

  handleConnection(client: Socket) {
    console.log(`Client connected: ${client.id}`);
  }

  handleDisconnect(client: Socket) {
    console.log(`Client disconnected: ${client.id}`);
  }

  @SubscribeMessage('chat:message')
  handleMessage(client: Socket, payload: { text: string }) {
    return {
      event: 'chat:message',
      data: {
        clientId: client.id,
        text: payload.text,
      },
    };
  }
}

// WebSocketsController is created by Nest internally during application startup.
// It discovers ChatGateway, attaches it to the Socket.IO server, subscribes to
// lifecycle events, and routes "chat:message" events to handleMessage().
```

AI Coding Instructions

- Treat `WebSocketsController` as framework infrastructure; applications should define gateways and message handlers rather than instantiate this class directly.
- Keep gateway event handlers compatible with the result pipeline: return values that can be converted into observable WebSocket responses when appropriate.
- When adding lifecycle behavior, preserve the separate initialization, connection, and disconnect subscription paths.
- Ensure new gateway entrypoint metadata can be discovered by `inspectEntrypointDefinitions()` and connected through `connectGatewayToServer()`.

- Avoid subscribing message handlers more than once for the same gateway/server pair, as this can cause duplicated client responses.

How it works

Role

`WebSocketsController` is the runtime coordinator that connects a gateway instance to a WebSocket server, discovers its message-mapped methods, creates their WebSocket execution contexts, registers lifecycle callbacks, and binds client connections and message handlers through the configured I/O adapter. packages/websockets/web-sockets-controller.ts:29-43 packages/websockets/web-sockets-controller.ts:66-127

`SocketModule` constructs this controller during WebSocket-module registration and calls `connectGatewayToServer()` only for provider metatypes carrying `GATEWAY_METADATA`. packages/websockets/socket-module.ts:50-65 packages/websockets/socket-module.ts:77-94

Gateway connection and validation

- `connectGatewayToServer(instance, metatype, moduleKey, instanceWrapperId)` reads gateway options and port metadata from the gateway metatype; missing metadata becomes `{}` and `0`, respectively. packages/websockets/web-sockets-controller.ts:45-53
- The port must be an integer. A non-integer value throws `InvalidSocketPortException`, whose message is `Invalid port (<port>) in gateway <type>`. packages/websockets/web-sockets-controller.ts:54-56 packages/websockets/errors/invalid-socket-port.exception.ts:3-6
- After validation, it delegates gateway setup to `subscribeToServerEvents()` with the resolved metadata, module key, and wrapper ID. packages/websockets/web-sockets-controller.ts:57-63

Message mappings and entrypoint inspection

- `subscribeToServerEvents()` scans the gateway prototype for methods marked with `MESSAGE_MAPPING_METADATA`; for each, it records the message metadata, method name, callback, and whether parameter metadata contains an ACK parameter. packages/websockets/web-sockets-controller.ts:73-86 packages/websockets/gateway-metadata-explorer.ts:26-57 packages/websockets/gateway-metadata-explorer.ts:60-79
- Each discovered callback is replaced with a function created by `WsContextCreator` using the gateway instance, original callback, module key, and method name. packages/websockets/web-sockets-controller.ts:74-85

- The resulting wrapper builds a WebSocket context, obtains exception filters, pipes, guards, and interceptors, runs guards, applies pipes when parameter metadata exists, and invokes the original callback through an exception-handling proxy. A denied guard throws `WsException` with the guards package's forbidden-message constant. packages/websockets/context/ws-context-creator.ts:57-129 packages/websockets/context/ws-context-creator.ts:143-162
- For every mapped method, `inspectEntrypointDefinitions()` inserts a graph entrypoint definition of type `websocket`, containing the method name, gateway constructor name, wrapper ID, port, and message as both `key` and `message`. packages/websockets/web-sockets-controller.ts:88-93 packages/websockets/web-sockets-controller.ts:203-225

Preview mode and server setup

- Entrypoint inspection occurs before the preview-mode check. When `appOptions.preview` is truthy, setup returns without locating or creating a socket server, assigning server properties, or binding events. packages/websockets/web-sockets-controller.ts:88-104
- Outside preview mode, the controller asks `SocketServerProvider` for a server keyed by port and path, assigns that server to gateway instance properties marked with `GATEWAY_SERVER_METADATA`, then subscribes gateway events. packages/websockets/web-sockets-controller.ts:98-104 packages/websockets/web-sockets-controller.ts:227-236
- The server provider reuses a matching stored server when possible; otherwise it calls the configured adapter's `create(port, partialOptions)`, stores the resulting server/event-stream host, and creates a namespace-specific host when gateway metadata has a namespace. packages/websockets/socket-server-provider.ts:14-32 packages/websockets/socket-server-provider.ts:34-55

Lifecycle events and client connections

- `subscribeEvents()` subscribes optional gateway lifecycle methods— `afterInit`, `handleConnection`, and `handleDisconnect` —to the server host's `init`, `connection`, and `disconnect` subjects, then registers a connection callback through `adapter.bindClientConnect(server, handler)`. packages/websockets/web-sockets-controller.ts:106-127 packages/websockets/interfaces/nest-gateway.interface.ts:4-8
- The server host's factory immediately emits its server through `init`, and creates subjects for connection and disconnect events. packages/websockets/factories/server-and-event-streams-factory.ts:4-17
- Connection lifecycle handling suppresses consecutive connection notifications whose first arguments are the same reference before invoking `handleConnection(...args)`. packages/websockets/web-sockets-controller.ts:154-162 packages/websockets/utils/compare-element.util.ts:1-7

- Disconnect lifecycle handling suppresses consecutive duplicate disconnect values before invoking `handleDisconnect`. packages/websockets/web-sockets-controller.ts:164-170
- When the adapter invokes the registered connection handler, it emits the complete connection argument list to the connection subject, binds message mappings for the first argument as the client, and, if the adapter implements `bindClientDisconnect`, emits that client to the disconnect subject when the adapter's disconnect callback runs. packages/websockets/web-sockets-controller.ts:129-146
- After binding, it logs one line per mapped message in the form `<GatewayClass> subscribed to the "<message>" message`; it emits no such logs when the instance has no constructor name. packages/websockets/web-sockets-controller.ts:238-251

Message invocation and return values

- For each connected client, `subscribeMessages()` binds every discovered handler through `adapter.bindMessageHandlers`; each callback is bound with the gateway as `this` and the client as its first argument. packages/websockets/web-sockets-controller.ts:172-188
- The transform passed to the adapter awaits the callback's deferred result, leaves observables unchanged, converts promises to observables, and wraps all other values in a one-value observable; `mergeAll()` then flattens the resulting observable. packages/websockets/web-sockets-controller.ts:185-201
- The adapter contract requires client-connect binding and message-handler binding; client-disconnect binding is optional. packages/common/interfaces/websockets/web-socket-adapter.interface.ts:15-29

Relationships

- IMPORTS → `NestApplicationContextOptions`
- IMPORTS → `Type`
- IMPORTS → `Logger`
- IMPORTS → `ApplicationConfig`
- IMPORTS → `GraphInspector`
- IMPORTS → `MetadataScanner`