ServerAndEventStreamsHost

August 17, 2026

ServerAndEventStreamsHost

Kind: Interface

Source: `packages/websockets/interfaces/server-and-event-streams-host.interface.ts`

Part of: Websockets

`ServerAndEventStreamsHost<T>` defines a shared host for a server instance and its lifecycle event streams. It exposes the server object along with RxJS subjects for initialization, client connections, and disconnections, allowing websocket integrations to publish and observe server events consistently.

Properties

Property	Type
`server`	`T`
`init`	`ReplaySubject<T>`
`connection`	`Subject<any>`
`disconnect`	`Subject<any>`

Diagram

```
graph LR
  Host[ServerAndEventStreamsHost<T>]
  Server[T server]
  Init[ReplaySubject<T><br/>init]
  Connection[Subject<any><br/>connection]
  Disconnect[Subject<any><br/>disconnect]

  Host --> Server
  Host --> Init
  Host --> Connection
  Host --> Disconnect
```

```
  Init --> Consumers[Server lifecycle consumers]
  Connection --> Consumers
  Disconnect --> Consumers
```

Usage

```ts
import { ReplaySubject, Subject } from 'rxjs';
import type { ServerAndEventStreamsHost } from './server-and-event-streams-host.interface';

interface WebSocketServer {
  close(): void;
}

const websocketServer: WebSocketServer = {
  close() {
    console.log('Server closed');
  },
};

const host: ServerAndEventStreamsHost<WebSocketServer> = {
  server: websocketServer,
  init: new ReplaySubject<WebSocketServer>(1),
  connection: new Subject<any>(),
  disconnect: new Subject<any>(),
};

// Notify subscribers that the server is ready.
host.init.next(host.server);

// Publish connection lifecycle events.
host.connection.subscribe((client) => {
  console.log('Client connected:', client.id);
});

host.disconnect.subscribe((client) => {
  console.log('Client disconnected:', client.id);
});

host.connection.next({ id: 'client-1' });
host.disconnect.next({ id: 'client-1' });
```

AI Coding Instructions

- Use `init` as a `ReplaySubject<T>` so late subscribers can receive the initialized server instance.

- Emit the server instance through `init.next(server)` only after the server is fully configured and ready to accept connections.
- Publish connection and disconnection payloads through the corresponding subjects; keep payload shapes consistent across adapters.
- Prefer strongly typed connection payloads over `any` when extending or consuming this interface.
- Clean up subject subscriptions and complete streams during server shutdown when the owning lifecycle supports it.

How it works

`ServerAndEventStreamsHost<T>`

`ServerAndEventStreamsHost<T>` is a public exported TypeScript interface for a websocket server value and three RxJS event streams. Its generic server type defaults to `any`. [server-and-event-streams-host.interface.ts:3-10]

Member	Type	Observed role
`server`	`T`	Holds the underlying server object. [server-and-event-streams-host.interface.ts:6-8] The websocket controller passes it to the adapter’s client-connect binding and assigns it to gateway properties marked as server hooks. [web-sockets-controller.ts:111-126] [web-sockets-controller.ts:227-235]
`init`	`ReplaySubject<T>`	Carries server initialization values. [server-and-event-streams-host.interface.ts:8] The factory creates this subject and immediately emits the supplied `server` value. [server-and-event-streams-factory.ts:5-7] When a gateway has `afterInit`, the controller subscribes that hook to this stream. [web-sockets-controller.ts:148-152]
`connection`	`Subject<any>`	Carries client connection argument arrays. [server-and-event-streams-host.interface.ts:9] The controller emits the adapter connection handler’s complete argument list to it, then invokes `handleConnection` with those arguments when that hook exists; duplicate consecutive connections with the same first argument are filtered before the hook call. [web-sockets-controller.ts:129-145] [web-sockets-controller.ts:154-161]
`disconnect`	`Subject<any>`	Carries disconnected client values. [server-and-event-streams-host.interface.ts:10] If the adapter exposes

Member	Type	Observed role
		`bindClientDisconnect`, the connection handler registers a callback that emits the client to this subject. [web-sockets-controller.ts:142-145] When a gateway has `handleDisconnect`, the controller subscribes that hook after `distinctUntilChanged()`. [web-sockets-controller.ts:164-169]