ServerTCP

August 18, 2026

ServerTCP

Kind: Class

Source: `packages/microservices/server/server-tcp.ts`

Part of: Microservices

`ServerTCP` is the TCP transport server implementation for NestJS microservices. It opens a TCP socket, registers message handlers by pattern, routes incoming packets to those handlers, and manages listening, error, and connection-close lifecycle events.

Extends: `Server`

Methods

Method	Signature	Returns
`listen`	`listen(callback: (err?: unknown, ...optionalParams: unknown[]) => void)`	`void`
`close`	`close()`	`void`
`bindHandler`	`bindHandler(socket: Socket)`	`void`
`handleMessage`	`handleMessage(socket: TcpSocket, rawMessage: unknown)`	`void`
`handleClose`	`handleClose()`	`undefined
`unwrap`	`unwrap()`	`T`
`on`	`on(event: EventKey, callback: EventCallback)`	`void`
`init`	`init()`	`void`
`registerListeningListener`	`registerListeningListener(socket: net.Server)`	`void`
`registerErrorListener`	`registerErrorListener(socket: net.Server)`	`void`

Method	Signature	Returns
`registerCloseListener`	`registerCloseListener(socket: net.Server)`	`void`
`getSocketInstance`	`getSocketInstance(socket: Socket)`	`TcpSocket`

Properties

Property	Type
`transportId`	`TransportId`
`server`	`NetSocket`
`port`	`number`
`host`	`string`
`socketClass`	`Type<TcpSocket>`
`maxBufferSize`	`number`
`isManuallyTerminated`	`any`
`retryAttemptsCount`	`any`
`tlsOptions`	`TlsOptions`
`pendingEventListeners`	`Array<{ event: keyof TcpEvents; callback: TcpEvents[keyof TcpEvents]; }>`

Where it refuses work

- `ServerTCP` stops the work with `Error` when `!this.server` — “Not initialized. Please call the "listen"/"startAllMicroservices" method before accessing…”.
- `ServerTCP` stops the work with an early return when `isUndefined((packet as IncomingRequest).id)`.
- `ServerTCP` stops the work with an early return when `this.isManuallyTerminated || !this.getOptionsProp(this.options, 'retryAttempts') || this.⋯`.
- `ServerTCP` stops the work with an early return when `this.maxBufferSize !== undefined && this.socketClass === JsonSocket`.

Diagram

```
graph LR
  Client[TCP Client] -->|Serialized message| Socket[TCP Socket]
```

```
  Socket --> ServerTCP[ServerTCP]
  ServerTCP -->|handleMessage| HandlerRegistry[Pattern Handler Registry]
  HandlerRegistry --> Handler[Message Handler]
  Handler -->|Response / Error| ServerTCP
  ServerTCP -->|Serialized response| Client
```

Usage

```ts
import { ServerTCP } from '@nestjs/microservices';

const server = new ServerTCP({
  host: '127.0.0.1',
  port: 3001,
});

server.bindHandler(
  { cmd: 'sum' },
  async (data: number[]) => data.reduce((total, value) => total + value, 0),
);

server.listen(() => {
  console.log('TCP microservice listening on port 3001');
});

// Shut down gracefully when the process stops.
process.on('SIGTERM', async () => {
  await server.close();
});
```

AI Coding Instructions

- Register handlers with `bindHandler()` before calling `listen()` so incoming patterns can be resolved immediately.
- Keep handler patterns stable and serializable; TCP clients must send a matching pattern for `handleMessage()` to dispatch correctly.
- Use `listen()` and `close()` for lifecycle management rather than interacting with the underlying TCP server directly.
- Preserve error and listening listener registration during changes; `registerErrorListener()` and `registerListeningListener()` protect startup and connection lifecycle behavior.
- Use `unwrap()` only when transport-specific access to the underlying Node.js TCP server is necessary.

How it works

`ServerTCP` is the TCP transport server implementation. It extends the shared `Server` base class, identifies itself as `Transport.TCP`, and creates either a Node TCP server or a TLS server to accept connections. `packages/microservices/server/server-tcp.ts:33-43` `packages/microservices/server/server-tcp.ts:172-185`

Relationships

- IMPORTS → `Type`
- IMPORTS → `isString`
- IMPORTS → `isUndefined`