TcpOptions

August 17, 2026

TcpOptions

Kind: Interface

Source: `packages/microservices/interfaces/microservice-configuration.interface.ts`

Part of: Microservices

`TcpOptions` configures a NestJS microservice that communicates over the TCP transport. It defines the required `transport` type and optional connection, retry, serialization, TLS, socket, and buffer settings used when creating the microservice.

Properties

Property	Type
`transport`	`Transport.TCP`
`options`	`{ host?: string; port?: number; retryAttempts?: number; retryDelay?: number; serializer?: Serializer; tlsOptions?: TlsOptions; deserializer?: Deserializer; socketClass?: Type<TcpSocket>; maxBufferSize?: number; }`

Diagram

```
graph LR
  A[Microservice Configuration] --> B[TcpOptions]
  B --> C[transport: Transport.TCP]
  B --> D[options]
  D --> E[host and port]
  D --> F[retryAttempts and retryDelay]
  D --> G[serializer and deserializer]
  D --> H[tlsOptions]
  D --> I[socketClass]
  D --> J[maxBufferSize]
  E --> K[TCP Microservice Server]
```

Usage

```
import { Transport, type TcpOptions } from '@nestjs/microservices';

const tcpConfig: TcpOptions = {
  transport: Transport.TCP,
  options: {
    host: '127.0.0.1',
    port: 3001,
    retryAttempts: 5,
    retryDelay: 1000,
    maxBufferSize: 1024 * 1024,
  },
};

// Example: NestFactory.createMicroservice(AppModule, tcpConfig)
```

AI Coding Instructions

- Always set `transport` to `Transport.TCP`; this interface is specific to TCP-based microservices.
- Configure both `host` and `port` explicitly for predictable local, container, and production networking behavior.
- Use `retryAttempts` and `retryDelay` when the application must reconnect to TCP dependencies that may start later.
- Provide matching `serializer` and `deserializer` implementations when communicating with services that use custom message formats.
- Set `tlsOptions`, `socketClass`, or `maxBufferSize` only when required by the deployment environment or protocol requirements.

How it works

`TcpOptions` is a public TypeScript interface for configuring a TCP microservice server. It is one member of the `MicroserviceOptions` union, and its optional `transport` discriminator is `Transport.TCP`. Its `options` object is also optional. microservice-configuration.interface.ts:25-33 microservice-configuration.interface.ts:98-118

The interface itself has no executable validation: every declared property, including `options`, is optional. microservice-configuration.interface.ts:101-117

When the server factory receives a non-custom configuration whose transport is not one of the explicitly handled non-TCP transports, it constructs `ServerTCP` with this

interface's `options` object. server-factory.ts:21-41