GrpcOptions

August 19, 2026

GrpcOptions

Kind: Interface

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

Part of: Microservices

`GrpcOptions` configures a NestJS microservice that uses the `Transport.GRPC` transport. It defines how protobuf packages are loaded, how the gRPC server listens for requests, and optional channel, message-size, keepalive, credential, and shutdown behavior.

Properties

Property	Type
`transport`	`Transport.GRPC`
`options`	`{ url?: string; maxSendMessageLength?: number; maxReceiveMessageLength?: number; maxMetadataSize?: number; keepalive?: { keepaliveTimeMs?: number; keepaliveTimeoutMs?: number; keepalivePermitWithoutCalls?: number; http2MaxPingsWithoutData?: number; http2MinTimeBetweenPingsMs?: number; http2MinPingIntervalWithoutDataMs?: number; http2MaxPingStrikes?: number; }; channelOptions?: ChannelOptions; credentials?: any; protoPath?: string`

Diagram

```
graph LR
  App[Nest Microservice] --> Config[GrpcOptions]
  Config --> Transport[Transport.GRPC]
  Config --> Endpoint[url]
  Config --> Proto[protoPath]
  Config --> Package[package]
  Config --> Loader[loader / protoLoader]
  Config --> Runtime[channelOptions & credentials]
  Config --> Connection[keepalive settings]
```

```
  Config --> Lifecycle[gracefulShutdown & onLoadPackageDefinition]
  Proto --> Service[gRPC Service Definitions]
  Package --> Service
```

Usage

```ts
import { NestFactory } from '@nestjs/core';
import { Transport, type MicroserviceOptions } from '@nestjs/microservices';
import { AppModule } from './app.module';
import { join } from 'path';

async function bootstrap() {
  const app = await NestFactory.createMicroservice<MicroserviceOptions>(
    AppModule,
    {
      transport: Transport.GRPC,
      options: {
        url: '0.0.0.0:50051',
        protoPath: join(__dirname, 'proto', 'users.proto'),
        package: 'users',
        loader: {
          keepCase: true,
          longs: String,
          enums: String,
          defaults: true,
          oneofs: true,
        },
        maxSendMessageLength: 4 * 1024 * 1024,
        maxReceiveMessageLength: 4 * 1024 * 1024,
        keepalive: {
          keepaliveTimeMs: 30_000,
          keepaliveTimeoutMs: 10_000,
          keepalivePermitWithoutCalls: true,
        },
        gracefulShutdown: true,
      },
    },
  );

  await app.listen();
}

bootstrap();
```

AI Coding Instructions

- Always set `transport: Transport.GRPC`; this interface is only valid for gRPC microservice configurations.

- Provide both `protoPath` and `package`; use arrays when loading multiple protobuf files or packages.
- Use `loader` options consistently between gRPC clients and servers, especially `keepCase`, `longs`, `enums`, and `defaults`.
- Configure `maxSendMessageLength` and `maxReceiveMessageLength` when services exchange large payloads; verify limits match connected clients.
- Enable `gracefulShutdown` for production services and use `onLoadPackageDefinition` only when custom access to the loaded package definition is required.

Used by

1 reference from 1 file. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (1)

- `grpcClientOptions` — `sample/04-grpc/src/grpc-client.options.ts`:5