RmqOptions

August 18, 2026

RmqOptions

Kind: Interface

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

Part of: Microservices

`RmqOptions` configures a NestJS microservice that uses the RabbitMQ (`Transport.RMQ`) transport. It defines connection URLs, queue and exchange behavior, consumer acknowledgement settings, message serialization, routing, and connection retry behavior.

Properties

Property	Type
`transport`	`Transport.RMQ`
`options`	`{ urls?: string[]

Diagram

```
graph LR
  App[NestJS Application] --> Config[RmqOptions]
  Config --> Transport[Transport.RMQ]
  Config --> Connection[AMQP Connection URLs]
  Config --> Queue[Queue Configuration]
  Config --> Exchange[Exchange and Routing]
  Config --> Consumer[Consumer Settings]
  Config --> Serialization[Serializer / Deserializer]

  Connection --> RabbitMQ[RabbitMQ Broker]
  Queue --> RabbitMQ
  Exchange --> RabbitMQ
  Consumer --> RabbitMQ
```

Usage

```
import { NestFactory } from '@nestjs/core';
import { Transport, RmqOptions } from '@nestjs/microservices';
import { AppModule } from './app.module';

async function bootstrap() {
  const rmqOptions: RmqOptions = {
    transport: Transport.RMQ,
    options: {
      urls: ['amqp://guest:guest@localhost:5672'],
      queue: 'orders',
      queueOptions: {
        durable: true,
      },
      prefetchCount: 10,
      noAck: false,
      persistent: true,
      exchange: 'events',
      exchangeType: 'topic',
      routingKey: 'orders.created',
      maxConnectionAttempts: 5,
    },
  };

  const app = await NestFactory.createMicroservice(AppModule, rmqOptions);
  await app.listen();
}

bootstrap();
```

AI Coding Instructions

- Always set `transport: Transport.RMQ`; this interface is specifically for RabbitMQ microservice configuration.
- Configure a durable `queue` and `queueOptions.durable: true` when messages must survive broker restarts.
- Set `noAck: false` for reliable processing, then explicitly acknowledge messages in handlers after successful work.
- Use `exchange`, `exchangeType`, and `routingKey` together when implementing publish/subscribe or topic-based routing.
- Provide custom `serializer` and `deserializer` only when all producers and consumers agree on the message format.