MqttOptions

August 17, 2026

MqttOptions

Kind: Interface

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

Part of: Microservices

`MqttOptions` configures a NestJS microservice that communicates through the MQTT transport. It combines standard MQTT client settings with Nest-specific serialization, deserialization, subscription, and MQTT v5 user-property options.

Properties

Property	Type
`transport`	`Transport.MQTT`
`options`	`MqttClientOptions & { url?: string; serializer?: Serializer; deserializer?: Deserializer; subscribeOptions?: { qos: QoS; nl?: boolean; rap?: boolean; rh?: number; }; userProperties?: Record<string, string`

Diagram

```
graph LR
  A[Microservice Configuration] --> B[MqttOptions]
  B --> C[transport: Transport.MQTT]
  B --> D[options]
  D --> E[MqttClientOptions]
  D --> F[url]
  D --> G[serializer / deserializer]
  D --> H[subscribeOptions]
  D --> I[userProperties]
  H --> J[qos, nl, rap, rh]
  B --> K[MQTT Broker]
```

Usage

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

const mqttConfig: MqttOptions = {
  transport: Transport.MQTT,
  options: {
    url: 'mqtt://localhost:1883',
    clientId: 'orders-service',
    clean: true,

    subscribeOptions: {
      qos: 1,
      nl: false,
      rap: true,
      rh: 0,
    },

    userProperties: {
      service: 'orders',
      environment: 'production',
    },
  },
};

// Example integration:
// NestFactory.createMicroservice(AppModule, mqttConfig);
```

AI Coding Instructions

- Always set `transport` to `Transport.MQTT`; this interface is specifically for MQTT microservice configurations.
- Use `options.url` for the broker connection string, or provide equivalent host, port, and protocol settings supported by `MqttClientOptions`.
- Configure `serializer` and `deserializer` together when using a custom message format so outgoing and incoming payloads remain compatible.
- Set `subscribeOptions.qos` intentionally based on delivery guarantees; higher QoS can increase broker and network overhead.
- Use `userProperties` only when the connected broker and clients support MQTT v5 properties.

How it works

`MqttOptions` is the public TypeScript configuration interface for selecting the MQTT microservice transport. It is one variant of `MicroserviceOptions`; its optional `transport` discriminator is `Transport.MQTT`, and its optional `options` object combines MQTT client connection settings with Nest-specific MQTT settings. microservice-configuration.interface.ts:25-33 microservice-configuration.interface.ts:142-168