KafkaOptions

August 17, 2026

KafkaOptions

Kind: Interface

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

Part of: Microservices

`KafkaOptions` configures a NestJS microservice that communicates through Apache Kafka. It combines Kafka client, consumer, producer, subscription, serialization, and message-parsing settings while enforcing `Transport.KAFKA` as the selected transport.

Properties

Property	Type
`transport`	`Transport.KAFKA`
`options`	`{ postfixId?: string; client?: KafkaConfig; consumer?: ConsumerConfig; run?: Omit<ConsumerRunConfig, 'eachBatch'`

Diagram

```
graph LR
  A[KafkaOptions] --> B[transport: Transport.KAFKA]
  A --> C[options]

  C --> D[client: KafkaConfig]
  C --> E[consumer: ConsumerConfig]
  C --> F[run: ConsumerRunConfig]
  C --> G[subscribe: ConsumerSubscribeTopics]
  C --> H[producer: ProducerConfig]
  C --> I[send: ProducerRecord defaults]
  C --> J[serializer / deserializer]
  C --> K[parser: KafkaParserConfig]
  C --> L[producerOnlyMode]

  D --> M[Kafka Broker]
```

```
  E --> M
  H --> M
```

Usage

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

async function bootstrap() {
  const kafkaOptions: KafkaOptions = {
    transport: Transport.KAFKA,
    options: {
      client: {
        clientId: 'billing-service',
        brokers: ['localhost:9092'],
      },
      consumer: {
        groupId: 'billing-consumer-group',
      },
      subscribe: {
        fromBeginning: false,
      },
      run: {
        autoCommit: true,
      },
      producer: {
        allowAutoTopicCreation: false,
      },
      send: {
        acks: -1,
      },
    },
  };

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

bootstrap();
```

AI Coding Instructions

- Always set `transport` to `Transport.KAFKA`; this interface is specifically for Kafka-based microservices.
- Configure a unique `client.clientId` and stable `consumer.groupId` to avoid conflicts between deployed service instances.

- Do not provide `topics` in `subscribe`, `topic` or `messages` in `send`, or `eachBatch` / `eachMessage` in `run`; NestJS manages these values per handler and message operation.
- Use custom `serializer`, `deserializer`, or `parser` options when integrating with non-Nest producers or consumers that use different payload formats.
- Set `producerOnlyMode: true` for services that publish Kafka messages but do not register Kafka message handlers.