RetryOptions

August 17, 2026

RetryOptions

Kind: Interface

Source: `packages/microservices/external/kafka.interface.ts`

Part of: Microservices

`RetryOptions` configures retry behavior for the Kafka microservice client or consumer connection workflow. It controls retry timing, exponential backoff, retry limits, and whether the service should restart after a terminal failure.

Properties

Property	Type
`maxRetryTime`	`number`
`initialRetryTime`	`number`
`factor`	`number`
`multiplier`	`number`
`retries`	`number`
`restartOnFailure`	`(e: Error) => Promise<boolean>`

Diagram

```
graph LR
  A[Kafka operation fails] --> B[RetryOptions]
  B --> C{Retries remaining?}
  C -->|Yes| D[Wait initialRetryTime<br/>with factor and multiplier]
  D --> E[Retry operation]
  E --> A
  C -->|No| F[restartOnFailure(error)]
```

```
  F -->|true| G[Restart service or connection]
  F -->|false| H[Propagate failure]
```

Usage

```ts
import type { RetryOptions } from '@nestjs/microservices';

const retryOptions: RetryOptions = {
  retries: 5,
  initialRetryTime: 300,
  maxRetryTime: 30_000,
  factor: 0.2,
  multiplier: 2,

  async restartOnFailure(error: Error): Promise<boolean> {
    console.error('Kafka connection failed:', error.message);

    // Restart only for recoverable infrastructure failures.
    return true;
  },
};

// Example: pass the options to Kafka transport configuration.
const kafkaOptions = {
  client: {
    clientId: 'orders-service',
    brokers: ['localhost:9092'],
    retry: retryOptions,
  },
};
```

AI Coding Instructions

- Set `initialRetryTime`, `maxRetryTime`, `factor`, and `multiplier` together so backoff growth remains bounded.
- Use `retries` to prevent infinite retry loops; choose a limit appropriate for the service's availability requirements.
- Keep `restartOnFailure` asynchronous and return `true` only when restarting the Kafka client or process is safe.
- Log or report the received `Error` in `restartOnFailure` to preserve diagnostics for terminal connection failures.
- Reuse a shared retry configuration across Kafka client and consumer setup when consistent recovery behavior is required.

How it works

`RetryOptions` is an exported KafkaJS-facing TypeScript interface. Its source file states that it is intended only to represent KafkaJS package types and must not contain NestJS logic. packages/microservices/external/kafka.interface.ts:1-6

All of its properties are optional:

- `maxRetryTime?: number`
- `initialRetryTime?: number`
- `factor?: number`
- `multiplier?: number`
- `retries?: number`
- `restartOnFailure?: (e: Error) => Promise<boolean>` — an optional callback that receives an `Error` and asynchronously returns a boolean. packages/microservices/external/kafka.interface.ts:253-260

`RetryOptions` can be assigned to the `retry` property of Kafka client configuration, producer configuration, and admin configuration. packages/microservices/external/kafka.interface.ts:60-74 packages/microservices/external/kafka.interface.ts:110-119 packages/microservices/external/kafka.interface.ts:262-264 Consumer configuration also accepts it through `retry`; that declaration additionally declares `restartOnFailure` with the same `Error`-to-`Promise<boolean>` signature. packages/microservices/external/kafka.interface.ts:164-181

In Nest Kafka transport options, these configurations are exposed as `options.client`, `options.consumer`, and `options.producer`. packages/microservices/interfaces/microservice-configuration.interface.ts:333-355 The client transport merges `options.client` into the Kafka constructor configuration, passes `options.consumer` to `client.consumer()`, and passes `options.producer` to `client.producer()`. packages/microservices/client/client-kafka.ts:162-184 The server transport likewise merges `options.client` into the Kafka constructor configuration and passes its consumer and producer options to the respective Kafka client methods. packages/microservices/server/server-kafka.ts:107-117 packages/microservices/server/server-kafka.ts:155-162

This interface itself contains no validation, defaults, error handling, or runtime side effects; it only declares the optional fields and their TypeScript types. packages/microservices/external/kafka.interface.ts:253-260