KafkaRetriableException

August 17, 2026

KafkaRetriableException

Kind: Class

Source: `packages/microservices/exceptions/kafka-retriable-exception.ts`

Part of: Microservices

Exception that instructs Kafka driver to instead of introspecting
error processing flow and sending serialized error message to the consumer,
force bubble it up to the "eachMessage" callback of the underlying "kafkajs" package
(even if interceptors are applied, or an observable stream is returned from the
message handler).

A transient exception that if retried may succeed.

`KafkaRetriableException` signals that Kafka message processing failed due to a transient error that may succeed on retry. It bypasses Nest's normal Kafka error serialization flow and bubbles the error to KafkaJS's `eachMessage` callback, allowing the underlying consumer to handle retries or failure behavior.

Extends: `RpcException`

Methods

Method	Signature	Returns
`getError`	`getError()`	`string

Diagram

```
graph LR
  A[Kafka message received] --> B[NestJS message handler]
  B --> C{Transient failure?}
  C -- No --> D[Normal error processing / serialization]
  C -- Yes --> E[Throw KafkaRetriableException]
```

```
  E --> F[Bypass Nest Kafka error flow]
  F --> G[KafkaJS eachMessage callback]
  G --> H[KafkaJS retry / consumer error handling]
```

Usage

```
import { Controller } from '@nestjs/common';
import {
  Ctx,
  KafkaContext,
  MessagePattern,
} from '@nestjs/microservices';
import { KafkaRetriableException } from '@nestjs/microservices/exceptions/kafka-retriable-excep

@Controller()
export class OrdersConsumer {
  @MessagePattern('orders.created')
  async handleOrder(@Ctx() context: KafkaContext) {
    try {
      await this.processOrder(context.getMessage().value);
    } catch (error) {
      // Throw for temporary failures that should be handled by KafkaJS.
      throw new KafkaRetriableException({
        message: 'Order service is temporarily unavailable',
        cause: error,
      });
    }
  }

  private async processOrder(order: unknown) {
    // Process the incoming order.
  }
}
```

AI Coding Instructions

- Throw `KafkaRetriableException` only for transient failures, such as temporary network, database, or downstream-service errors.
- Do not use this exception for validation errors or permanently invalid messages; those should follow normal application error handling.
- The exception intentionally bypasses Nest's Kafka error serialization and reaches KafkaJS's `eachMessage` processing flow.
- Use `getError()` when integration code needs the original string or object error payload.

- Ensure KafkaJS consumer retry and offset-commit settings match the intended retry behavior before introducing this exception.