KafkaContext

August 18, 2026

KafkaContext

Kind: Class

Source: `packages/microservices/ctx-host/kafka.context.ts`

Part of: Microservices

`KafkaContext` provides access to Kafka-specific metadata and clients for a message handled by a NestJS microservice. It lets message handlers inspect the raw message, topic, and partition, while also exposing the consumer, producer, and heartbeat callback for advanced Kafka workflows.

Extends: `BaseRpcContext`

Methods

Method	Signature	Returns
`getMessage`	`getMessage()`	`void`
`getPartition`	`getPartition()`	`void`
`getTopic`	`getTopic()`	`void`
`getConsumer`	`getConsumer()`	`void`
`getHeartbeat`	`getHeartbeat()`	`void`
`getProducer`	`getProducer()`	`void`

Diagram

```
graph LR
  A[Kafka Consumer] --> B[Kafka Message]
  B --> C[KafkaContext]
  C --> D[getMessage()]
  C --> E[getTopic()]
  C --> F[getPartition()]
```

```
  C --> G[getConsumer()]
  C --> H[getHeartbeat()]
  C --> I[getProducer()]
  C --> J[NestJS Message Handler]
```

Usage

```ts
import { Controller } from '@nestjs/common';
import { Ctx, MessagePattern, Payload } from '@nestjs/microservices';
import { KafkaContext } from '@nestjs/microservices';

@Controller()
export class OrdersConsumer {
  @MessagePattern('orders.created')
  async handleOrderCreated(
    @Payload() order: { id: string; customerId: string },
    @Ctx() context: KafkaContext,
  ) {
    const message = context.getMessage();
    const topic = context.getTopic();
    const partition = context.getPartition();

    console.log(`Received order ${order.id} from ${topic}[${partition}]`);
    console.log('Kafka message key:', message.key?.toString());

    // Use this during long-running processing to keep the consumer session alive.
    await context.getHeartbeat();
  }
}
```

AI Coding Instructions

- Use `KafkaContext` as the `@Ctx()` argument in Kafka message handlers; avoid manually constructing it.
- Prefer `getTopic()`, `getPartition()`, and `getMessage()` when logging or diagnosing message processing behavior.
- Call `getHeartbeat()` during long-running handler work to prevent consumer-group rebalancing caused by missed heartbeats.
- Use `getConsumer()` and `getProducer()` only for Kafka-specific operations that cannot be handled through normal NestJS messaging patterns.
- Treat the raw Kafka message returned by `getMessage()` as transport metadata; decode keys, headers, and values defensively.