KafkaParser

August 18, 2026

KafkaParser

Kind: Class

Source: `packages/microservices/helpers/kafka-parser.ts`

Part of: Microservices

`KafkaParser` transforms raw Kafka message payloads into application-friendly objects. It preserves Kafka metadata and headers while decoding message values as JSON when possible, falling back to strings, buffers, or `null` when appropriate.

Methods

Method	Signature	Returns
`parse`	`parse(data: any)`	`T`
`decode`	`decode(value: Buffer)`	`object

Properties

Property	Type
`keepBinary`	`boolean`

Where it refuses work

- `KafkaParser` stops the work with an early return when `isNil(value)`.
- `KafkaParser` stops the work with an early return when `Buffer.isBuffer(value) && value.length > 0 && value.readUInt8(0) === 0`.

When something fails

- `KafkaParser` handles failure in 1 place: it discards it silently in all 1.

Diagram

```
graph LR
  A[Raw Kafka Message] --> B[KafkaParser.parse]
  B --> C[Extract metadata and headers]
  C --> D[KafkaParser.decode]
  D --> E{Valid JSON?}
  E -->|Yes| F[Parsed object]
  E -->|No| G[String or Buffer value]
  F --> H[Normalized message object]
  G --> H
```

Usage

```
import { KafkaParser } from '@nestjs/microservices';

const parser = new KafkaParser();

const rawMessage = {
  topic: 'orders',
  partition: 0,
  offset: '42',
  headers: {
    'correlation-id': Buffer.from('request-123'),
  },
  value: Buffer.from(
    JSON.stringify({
      orderId: 'order-456',
      status: 'created',
    }),
  ),
};

const message = parser.parse<{
  topic: string;
  partition: number;
  offset: string;
  headers: Record<string, Buffer>;
  value: {
    orderId: string;
    status: string;
  };
}>(rawMessage);

console.log(message.value.orderId); // "order-456"
console.log(message.value.status); // "created"
```

AI Coding Instructions

- Pass complete Kafka message objects to `parse()` so topic, partition, offset, and headers remain available after parsing.
- Expect `decode()` to return a parsed object for JSON payloads, a string for non-JSON text payloads, `null` for empty values, or a `Buffer` when applicable.
- Do not assume every Kafka message value is JSON; validate the parsed `value` shape before using object properties.
- Preserve Kafka headers and metadata when wrapping or extending parsed messages for downstream handlers.
- Use a custom value parser configuration when consumers require specialized serialization formats beyond JSON.

Relationships

- IMPORTS → `isNil`