Message

August 18, 2026

Message

Kind: Interface

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

Part of: Microservices

`Message` represents a Kafka record consumed or produced by the microservices transport layer. It stores the message payload, optional key, partition metadata, headers, and the Kafka-provided timestamp so downstream code can process records consistently.

Properties

Property	Type
`key`	`Buffer
`value`	`Buffer
`partition`	`number`
`headers`	`IHeaders`
`timestamp`	`string`

Diagram

```
graph LR
  Producer[Kafka Producer] --> Message[Message]
  Message --> Key[key: Buffer | string | null]
  Message --> Value[value: Buffer | string | null]
  Message --> Partition[partition: number]
  Message --> Headers[headers: IHeaders]
  Message --> Timestamp[timestamp: string]
  Message --> Consumer[Kafka Consumer]
```

Usage

```ts
import type { Message } from './kafka.interface';

function handleKafkaMessage(message: Message): void {
  const payload =
    message.value === null
      ? null
      : Buffer.isBuffer(message.value)
        ? message.value.toString('utf8')
        : message.value;

  console.log({
    key: message.key?.toString(),
    partition: message.partition,
    timestamp: new Date(message.timestamp),
    headers: message.headers,
    payload,
  });
}
```

AI Coding Instructions

- Treat `key` and `value` as nullable; check for `null` before decoding or parsing them.
- Support both `Buffer` and `string` payloads, using `Buffer.isBuffer()` before calling buffer-specific methods.
- Preserve `partition`, `headers`, and `timestamp` when forwarding or transforming a message.
- Parse `timestamp` only when needed; Kafka timestamps are represented as strings in this interface.
- Use `headers` for cross-cutting metadata such as tracing, correlation IDs, and content type.