EachBatchPayload

August 17, 2026

EachBatchPayload

Kind: Interface

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

Part of: Microservices

`EachBatchPayload` represents the payload provided to Kafka batch-processing handlers. It exposes the consumed `Batch`, allowing consumers to inspect batch metadata and process the messages it contains as a unit.

Properties

Property	Type
`batch`	`Batch`

Diagram

```
graph LR
  Consumer[Kafka Consumer Handler] --> Payload[EachBatchPayload]
  Payload --> Batch[Batch]
  Batch --> Messages[Kafka Messages]
```

Usage

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

async function handleBatch(payload: EachBatchPayload) {
  const { batch } = payload;

  for (const message of batch.messages) {
    const value = message.value?.toString();

    console.log(`Processing message from ${batch.topic}:`, value);
```

```
  }
}
```

AI Coding Instructions

- Use `payload.batch.messages` to iterate through all messages received in the Kafka batch.
- Treat `message.value` as nullable and check for `null` before converting or parsing it.
- Use batch metadata such as `batch.topic` and `batch.partition` when logging, routing, or handling failures.
- Keep batch handlers idempotent because Kafka may redeliver messages after consumer failures.

How it works

`EachBatchPayload` is an exported TypeScript interface that models the argument passed to an `EachBatchHandler`. This declaration file is intended to represent KafkaJS types only, without NestJS logic. packages/microservices/external/kafka.interface.ts:1-8 An `EachBatchHandler` accepts this payload and must return `Promise<void>`; such a handler can be assigned to the optional `eachBatch` setting of `ConsumerRunConfig`, which can be passed to `Consumer.run()`. packages/microservices/external/kafka.interface.ts:1025-1035 packages/microservices/external/kafka.interface.ts:1043-1049

Its members are:

- `batch: Batch` exposes the batch's `topic`, `partition`, `highWatermark`, and `KafkaMessage[]`, plus methods to test whether it is empty and read its first/last offsets and offset-lag values. packages/microservices/external/kafka.interface.ts:887-897
- `resolveOffset(offset: string): void` accepts one string offset and has no return value. packages/microservices/external/kafka.interface.ts:1002-1005
- `heartbeat(): Promise<void>` is asynchronous and resolves with no value. packages/microservices/external/kafka.interface.ts:1004-1006
- `pause(): () => void` returns a zero-argument function that returns no value. packages/microservices/external/kafka.interface.ts:1005-1007
- `commitOffsetsIfNecessary(offsets?: Offsets): Promise<void>` optionally accepts offsets and resolves with no value. `Offsets` contains `topics`, each with a topic name and partition/offset pairs. packages/microservices/external/kafka.interface.ts:1007-1008 packages/microservices/external/kafka.interface.ts:771-783

- `uncommittedOffsets(): OffsetsByTopicPartition` returns an object with `topics: TopicOffsets[]`. packages/microservices/external/kafka.interface.ts:990-992 packages/microservices/external/kafka.interface.ts:1007-1009
- `isRunning(): boolean` and `isStale(): boolean` return boolean status values. packages/microservices/external/kafka.interface.ts:1008-1011

`ConsumerEachBatchPayload` is an alias of `EachBatchPayload`, retained for compatibility with `@types/kafkajs`. packages/microservices/external/kafka.interface.ts:1019-1023

This is a type declaration: it contains no implementation, runtime validation, explicit thrown errors, or observable runtime side effects. packages/microservices/external/kafka.interface.ts:1-8 packages/microservices/external/kafka.interface.ts:1002-1011