TopicOffsets

August 18, 2026

TopicOffsets

Kind: Interface

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

Part of: Microservices

`TopicOffsets` represents Kafka offset information for a single topic across one or more partitions. It is typically used when committing offsets, seeking to a position, or reporting consumer progress within the microservices Kafka integration.

Properties

Property	Type
`topic`	`string`
`partitions`	`PartitionOffset[]`

Diagram

```
graph LR
  T[TopicOffsets] --> Topic[topic: string]
  T --> Partitions[partitions: PartitionOffset[]]
  Partitions --> P1[Partition offset]
  Partitions --> P2[Partition offset]
```

Usage

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

const offsets: TopicOffsets = {
  topic: 'orders.created',
  partitions: [
    { partition: 0, offset: '125' },
```

```
    { partition: 1, offset: '98' },
  ],
};

// Example: pass topic partition offsets to a Kafka consumer operation.
await consumer.commitOffsets(offsets.partitions);
```

AI Coding Instructions

- Provide a valid Kafka topic name in `topic` and include offsets only for partitions belonging to that topic.
- Use the `PartitionOffset` shape expected by the Kafka client; offsets are commonly represented as strings to preserve large integer values.
- Keep partition offsets grouped under one `TopicOffsets` object when handling topic-level consumer state.
- Validate partition numbers and offsets before committing or seeking, especially when offsets originate from external storage.
- When integrating with consumer APIs, check whether the API expects `PartitionOffset[]` directly or a topic-aware wrapper such as `TopicOffsets`.

How it works

`TopicOffsets` is an exported TypeScript interface that groups offset entries under one Kafka topic. It declares two required fields: `topic`, a string, and `partitions`, an array of `PartitionOffset` objects. [packages/microservices/external/kafka.interface.ts:776-779]

Each `PartitionOffset` requires a numeric `partition` and a string `offset`; therefore, every entry in `partitions` identifies one partition and its offset as a string. [packages/microservices/external/kafka.interface.ts:771-774]

The surrounding file declares types intended to represent the KafkaJS package rather than NestJS logic. [packages/microservices/external/kafka.interface.ts:1-8] `TopicOffsets` has no implementation, validation, error declaration, or direct side effect in this file. [packages/microservices/external/kafka.interface.ts:776-779]

It appears in several Kafka-facing type signatures:

- `Cluster.fetchTopicsOffset(...)` resolves to `Promise<TopicOffsets[]>`. [packages/microservices/external/kafka.interface.ts:221-230]

- `Broker.offsetCommit(...)` accepts `topics: TopicOffsets[]`. [packages/microservices/external/kafka.interface.ts:674-680]
- `Broker.offsetFetch(...)` accepts `topics: TopicOffsets[]` and returns `responses: TopicOffsets[]`. [packages/microservices/external/kafka.interface.ts:681-683]
- The consumer commit-offset instrumentation payload contains `topics: TopicOffsets[]`. [packages/microservices/external/kafka.interface.ts:928-938]
- `Offsets` and `OffsetsByTopicPartition` each wrap a `topics: TopicOffsets[]` array; `EachBatchPayload.commitOffsetsIfNecessary` accepts an optional `Offsets`, and `uncommittedOffsets` returns `OffsetsByTopicPartition`. [packages/microservices/external/kafka.interface.ts:781-783] [packages/microservices/external/kafka.interface.ts:990-1008]