BrokerMetadata

August 17, 2026

BrokerMetadata

Kind: Interface

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

Part of: Microservices

`BrokerMetadata` represents Kafka cluster metadata returned by the microservices Kafka transport layer. It combines the list of available brokers with per-topic and per-partition metadata, allowing clients to discover broker endpoints and topic partition status.

Properties

Property	Type
`brokers`	`Array<{ nodeId: number; host: string; port: number; rack?: string }>`
`topicMetadata`	`Array<{ topicErrorCode: number; topic: string; partitionMetadata: PartitionMetadata[]; }>`

Diagram

```
graph LR
  BM[BrokerMetadata]
  B[brokers]
  TM[topicMetadata]

  BM --> B
  BM --> TM

  B --> Broker["Broker endpoint<br/>nodeId, host, port, rack?"]
  TM --> Topic["Topic metadata<br/>topic, topicErrorCode"]
  Topic --> Partitions["PartitionMetadata[]"]
```

Usage

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

function logClusterMetadata(metadata: BrokerMetadata): void {
  for (const broker of metadata.brokers) {
    console.log(
      `Broker ${broker.nodeId} is available at ${broker.host}:${broker.port}`,
    );
  }

  for (const topicMetadata of metadata.topicMetadata) {
    if (topicMetadata.topicErrorCode !== 0) {
      console.warn(
        `Unable to read metadata for topic "${topicMetadata.topic}"`,
      );
      continue;
    }

    console.log(
      `Topic "${topicMetadata.topic}" has ` +
        `${topicMetadata.partitionMetadata.length} partition(s)`,
    );
  }
}
```

AI Coding Instructions

- Treat `topicErrorCode` as a Kafka protocol status code; verify it before relying on `partitionMetadata`.
- Use `nodeId` as the stable broker identifier, and use `host` plus `port` only for connection endpoints.
- Handle the optional `rack` field defensively because rack-aware metadata may not be configured.
- Keep broker and topic metadata aligned with Kafka protocol responses; do not assume every requested topic has valid partition metadata.
- Reuse `PartitionMetadata` when extending topic-level metadata rather than duplicating partition fields.

How it works

`BrokerMetadata` is an exported TypeScript interface in a file explicitly intended to represent KafkaJS package types rather than NestJS logic. packages/microservices/external/kafka.interface.ts:1-8

It describes metadata with two required arrays:

- `brokers`: broker records containing required numeric `nodeId`, string `host`, and numeric `port`; a broker record may also contain a string `rack`. packages/microservices/external/kafka.interface.ts:651-652
- `topicMetadata`: topic records containing a numeric `topicErrorCode`, string `topic`, and `partitionMetadata` array. packages/microservices/external/kafka.interface.ts:653-657 Each partition record has error code, partition ID, leader ID, replica IDs, and in-sync replica IDs; `offlineReplicas` is optional. packages/microservices/external/kafka.interface.ts:151-158

It is the resolved type returned by `Cluster.metadata()` and by `Broker.metadata(topics)`. The cluster method takes no arguments, while the broker method requires a string array of topic names. packages/microservices/external/kafka.interface.ts:199-201 packages/microservices/external/kafka.interface.ts:667-672

The interface declaration contains no implementation, runtime validation, thrown errors, or side effects. packages/microservices/external/kafka.interface.ts:651-658