MatchingAcl

August 18, 2026

MatchingAcl

Kind: Interface

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

Part of: Microservices

`MatchingAcl` represents a Kafka ACL entry returned when querying or filtering access-control rules. It combines the resource being protected, the principal and host being evaluated, the allowed or denied operation, and any broker-reported error details.

Properties

Property	Type
`errorCode`	`number`
`errorMessage`	`string`
`resourceType`	`AclResourceTypes`
`resourceName`	`string`
`resourcePatternType`	`ResourcePatternTypes`
`principal`	`string`
`host`	`string`
`operation`	`AclOperationTypes`
`permissionType`	`AclPermissionTypes`

Diagram

```
graph LR
  MatchingAcl["MatchingAcl"]
  MatchingAcl --> Error["errorCode<br/>errorMessage"]
  MatchingAcl --> Resource["Kafka Resource"]
```

```
  MatchingAcl --> Subject["Principal and Host"]
  MatchingAcl --> Permission["Operation and Permission"]

  Resource --> ResourceType["resourceType: AclResourceTypes"]
  Resource --> ResourceName["resourceName: string"]
  Resource --> PatternType["resourcePatternType: ResourcePatternTypes"]

  Subject --> Principal["principal: string"]
  Subject --> Host["host: string"]

  Permission --> Operation["operation: AclOperationTypes"]
  Permission --> PermissionType["permissionType: AclPermissionTypes"]
```

Usage

```
import {
  AclOperationTypes,
  AclPermissionTypes,
  AclResourceTypes,
  ResourcePatternTypes,
} from '@nestjs/microservices';
import type { MatchingAcl } from '@nestjs/microservices';

const matchingAcl: MatchingAcl = {
  errorCode: 0,
  errorMessage: '',
  resourceType: AclResourceTypes.TOPIC,
  resourceName: 'orders',
  resourcePatternType: ResourcePatternTypes.LITERAL,
  principal: 'User:order-service',
  host: '*',
  operation: AclOperationTypes.READ,
  permissionType: AclPermissionTypes.ALLOW,
};

if (
  matchingAcl.errorCode === 0 &&
  matchingAcl.permissionType === AclPermissionTypes.ALLOW
) {
  console.log(
    `${matchingAcl.principal} can ${matchingAcl.operation} from ${matchingAcl.resourceName}`,
  );
}
```

AI Coding Instructions

- Treat `errorCode` and `errorMessage` as broker response metadata; check for non-zero error codes before relying on an ACL result.

- Use the Kafka ACL enum values for `resourceType`, `resourcePatternType`, `operation`, and `permissionType`; avoid raw numeric values.
- Preserve Kafka principal formatting, such as `User:service-name`, when comparing or constructing ACL-related data.
- Account for wildcard hosts (`*`) and resource pattern types when evaluating whether an ACL applies to a request.