IORedisOptions

August 18, 2026

IORedisOptions

Kind: Interface

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

Part of: Microservices

`IORedisOptions` defines the Redis client connection and behavior options used by the microservices Redis integration. It configures authentication, database selection, TCP settings, command timeouts, client identification, and reconnect behavior for an ioredis-compatible connector.

Properties

Property	Type
`Connector`	`any`
`retryStrategy`	`(times: number) => number
`commandTimeout`	`number`
`keepAlive`	`number`
`noDelay`	`boolean`
`connectionName`	`string`
`clientInfoTag`	`string`
`username`	`string`
`password`	`string`
`db`	`number`
`autoResubscribe`	`boolean`
`autoResendUnfulfilledCommands`	`boolean`
`reconnectOnError`	`((err: Error) => boolean

Property	Type
`readOnly`	`boolean`
`stringNumbers`	`boolean`
`connectTimeout`	`number`
`monitor`	`boolean`
`maxRetriesPerRequest`	`number
`maxLoadingRetryTime`	`number`
`enableAutoPipelining`	`boolean`
`autoPipeliningIgnoredCommands`	`string[]`
`offlineQueue`	`boolean`
`commandQueue`	`boolean`
`enableOfflineQueue`	`boolean`
`enableReadyCheck`	`boolean`
`lazyConnect`	`boolean`
`scripts`	`Record< string, { lua: string; numberOfKeys?: number; readOnly?: boolean } >`
`keyPrefix`	`string`
`showFriendlyErrorStack`	`boolean`
`disconnectTimeout`	`number`
`tls`	`ConnectionOptions`
`name`	`string`
`role`	`'master'
`sentinelUsername`	`string`
`sentinelPassword`	`string`
`sentinels`	`Array<Partial<any>>`
`sentinelRetryStrategy`	`(retryAttempts: number) => number
`sentinelReconnectStrategy`	`(retryAttempts: number) => number
`preferredSlaves`	`any`
`sentinelCommandTimeout`	`number`

Property	Type
`enableTLSForSentinelMode`	`boolean`
`sentinelTLS`	`ConnectionOptions`
`natMap`	`any`
`updateSentinels`	`boolean`
`sentinelMaxConnections`	`number`
`failoverDetector`	`boolean`

Diagram

```
graph LR
  App[Microservice] --> Options[IORedisOptions]
  Options --> Connector[Connector]
  Options --> Auth[username / password]
  Options --> Database[db]
  Options --> Network[keepAlive / noDelay]
  Options --> Retry[retryStrategy]
  Options --> Timeout[commandTimeout]
  Options --> Identity[connectionName / clientInfoTag]
  Connector --> Redis[(Redis Server)]
```

Usage

```
import type { IORedisOptions } from './redis.interface';
import Redis from 'ioredis';

const redisOptions: IORedisOptions = {
  Connector: Redis,
  host: 'localhost',
  port: 6379,
  username: 'default',
  password: process.env.REDIS_PASSWORD,
  db: 0,

  connectionName: 'orders-service',
  clientInfoTag: 'orders-worker',

  commandTimeout: 5_000,
  keepAlive: 10_000,
  noDelay: true,

  retryStrategy: (times) => {
    if (times > 10) {
```

```
      return null; // Stop retrying after 10 attempts.
    }

    return Math.min(times * 200, 2_000);
  },
};
```

AI Coding Instructions

- Provide an ioredis-compatible constructor through `Connector`; ensure it supports the configured connection options.
- Keep credentials in environment variables or a secrets manager rather than hardcoding `username` or `password`.
- Return a delay in milliseconds from `retryStrategy`; return `null` or `void` when reconnect attempts should stop.
- Use distinct `connectionName` and `clientInfoTag` values per service to simplify Redis monitoring and debugging.
- Configure `commandTimeout`, `keepAlive`, and `noDelay` according to the deployment network and service latency requirements.