RedisContext

August 18, 2026

RedisContext

Kind: Class

Source: `packages/microservices/ctx-host/redis.context.ts`

Part of: Microservices

`RedisContext` provides Redis-specific metadata to microservice message handlers. It wraps the incoming Redis message context and exposes the channel name through `getChannel()`, allowing handlers to identify the source channel.

Extends: `BaseRpcContext`

Methods

Method	Signature	Returns
`getChannel`	`getChannel()`	`void`

Diagram

```
graph LR
  Redis[Redis Pub/Sub Channel] --> Server[Redis Transport Server]
  Server --> Context[RedisContext]
  Context --> Handler[Message Handler]
  Handler --> Channel[getChannel()]
```

Usage

```
import { Controller } from '@nestjs/common';
import { Ctx, MessagePattern, RedisContext } from '@nestjs/microservices';

@Controller()
export class NotificationsController {
  @MessagePattern('notifications.*')
```

```
  handleNotification(
    payload: { message: string },
    @Ctx() context: RedisContext,
  ) {
    const channel = context.getChannel();

    console.log(`Received "${payload.message}" from ${channel}`);

    return { received: true, channel };
  }
}
```

AI Coding Instructions

- Use `RedisContext` only in handlers executed through the Redis microservice transport.
- Retrieve the originating Redis channel with `context.getChannel()` rather than relying on a hard-coded channel name.
- Inject the context with `@Ctx()` alongside payload parameters in `@MessagePattern()` handlers.
- Treat the channel value as transport metadata; keep business logic independent of Redis-specific context where possible.

How it works

`RedisContext` is an exported RPC context class for Redis message handling. It extends `BaseRpcContext` with a one-element argument tuple whose item is a `string`. redis.context.ts:3-10

- **Construction:** `new RedisContext(args)` accepts the `[string]` tuple and passes that same tuple to the base-class constructor, which stores it as protected, read-only `args`. redis.context.ts:3-10 base-rpc.context.ts:4-5
- `getChannel()` returns the tuple's first item (`args[0]`); the class documentation identifies this value as the channel name. redis.context.ts:13-18
- **Inherited accessors:** `getArgs()` returns the stored tuple, and `getArgByIndex(index)` returns the item at the requested index. base-rpc.context.ts:7-20
- **Server construction:** `ServerRedis` creates this context as `new RedisContext([pattern])` while handling an incoming message. Without wildcard mode, it passes the received `channel` as `pattern`; with wildcard mode, it passes the callback's `pattern` argument. server-redis.ts:126-142 The resulting context is

passed to event handling, request handlers, and processing hooks. server-redis.ts:144-172

- **Validation, errors, and side effects:** Neither the constructor nor `getChannel()` performs runtime validation, throws an error, or mutates external state in the visible code; they store and read the argument tuple. redis.context.ts:8-18 base-rpc.context.ts:4-5