RpcArgumentsHost

August 18, 2026

RpcArgumentsHost

Kind: Interface

Source: `packages/common/interfaces/features/arguments-host.interface.ts`

Part of: Common

Methods to obtain RPC data object.

`RpcArgumentsHost` provides access to the data payload and transport-specific context for an RPC request. It is typically obtained from an `ArgumentsHost` through `switchToRpc()` inside guards, interceptors, filters, or pipes that support NestJS microservice transports.

Diagram

```
graph LR
  A[ArgumentsHost] -->|switchToRpc()| B[RpcArgumentsHost]
  B -->|getData()| C[RPC Data Payload]
  B -->|getContext()| D[Transport Context]
  C --> E[Guard / Interceptor / Filter]
  D --> E
```

Usage

```
import { CanActivate, ExecutionContext, Injectable } from '@nestjs/common';

@Injectable()
export class RpcAuthGuard implements CanActivate {
  canActivate(context: ExecutionContext): boolean {
    const rpc = context.switchToRpc();

    const data = rpc.getData<{ token?: string; action: string }>();
    const rpcContext = rpc.getContext<{ clientId?: string }>();

    if (!data.token) {
```

```
      return false;
    }

    console.log(`Authorizing ${data.action} for ${rpcContext.clientId}`);
    return true;
  }
}
```

AI Coding Instructions

- Obtain this interface through `context.switchToRpc()` when handling microservice or RPC requests.
- Use generic type parameters with `getData<T>()` and `getContext<T>()` to avoid untyped payload access.
- Treat `getData()` as the message payload and `getContext()` as transport-specific metadata, such as client or broker context.
- Do not use HTTP-specific methods such as `switchToHttp()` when the handler is invoked through an RPC transport.
- Keep transport-specific context handling isolated so shared guards and interceptors remain portable across RPC transports.

Used by

1 reference from 1 file. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (1)

- `ExecutionContextHost` — `packages/core/helpers/execution-context-host.ts`:10