# RpcParamsFactory

August 18, 2026

## RpcParamsFactory

**Kind:** Class

**Source:** `packages/microservices/factories/rpc-params-factory.ts`

**Part of:** Microservices

`RpcParamsFactory` resolves RPC handler parameter values from the arguments supplied by a microservice transport. Its `exchangeKeyForValue()` method maps an RPC parameter type—such as payload or context—to the appropriate value, including selecting a named property from the payload.

### Methods

| Method | Signature | Returns |
| --- | --- | --- |
| `exchangeKeyForValue` | `exchangeKeyForValue(type: number, data: string | undefined, args: unknown[])` |

### Where it refuses work

- `RpcParamsFactory` stops the work with an early return when `!args`.

### Diagram

```
graph LR
  A[Transport invokes RPC handler] --> B[Handler arguments array]
  B --> C[RpcParamsFactory.exchangeKeyForValue]
  C --> D{RPC parameter type}
  D -->|PAYLOAD| E[Full payload or payload property]
  D -->|CONTEXT| F[Transport context]
  D -->|Unknown type| G[null]
```

### Usage

```
import { RpcParamsFactory } from '@nestjs/microservices/factories/rpc-params-factory';
import { RpcParamtype } from '@nestjs/microservices/enums/rpc-paramtype.enum';

const paramsFactory = new RpcParamsFactory();

const payload = { id: 'user-123', action: 'created' };
const context = { pattern: 'users.create' };
const args = [payload, context];

// Resolve the complete message payload.
const message = paramsFactory.exchangeKeyForValue(
  RpcParamtype.PAYLOAD,
  undefined,
  args,
);

// Resolve a property from the message payload.
const userId = paramsFactory.exchangeKeyForValue(
  RpcParamtype.PAYLOAD,
  'id',
  args,
);

// Resolve the RPC transport context.
const rpcContext = paramsFactory.exchangeKeyForValue(
  RpcParamtype.CONTEXT,
  undefined,
  args,
);
```

## AI Coding Instructions

- Pass handler arguments in their expected order: payload at index `0` and RPC context at index `1`.
- Use `RpcParamtype.PAYLOAD` with a property key to extract a specific payload field; omit the key to return the full payload.
- Use `RpcParamtype.CONTEXT` only when the transport provides context as the second handler argument.
- Preserve the `null` fallback for unsupported parameter types or missing argument arrays so parameter resolution fails safely.

## How it works

`RpcParamsFactory` is a class with one method, `exchangeKeyForValue(type, data, args)`, that selects an RPC handler parameter value from an invocation-argument array. Its

parameters are a numeric type, optional string data, and `unknown[]` arguments. [packages/microservices/factories/rpc-params-factory.ts:3-8]

- For `RpcParamtype.PAYLOAD`, it returns `args[0]` when `data` is falsy; when `data` is truthy, it returns the property named by `data` from `args[0]`. Optional chaining means a missing or nullish first argument yields `undefined` for property extraction. [packages/microservices/factories/rpc-params-factory.ts:12-15]
- For `RpcParamtype.CONTEXT`, it returns `args[1]`; `data` is ignored. [packages/microservices/factories/rpc-params-factory.ts:12-17]
- For `RpcParamtype.GRPC_CALL`, it returns `args[2]`; `data` is ignored. [packages/microservices/factories/rpc-params-factory.ts:12-19]
- The enum defines these three kinds as `PAYLOAD`, `CONTEXT`, and `GRPC_CALL`. [packages/microservices/enums/rpc-paramtype.enum.ts:3-7]
- If `args` is falsy, or if `type` does not match one of those enum values, the method returns `null`. [packages/microservices/factories/rpc-params-factory.ts:9-11] [packages/microservices/factories/rpc-params-factory.ts:19-20]
- It does not check whether the selected array slot or requested payload property exists. Thus, with a truthy `args` array, missing slots or properties can result in `undefined`. [packages/microservices/factories/rpc-params-factory.ts:14-18]
- The method only reads from `args` and does not write to it or other state. [packages/microservices/factories/rpc-params-factory.ts:9-21]

`RpcContextCreator` owns an instance of this class and creates extraction callbacks that pass the callback's received arguments into `exchangeKeyForValue`. [packages/microservices/context/rpc-context-creator.ts:41-45] [packages/microservices/context/rpc-context-creator.ts:237-241] The default non-gRPC metadata maps the payload parameter to argument slot `0`; the gRPC defaults additionally map context and gRPC call parameters to slots `1` and `2`. [packages/microservices/context/rpc-metadata-constants.ts:3-10]