RequestContext

August 17, 2026

RequestContext

Kind: Interface

Source: `packages/microservices/interfaces/request-context.interface.ts`

Part of: Microservices

`RequestContext` describes the normalized payload passed through a microservice request handler. It groups the message `pattern`, request `data`, and transport-specific `context` so handlers can process incoming requests consistently across transports.

Properties

Property	Type
`pattern`	`string
`data`	`TData`
`context`	`TContext`

Diagram

```
graph LR
  Client[Client Request] --> Pattern[pattern<br/>Routing pattern]
  Client --> Data[data<br/>Request payload]
  Transport[Microservice Transport] --> Context[context<br/>Transport metadata]

  Pattern --> RequestContext[RequestContext]
  Data --> RequestContext
  Context --> RequestContext

  RequestContext --> Handler[Message Handler]
```

Usage

```ts
import type { RequestContext } from './interfaces/request-context.interface';

interface CreateUserData {
  email: string;
  name: string;
}

interface TransportContext {
  correlationId: string;
  headers?: Record<string, string>;
}

function handleCreateUser(
  request: RequestContext<CreateUserData, TransportContext>,
) {
  const { pattern, data, context } = request;

  console.log(`Handling pattern: ${String(pattern)}`);
  console.log(`Correlation ID: ${context.correlationId}`);

  return {
    email: data.email,
    name: data.name,
  };
}

const request: RequestContext<CreateUserData, TransportContext> = {
  pattern: 'users.create',
  data: {
    email: 'ada@example.com',
    name: 'Ada Lovelace',
  },
  context: {
    correlationId: 'req-123',
  },
};

handleCreateUser(request);
```

AI Coding Instructions

- Use generic type parameters for `data` and `context` to preserve payload and transport metadata types.
- Treat `pattern` as either a string route or a structured object when supporting complex message routing.
- Keep business-specific payload validation in handlers or validation layers rather than expanding this shared interface.

- Pass transport-specific metadata through `context`, such as headers, correlation IDs, acknowledgements, or connection details.
- Avoid assuming a specific transport context shape unless the handler constrains `TContext` explicitly.

How it works

`RequestContext<TData, TContext>` is an exported generic interface for an RPC request-shaped object. `TData` defaults to `any`; `TContext` defaults to `any` and is constrained to extend `BaseRpcContext`. request-context.interface.ts:3-6

An implementation must expose:

- `pattern`, typed as either a `string` or `Record<string, any>`. request-context.interface.ts:7
- `data`, typed as `TData`. request-context.interface.ts:8
- An optional `context` property typed as `TContext`. request-context.interface.ts:9
- `getData()`, `getPattern()`, and `getContext()` methods that return the corresponding declared types; notably, `getContext()` has return type `TContext` even though the `context` property is optional. request-context.interface.ts:11-13

The `TContext` constraint means a typed context can inherit `BaseRpcContext`'s argument accessors: `getArgs()` returns its stored arguments, and `getArgByIndex(index)` returns the argument at that index. base-rpc.context.ts:4-5 base-rpc.context.ts:10-12 base-rpc.context.ts:18-20

`RequestContextHost` is the concrete class in this package that implements this contract. Its constructor stores readonly `pattern`, `data`, and `context` values, and each getter returns the stored value without transformation. request-context-host.ts:7-15 request-context-host.ts:26-36 Its static `create()` accepts a pattern, data, and `BaseRpcContext` subtype, constructs a host, and returns it typed as `RequestContext<TData, TContext>`. request-context-host.ts:17-24

For request-scoped microservice handlers, `ListenersController` creates a `RequestContextHost` from the handler's pattern, first argument (`data`), and second argument cast as `BaseRpcContext` when the first argument is not already a host. listeners-controller.ts:238-255 It then obtains a context ID from that request object. listeners-controller.ts:302-306 If the request has no existing request-context ID property, this path defines a non-enumerable, non-writable, non-configurable ID property on it, registers either the ID payload or the request object merged with that

payload as the request-provider value, and returns the ID. listeners-controller.ts:307-320

The interface itself declares no validation, error handling, construction logic, or side effects. request-context.interface.ts:3-14