ReadPacket

August 18, 2026

ReadPacket

Kind: Interface

Source: `packages/microservices/interfaces/packet.interface.ts`

Part of: Microservices

`ReadPacket<T>` represents a normalized incoming microservice message. It carries the routing `pattern` used to identify the handler and the typed `data` payload passed to that handler.

Properties

Property	Type
`pattern`	`any`
`data`	`T`

Diagram

```
graph LR
  Client[Microservice Client] --> Packet[ReadPacket<T>]
  Packet --> Pattern[pattern: message route]
  Packet --> Data[data: T payload]
  Pattern --> Handler[Matching Message Handler]
  Data --> Handler
```

Usage

```
import type { ReadPacket } from './interfaces/packet.interface';

interface CreateUserPayload {
  email: string;
  name: string;
```

```ts
}

const packet: ReadPacket<CreateUserPayload> = {
  pattern: 'users.create',
  data: {
    email: 'ada@example.com',
    name: 'Ada Lovelace',
  },
};

function handleCreateUser(message: ReadPacket<CreateUserPayload>) {
  if (message.pattern !== 'users.create') {
    return;
  }

  return createUser(message.data);
}
```

AI Coding Instructions

- Use a generic type parameter for `data` so message payloads remain strongly typed.
- Treat `pattern` as the message-routing identifier; ensure it matches the corresponding microservice handler pattern.
- Validate or transform `data` at the application boundary before relying on its fields.
- Avoid assuming a specific `pattern` type, because it is intentionally declared as `any`.

How it works

`ReadPacket<T = any>` is a generic TypeScript interface representing the input portion of a microservice packet: a required `pattern` field typed as `any` and a required `data` field typed as `T`. If no type argument is supplied, `data` is `any`. It declares no methods, validation, error handling, or runtime side effects itself. [packet.interface.ts:5-8]

It is the base type for both request and event packet aliases. `OutgoingRequest` and `IncomingRequest` add a string `id` through intersection with `PacketId`; `OutgoingEvent` and `IncomingEvent` are `ReadPacket` directly. [packet.interface.ts:1-3] [packet.interface.ts:17-20]

`ClientProxy.send()` and `emit()` construct packets as `{ pattern, data }` only after rejecting `null` or `undefined` values for either argument with `InvalidMessageException`.

[client-proxy.ts:86-101] [client-proxy.ts:111-120] For request flows, `assignPacketId()` mutates the passed packet with a generated `id` and returns it as `ReadPacket & PacketId`. [client-proxy.ts:160-163]

The interface is exported from the microservices interfaces barrel. [interfaces/index.ts:1-12]