

Response Shape

Response Shape

Anonymous · 10/10/2026

Response Shape

A successful, JSON-wanting request (Accept: application/json or */*) gets wrapped in a small envelope — for an operation result, meta.timestamp/meta.data_type come straight from `OperationResultInterface::getMetadata()->getTimestamp()/::getDataType()` (see [Backbone Dispatcher](#)), not recomputed from the already-serialized value:

```
{
  "meta": { "timestamp": 1755500000.123456, "data_type": "integer" },
  "data": 12
}
```

Two results bypass the envelope entirely: a PSR-7 `ResponseInterface` returned by the underlying dispatch (passed through unchanged), and the OpenAPI document itself (detected by its own `openapi` key). If the request doesn't ask for JSON, the raw result is returned unwrapped for the framework layer to render however it sees fit.

Failure: a Real Status Code, Never a Silent 200

`SafeDispatcherInterface` never throws, so `AbstractController` is also the one place that decides what a failed operation actually looks like to the client — a real PSR-7 response (built via an injected `Psr\Http\Message\ResponseFactoryInterface`, so this package stays implementation-agnostic about which PSR-17 factory produces it) carrying:

- The resolved HTTP status, from `HttpStatusResolver` (below) — never a default 200 masking a real failure.
- The exact same `ProblemDetailInterface::toArray()` body [Backbone Dispatcher](#) and [Backbone Console](#) already produce — the same failure looks the same across every transport.

```
$problem = $result->getProblem();
$status = $this->httpStatusResolver->resolve($problem->getThrowable()->getClass());

$response = $this->responseFactory
    ->createResponse($status)
    ->withHeader('Content-Type', 'application/json')
;
```

```
$response->getBody()->write((string) json_encode($problem->toArray()));
```

HttpStatusResolver: Mapping Exceptions to Status Codes

Shared by two consumers: `AbstractController` above (a real failure's actual status) and `Documenter` below (what to document for each `#[Operation(results: ...)]` scenario) — the same resolution, kept in one place instead of two copies that could drift apart. Only `derafu/backbone-dispatcher`'s own 7 generic exceptions are mapped (the same 7 [Backbone Console](#)'s `DefaultExitCodeResolver` maps, mirroring that same split) — anything else, including a business-specific exception, falls back to 500:

Status	Exception
200	('success')
403	<code>OperationNotAllowedException</code>
422	<code>MissingParameterException</code> , <code>InvalidParameterTypeException</code> , <code>NoDeserializerFoundException</code> , <code>ClassNotFoundException</code> , <code>FromArrayMethodNotFoundException</code>
500	anything else, including <code>OperationNotFoundException</code> and any business-specific exception

There is no built-in way to extend this mapping with a project's own business exceptions yet — unlike `DefaultExitCodeResolver`, `HttpStatusResolver` is not currently designed to be subclassed for that. A project needing its own status codes today would wrap or replace it in its own DI wiring.

Last updated on 10/10/2026