

Routing and Invocation

Routing and Invocation

Anonymous · 10/10/2026

Routing and Invocation

Routing Convention

`Router::parse()` resolves an incoming request into up to four path segments:

```
/[api/] :package/:component/:worker/:operation
```

The `/api` prefix is optional and stripped if present — `/api/billing/document` and `/billing/document` resolve identically. A path with more than 4 segments throws `InvalidRouteException`. Trailing segments are simply omitted rather than required, which is what drives the dispatch behavior below.

Two special one-segment paths are intercepted before being treated as a package name:

- Empty path or `index` → the root HATEOAS listing.
- `openapi-docs.json` → the generated OpenAPI document.

```
public function dispatch(ServerRequestInterface $request): mixed
{
    $route = $this->router->parse($request);

    return match (true) {
        $route->getId() === 'index' || $route->getId() === null =>
            $this->handleRoot(),
        $route->getId() === 'openapi-docs.json' => $this->documenter->document(),
        $route->getComponent() === null => $this->handlePackage($route->getPackage()),
        $route->getWorker() === null => $this->handleComponent($route->getPackage(),
            $route->getComponent()),
        $route->getOperation() === null => $this->handleWorker(...),
        default => $this->handleOperation($request, ...),
    };
}
```

So GET `/api/billing` lists `billing`'s components, GET `/api/billing/document` lists that component's workers, and only a full 4-segment path actually invokes an operation.

Invoking an Operation

Every operation is invoked with the parameters read from the JSON request body's `parameters` key — nothing else (no query string, no path params beyond routing):

```
POST /api/billing/invoice/builder/build
Content-Type: application/json
Accept: application/json

{ "parameters": { "number": "F-001", "amount": 15000 } }
```

```
$requestContent = json_decode($request->getBody()->getContents(), true);
$params = $requestContent['parameters'] ?? [];

$operationRequest = new OperationRequest($package, $component, $worker, $operation,
$params);

return $this->dispatcher->dispatch($operationRequest); // SafeDispatcherInterface –
never throws.
```

`handleOperation()` returns the `OperationResultInterface` itself, unwrapped — `AbstractController` (below) is the single place that turns it into the actual response, since it is also the place that already builds every other response shape this package produces.

Last updated on 10/10/2026