Architecture

Architecture

Anonymous · 09/11/2026

Architecture

`derafu/escpos` wraps mike42/escpos-php instead of implementing the ESC/POS protocol itself — but that library is treated as a replaceable implementation detail, not part of this package's contract. Nothing in the public API accepts or returns a `Mike42\Escpos\*` type or constant; see Vocabulary and Printing Images for the two places that leak would normally happen (command values, and image types) and how they are closed.

Why this matters

If a consumer's code imports `Mike42\Escpos\*` directly to call this package, upgrading `mike42/escpos-php` to a new major version can break that consumer's code too — even though they never touched the underlying library's version themselves. Isolating it behind this package's own API means a breaking change in the underlying library only needs to be absorbed in one place: here.

The seam: `PrinterInterface`

`Derafu\Escpos\Contract\PrinterInterface` defines every operation `EscposPrinter` needs from a backend: `text()`, `feed()`, `selectPrintMode()`, `setJustification()`, `barcode()`, `pdf417()`, `image()`, `cut()`, `pulse()`, `getOutput()`, `close()`.

`Derafu\Escpos\Adapter\Mike42Printer` is the only class in this package allowed to `use Mike42\Escpos\*`, and it is the default implementation of `PrinterInterface`. `EscposPrinter` is a thin facade: it owns option resolution, the `sprintf()`/line-break convenience of `print()`/`println()`, special character handling, and the `end()` workflow (finish, optionally cut and pulse) — and delegates everything else to whichever `PrinterInterface` it was given.

```
use Derafu\Escpos\Contract\PrinterInterface;
use Derafu\Escpos\EscposPrinter;

// Normally you never need this — omitting the second argument builds the
// default Mike42Printer from your options array.
$printer = new EscposPrinter($options, $customPrinterImplementation);
```

That second constructor argument is the extension point: a different backend (or a test double) can

be swapped in without changing `EscposPrinter`'s public API, or any code that already depends on `PrinterInterface` instead of the concrete class.

Dependency Injection

Import the package configuration to register the reusable rendering and delivery helpers, together with the translation resource provider:

```
imports:
    - { resource: '../vendor/derafu/escpos/resources/config/escpos-services.yaml' }
```

The configuration registers these shared services under their concrete class IDs:

- `Derafu\Escpos\Renderer\Pdf417Renderer`
- `Derafu\Escpos\Transport\NetworkTransport`

Both helpers have no state between operations. Inject them into the consuming service rather than constructing them at each call. The renderer returns a `RasterImage`; the transport receives the finished bytes, host, port and timeout when `send()` is called. Importing the YAML does not open a network connection or send a print job.

`EscposPrinter` and its backend are deliberately absent from the service configuration: each print job owns its options, buffer and printer state. Construct a fresh `EscposPrinter($options)` per job and use the shared helpers where needed:

```
// $pdf417Renderer and $networkTransport are injected shared helpers.
$printer = new EscposPrinter($options);
$printer->println('Receipt');
$printer->image($pdf417Renderer->render($barcodeContent));
$bytes = $printer->end();
$networkTransport->send($bytes, $host, $port);
```

Sharing the helpers does not share receipt buffers or printer configuration. Mike42 remains encapsulated by `Adapter\Mike42Printer`.

Exceptions

Every exception this package throws implements `Derafu\Escpos\Exception\EscposExceptionInterface`, so you can catch anything it raises without needing to know the specific class:

Exception	Thrown when
`ConnectorNotSupportedException`	The configured connector type has no known implementation (only `memory` exists today).
`ProfileNotFoundException`	The configured capability profile does not exist.
`UnbufferedConnectorException`	`dump()`/`end()` is called on a connector that does not buffer data in memory (`memory` always does; this exists so the failure mode is a clear exception if that ever changes, not a corrupted read).
`InvalidImageDataException`	`RasterImage::fromPngData()` is given data that cannot be decoded as an image.
`NetworkDeliveryException`	`NetworkTransport::send()` cannot connect, or cannot deliver all bytes.

None of these extend a `Mike42\Escpos\*` exception, for the same reason described above.

Last updated on 09/11/2026