Translations

Translations

Anonymous · 09/09/2026

Translations

`derafu/data-processor`'s exceptions can be translated. This page explains what's translatable, what ships out of the box, and how to activate it in your application.

What's Translatable

Four exception classes implement `Derafu\Translation\Contract\TranslatableInterface` (via `TranslatableExceptionTrait`, from `derafu/translation`):

- `Derafu\DataProcessor\Exception\ValidationException`
- `Derafu\DataProcessor\Exception\SanitizationException`
- `Derafu\DataProcessor\Exception\TransformationException`
- `Derafu\DataProcessor\Exception\CastingException`

`Derafu\DataProcessor\Exception\RuleNotFoundException` is **not** translatable on purpose — it signals a misconfigured rule name (a programming/configuration error), not something an end user should ever see.

All four use the `errors` domain (`TranslatableExceptionTrait`'s default). Every message thrown by a built-in rule is written in ICU MessageFormat syntax and uses the literal English text as its translation id — for example, `'Value must be greater than {value}.'` — following the same convention documented in derafu/translation's Exceptions guide.

Without any translator configured, `getMessage()` already returns the fully formatted English text — this library works exactly as before, whether or not you set up translation at all:

```
use Derafu\DataProcessor\Exception\ValidationException;

$e = new ValidationException(['Array must not contain more than {max} items.', 'max'
=> 5]);

echo $e->getMessage();
// "Array must not contain more than 5 items."
```

What Ships

`derafu/data-processor` includes:

- `resources/translations/errors+intl-icu.es.php` — a Spanish translation for every distinct message used by the four exception classes above (the `+intl-icu` domain suffix is required for `derafu/translation`'s ICU placeholders like `{max}` to be substituted — see ICU Formatting).
- `Derafu\DataProcessor\Translation\DataProcessorTranslationResourceProvider` — a `TranslationResourceProviderInterface` implementation pointing at that directory, ready to hand to a `TranslationResourceRegistrar` or a dependency-injection container.

Neither of these does anything by itself — `derafu/data-processor` is a library, it doesn't build or own a `Translator`. Activating translation is entirely up to the application that uses it.

Activating It (Plain PHP)

```
use Derafu\DataProcessor\Exception\ValidationException;
use Derafu\DataProcessor\Translation\DataProcessorTranslationResourceProvider;
use Derafu\Translation\TranslatorFactory;

$translator = TranslatorFactory::create(
    defaultLocale: 'es',
    fallbackLocales: ['es', 'en'],
    resourceProviders: [new DataProcessorTranslationResourceProvider()],
);

$e = new ValidationException(['Array must not contain more than {max} items.', 'max'
=> 5]);

echo $e->trans($translator, 'es');
// "El arreglo no debe contener más de 5 elementos."
```

If you'd rather point at the directory directly (for example, to combine it with your own translation files) without going through the provider class, use `TranslationResourceRegistrar` instead:

```
use Derafu\Translation\TranslationResourceRegistrar;

$registrar = new TranslationResourceRegistrar($translator);
$registrar->registerDirectory(__DIR__ . '/vendor/derafu/data-
processor/resources/translations');
$registrar->registerDirectory(__DIR__ . '/translations'); // Your own, registered
last: it can override.
```

Activating It (Dependency Injection)

If your application uses `symfony/dependency-injection`, import both recipe files and feed the tagged providers into your own `Translator` registration:

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

services:
    Symfony\Component\Translation\Translator:
        factory: ['Derafu\Translation\TranslatorFactory', 'create']
        arguments:
            $defaultLocale: 'es'
            $fallbackLocales: ['es', 'en']
            $resourceProviders: !tagged_iterator derafu_translation.resource_provider
```

`data-processor-services.yaml` already tags `DataProcessorTranslationResourceProvider` with `derafu_translation.resource_provider`, so it's picked up automatically — nothing else to configure. Anything in your own application that type-hints `Symfony\Contracts\Translation\TranslatorInterface` (as `derafu/translation`'s own exceptions do) resolves to this `Translator` via the alias `translation-services.yaml` provides.

A Note on Rule Names vs. Messages

Rule names (`'email'`, `'max_length'`, the strings you pass to `process()`) are never translated — they're an internal API, not user-facing text. Only the **exception messages** thrown when a rule fails go through translation.

Last updated on 09/09/2026