

Exceptions and Translations

Catch typed value failures, preserve their causes and translate package exceptions

Anonymous · 26/10/2026

Exceptions and Translations

All package exceptions are translatable and can be caught as PHP's `InvalidArgumentException`.

Exception in Derafu\Value\Exception	Parent	Cause
<code>ValueException</code>	<code>TranslatableInvalidArgumentException</code>	Invalid value-type definitions, such as a non-backed enum class.
<code>TypeMismatchException</code>	<code>ValueException</code>	A value does not match a <code>ValueType</code> definition or a strict <code>ValueCaster</code> read.
<code>ValueConversionException</code>	<code>TranslatableInvalidArgumentException</code>	A requested scalar conversion fails.

`ValueConversionException` is separate from `ValueException`. Catch the specific classes or their common `TranslatableInvalidArgumentException` parent when handling both categories.

Strict `ValueCaster::getXXX()` / `requireXXX()` methods reuse `ValueType` validation and raise `TypeMismatchException` for rejected input. If a `Stringable` read fails during conversion, the cause chain contains `ValueConversionException` followed by the original throwable. `ValueType::string()` itself still accepts PHP strings only.

Conversion Failures

```
use Derafu\Value\Exception\ValueConversionException;
use Derafu\Value\ValueCaster;

try {
    ValueCaster::toInt('text');
} catch (ValueConversionException $exception) {
    echo $exception->getMessage();
    // A value of type string cannot be converted to int.
}
```

`toNumber()` propagates the exception from its final `toFloat()` attempt if both conversions fail. A failing Stringable conversion is wrapped with the original throwable available through `getPrevious()`.

Type Failures and Causes

```
use Derafu\Value\Exception\TypeMismatchException;
use Derafu\Value\ValueType;

try {
    ValueType::toInt()->parse('text', 'quantity', 'settings.php');
} catch (TypeMismatchException $exception) {
    echo $exception->getMessage();
    // The value at "quantity" of settings.php must be of type int, string given.

    $cause = $exception->getPrevious(); // ValueConversionException
}
```

Conversion types wrap `ValueConversionException` as `TypeMismatchException` to add location and source context, retaining the original cause. Strict type mismatches do not need an underlying conversion failure. A missing path is displayed as `$`; a source is included only when one is supplied.

Spanish Translations

The package supplies `resources/translations/errors+intl-icu.es.php` through `Derafu\Value\Translation\ValueTranslationResourceProvider`. The provider supplies resources; an application creates and configures its translator.

```
use Derafu\Translation\TranslatorFactory;
use Derafu\Value\Exception\ValueConversionException;
use Derafu\Value\Translation\ValueTranslationResourceProvider;
use Derafu\Value\ValueCaster;

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

try {
    ValueCaster::toInt('text');
} catch (ValueConversionException $exception) {
    echo $exception->trans($translator);
    // Un valor de tipo string no se puede convertir a int.
}
```

Dependency Injection

Applications using Symfony DependencyInjection can import the optional recipe:

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

It tags `ValueTranslationResourceProvider` with `derafu_translation.resource_provider`. Pass providers with that tag to `TranslatorFactory::create()` through `$resourceProviders`, following [Derafu Translation](#). Symfony DependencyInjection is not a runtime requirement of the value utilities.

Last updated on 26/10/2026