

Translations

Translate array validation errors and register the package resource provider

Anonymous · 26/10/2026

Translations

Errors from `ArrayReader` and `ArrayTable` belong to `Derafu\Array\Exception\ArrayException`, which extends `Derafu\Translation\Exception\Logic\TranslatableInvalidArgumentException`. `MissingValueException`, `TypeMismatchException` and `UnexpectedKeyException` specialize array read failures. They can be caught as PHP's `InvalidArgumentException` and translated through [Derafu Translation](#).

```
use Derafu\Array\ArrayReader;

try {
    ArrayReader::fromArray(['port' => 'abc'], 'settings.php')->requireInt('port');
} catch (InvalidArgumentException $exception) {
    echo $exception->getMessage();
    // The key "port" of settings.php must be of type int, string given.
}
```

`ArrayCaster` delegates to `ValueParser` and propagates its `ValueConversionException` when `Stringable` conversion fails. Register `Value`'s provider when translating those exceptions; see [Value Translations](#).

Messages and Context

Message	Cause
The key "{key}" must be of type {expected}, {given} given.	A value has an invalid type.
The key "{key}" is required.	A required read finds a missing or null value.
The key "{key}" is not allowed; the allowed keys are: {allowed}.	<code>assertOnlyKeys()</code> finds an unexpected key.
Row {index} must have exactly 2 columns.	<code>pairsToMap()</code> finds an invalid row.
The first column of row {index} must be a scalar or null to be a key.	A pair has an invalid key value.

Reader and type messages also have variants containing of `{source}`. Nested readers retain the source and append their keys to the diagnostic path. Expected type names come from `ValueType`; `map`

refers to string-keyed arrays. `ArrayReader` wraps `Value` type mismatches while preserving their nested path, element type and original cause. Direct `ValueType` calls use the [Value exception contract](#).

Resource Provider

The package ships Spanish translations in `resources/translations/errors+intl-icu.es.php`, with the provider `Derafu\Array\Translation\ArrayTranslationResourceProvider`.

```
use Derafu\Array\Translation\ArrayTranslationResourceProvider;
use Derafu\Translation\TranslatorFactory;

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

A translatable exception can then be translated with `$exception->trans($translator)`. The resource provider supplies translations; the library does not create a translator automatically.

Dependency Injection

Applications using Symfony DependencyInjection can import:

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

The recipe tags `ArrayTranslationResourceProvider` with `derafu_translation.resource_provider`. Pass providers with that tag to `TranslatorFactory::create()` through its `$resourceProviders` argument, following the setup in [Derafu Translation](#).

Last updated on 26/10/2026