API Reference

API Reference

Anonymous · 09/09/2026

API Reference

Complete reference for every class and interface `derafu/translation` ships.

Interfaces

TranslatableInterface

`Derafu\Translation\Contract\TranslatableInterface`. Extends Symfony's own `Symfony\Contracts\Translation\TranslatableInterface` and adds `Stringable`, so a translatable value can always be cast to a string (the untranslated, eagerly ICU-formatted version) even without a translator.

```
interface TranslatableInterface extends
    \Symfony\Contracts\Translation\TranslatableInterface,
    Stringable
{
    public function __toString(): string;
}
```

`Symfony\Contracts\Translation\TranslatableInterface` itself requires:

```
public function trans(TranslatorInterface $translator, ?string $locale = null):
string;
```

TranslationResourceProviderInterface

`Derafu\Translation\Contract\TranslationResourceProviderInterface`. Declares directories containing translation files, meant to be collected (e.g. via a dependency-injection tagged iterator) so `TranslationResourceRegistrar` can register every directory without a central place having to know about each source in advance.

```
interface TranslationResourceProviderInterface
{
    /**
```

```
     * @return iterable<string> Absolute paths to directories with files
     * named `domain(+intl-icu)?.locale.format`.
     */
    public function getDirectories(): iterable;
}
```

Classes

TranslatableMessage

`Derafu\Translation\TranslatableMessage`. A message with ICU placeholders that can be translated using a translator, or eagerly ICU-formatted (as-is) when no translator is available.

```
final class TranslatableMessage implements TranslatableInterface
{
    /**
     * @param string $message ICU MessageFormat string, used as the
     * translation id and as the fallback text.
     * @param array<string, mixed> $parameters ICU placeholder values.
     * @param string|null $domain Translation domain, null for Symfony's
     * default ('messages').
     * @param string|null $defaultLocale Locale used for eager ICU
     * formatting in __toString() when no translator is available, and as
     * the fallback locale in trans() when none is given.
     */
    public function __construct(
        string $message,
        array $parameters = [],
        ?string $domain = null,
        ?string $defaultLocale = null,
    );

    public function trans(TranslatorInterface $translator, ?string $locale = null):
string;

    /**
     * Formats the raw message via ICU, without a translator. Returns the
     * raw message unchanged if the pattern is invalid.
     */
    public function __toString(): string;
}
```

TranslationResourceRegistrar

`Derafu\Translation\TranslationResourceRegistrar`. Discovers translation files in directories and registers them as resources of a `Symfony\Component\Translation\Translator`, using the

`LoaderInterface`s registered on it. This is the piece `symfony/translation` doesn't provide standalone (directory discovery is normally wired by a full framework).

```
final class TranslationResourceRegistrar
{
    public function __construct(private readonly Translator $translator);

    /**
     * Registers every file directly inside a directory.
     *
     * @throws InvalidArgumentException If the directory doesn't exist, a
     * file's name doesn't follow the `domain.locale.format` convention, or
     * its extension isn't supported.
     */
    public function registerDirectory(string $directory): void;

    /**
     * @param iterable<string> $directories
     */
    public function registerDirectories(iterable $directories): void;

    /**
     * @param iterable<TranslationResourceProviderInterface> $providers
     */
    public function registerFromProviders(iterable $providers): void;

    /** @return array<string> Every locale registered so far. */
    public function getRegisteredLocales(): array;

    /** @return array<string> Every domain registered so far. */
    public function getRegisteredDomains(): array;
}
```

Registration order is precedence order: for the same key in the same domain/locale, the **last** directory registered wins.

Supported file extensions: `yaml`, `yml`, `json`, `php`, `xlf`, `xliff`, `po`, `mo`, `csv`, `ini`.

SimpleTranslationResourceProvider

`Derafu\Translation\SimpleTranslationResourceProvider`. Default implementation of `TranslationResourceProviderInterface` that wraps a fixed list of directories.

```
final class SimpleTranslationResourceProvider implements
TranslationResourceProviderInterface
{
    /** @param iterable<string> $directories */
    public function __construct(private readonly iterable $directories);
```

```
    public function getDirectories(): iterable;
}
```

TranslatorFactory

`Derafu\Translation\TranslatorFactory`. Builds a `Symfony\Component\Translation\Translator` with every file loader `TranslationResourceRegistrar` supports already registered, plus the given resource providers, plus fallback locales.

```
final class TranslatorFactory
{
    /**
     * @param array<string> $fallbackLocales
     * @param iterable<TranslationResourceProviderInterface> $resourceProviders
     * Registered immediately, inside this call, so registration always
     * happens whenever a Translator is built — regardless of whether
     * anything else references a registrar directly (a dependency-injected
     * service nothing depends on is never instantiated by the container).
     */
    public static function create(
        string $defaultLocale,
        array $fallbackLocales = [],
        iterable $resourceProviders = [],
    ): Translator;
}
```

Trait

TranslatableExceptionTrait

`Derafu\Translation\Trait\TranslatableExceptionTrait`. Adds translation support to any exception. Used internally by every exception class listed below — apply it directly only if you can't extend one of them (see Exceptions).

```
trait TranslatableExceptionTrait
{
    protected string $defaultDomain = 'errors';
    protected string $defaultLocale = 'en';

    /**
     * @param string|array|TranslatableInterface $message
     *   - string: used as both message and translation id.
     *   - array: first element is the message/id, the rest are named ICU
     *     parameters.
     *   - TranslatableInterface: used directly.
     * @throws InvalidArgumentException When an empty array is given, or
```

```
     * its first element isn't a string.
     */
    public function __construct(
        string|array|TranslatableInterface $message,
        int $code = 0,
        ?Throwable $previous = null,
    );

    public function trans(TranslatorInterface $translator, ?string $locale = null):
string;

    /** @return array<string, mixed> */
    public function __serialize(): array;

    /** @param array<string, mixed> $data */
    public function __unserialize(array $data): void;
}
```

`__serialize()`/`__unserialize()` intentionally drop the stack trace (it may contain non-serializable values such as closures or resources), so these exceptions can safely be stored in sessions (e.g. for flash messages). After unserializing, `getTrace()` no longer reflects the exception's original call stack.

Exception Hierarchy

Every class below implements `TranslatableInterface` via `TranslatableExceptionTrait`, and mirrors the equivalent native PHP SPL exception:

Class	Extends
`Exception\Core\TranslatableException`	`Exception`
`Exception\Core\TranslatableLogicException`	`LogicException`
`Exception\Core\TranslatableRuntimeException`	`RuntimeException`
`Exception\Logic\TranslatableDomainException`	`DomainException`
`Exception\Logic\TranslatableInvalidArgumentException`	`InvalidArgumentException`
`Exception\Logic\TranslatableLengthException`	`LengthException`
`Exception\Logic\TranslatableOutOfRangeException`	`OutOfRangeException`
`Exception\Runtime\TranslatableOutOfBoundsException`	`OutOfBoundsException`
`Exception\Runtime\TranslatableOverflowException`	`OverflowException`
`Exception\Runtime\TranslatableRangeException`	`RangeException`
`Exception\Runtime\TranslatableUnderflowException`	`UnderflowException`
`Exception\Runtime\TranslatableUnexpectedValueException`	`UnexpectedValueException`

All under the `Derafu\Translation\Exception\` namespace.

Dependencies

`derafu/translation` requires:

- `php: ^8.5`
- `ext-intl` (PHP's ICU bindings, for `\MessageFormatter`)
- `symfony/translation-contracts: ^3.7`
- `symfony/yaml: ^8.1`
- `symfony/translation: ^8.1`

No dependency on `symfony/framework-bundle`, `symfony/http-kernel`, or `symfony/dependency-injection` — the dependency-injection recipe described in Advanced Usage is entirely optional wiring that lives in the consuming application.

Last updated on 09/09/2026