ICU Formatting

ICU Message Formatting Guide

Anonymous · 09/09/2026

ICU Message Formatting Guide

`derafu/translation` doesn't implement its own message formatter. ICU formatting comes directly from PHP's `intl` extension (`\MessageFormatter`), used in two places:

1. **`TranslatableMessage::__toString()`** — formats the raw message eagerly, without a translator. This is what makes `getMessage()` on a translatable exception return a fully formatted string with zero setup.
2. **`symfony/translation`'s own ICU formatter** — used when translating through a real `Translator`, for any domain whose name ends in `+intl-icu` (see Resource Providers and the API Reference).

Both paths accept the exact same ICU MessageFormat syntax described below.

Basic Placeholders

The simplest form uses named placeholders:

```
throw new ValidationException(
    'The field {field} is required.',
    ['field' => 'email'],
);
// getMessage(): "The field email is required."
```

Multiple placeholders are supported:

```
throw new ValidationException(
    'Value {value} for field {field} is invalid.',
    ['value' => 'test@', 'field' => 'email'],
);
// getMessage(): "Value test@ for field email is invalid."
```

Pluralization

```
$id = '{count, plural, =0{No messages} one{# message} other{# messages}}';
```

```
new TranslatableMessage($id, ['count' => 2]); // "2 messages"
new TranslatableMessage($id, ['count' => 1]); // "1 message"
new TranslatableMessage($id, ['count' => 0]); // "No messages"
```

Available plural categories:

- `zero`: for languages with a special zero form.
- `one`: singular form.
- `two`: dual form (for languages that have it).
- `few` / `many`: for languages with special handling of small/large numbers.
- `other`: default form (**required** — always provide it).
- `=n`: exact number matches (e.g. `=0`).

Gender / Select

```
$id = '{gender, select, female {She liked your post} male {He liked your post} other
{They liked your post}}';

(string) new TranslatableMessage($id, ['gender' => 'female']);
// "She liked your post"
```

Nested placeholders work too:

```
$id = '{gender, select, female {{name} added her comment} male {{name} added his
comment} other {{name} added their comment}}';

(string) new TranslatableMessage($id, ['gender' => 'female', 'name' => 'Alice']);
// "Alice added her comment"
```

Number Formatting

```
'{value, number}'          // Plain number.
'{value, number, percent}' // Percentage.
'{value, number, currency}' // Currency, using the message's own locale.
```

Currency formatting depends on the locale carrying a specific currency association (e.g. `en_US`, not just `en`):

```
new TranslatableMessage(
    'Balance must be greater than {min, number, currency}.',
    ['min' => 100],
    null,
    'en_US',
);
```

```
// "Balance must be greater than $100.00."
```

Nested Plural + Select

Patterns can be nested for combined scenarios:

```
$id = '{gender, select,
    female {{count, plural, =0{She has no messages} one{She has # message} other{She
has # messages}}}
    male {{count, plural, =0{He has no messages} one{He has # message} other{He has #
messages}}}
    other {{count, plural, =0{They have no messages} one{They have # message}
other{They have # messages}}}
}';

(string) new TranslatableMessage($id, ['gender' => 'female', 'count' => 5]);
// "She has 5 messages"
```

Common Patterns

```
// Range validation.
'Value must be between {min} and {max}.'

// List validation.
'{count, plural, =0{List cannot be empty} one{At least one item is required} other{At
least # items are required}}'

// Status messages.
'{status, select, pending{Waiting for approval} approved{Approved on {date}}
rejected{Rejected: {reason}} other{Unknown status}}'
```

Troubleshooting

1. **Missing the `other` category** — required for every `select`/`plural` pattern. Without it, a value that doesn't match any listed category falls back to the raw, unformatted message (same as an invalid pattern, see below):

   ```
   // Wrong.
   '{gender, select, male{He} female{She}}'

   // Correct.
   '{gender, select, male{He} female{She} other{They}}'
   ```

2. **Unmatched braces:**

   ```
   // Wrong.
   'Hello {name'

   // Correct.
   'Hello {name}'
   ```

   `TranslatableMessage::__toString()` catches this: an invalid ICU pattern returns the raw message unchanged rather than throwing.

3. **Missing parameters** don't throw either — the placeholder is simply left unexpanded:

   ```
   (string) new TranslatableMessage('{count} items'); // "{count} items"
   ```

For more on ICU MessageFormat syntax:

- ICU User Guide
- PHP `MessageFormatter` manual

Last updated on 09/09/2026