

Formats

Reading and writing a UBL document as json, yaml or xml with Serializer and Deserializer

Anonymous · 12/10/2026

Formats

`Serializer` and `Deserializer` turn a document into a string and back, across three formats: `json`, `yaml` and `xml`. Both resolve the right format handler by name internally, so the same two classes are the only thing a consumer ever needs, regardless of format.

Supported Formats

Format	Dependency	Notes
<code>json</code>	PHP built-in	Always available.
<code>yaml</code>	<code>symfony/yaml</code>	Optional: see Optional Dependencies below.
<code>xml</code>	<code>xterr/php-ubl</code>	The real UBL XML, checked by Validation .

Serializing a Document

```
use Derafu\Ubl\Service\Serializer;

$serializer = new Serializer();

$yaml = $serializer->serialize($quotation, 'yaml');
$json = $serializer->serialize($quotation, 'json');
$xml = $serializer->serialize($quotation, 'xml');
```

Deserializing a Document

```
use Derafu\Ubl\Service\Deserializer;

$deserializer = new Deserializer();

$quotation = $deserializer->deserialize($yaml, 'yaml');
```

An explicit target class can be passed as a third argument to skip or override the automatic detection described below:

```
use Derafu\Ubl\Oasis\Doc\Quotation;

$quotation = $deserializer->deserialize($yaml, 'yaml', Quotation::class);
```

Self-Describing Content

Neither `json` nor `yaml` needs an explicit target class to know what to deserialize into: the single top-level key of the decoded content names the document type directly, e.g. `{"Quotation": {...}}` or `Quotation:` at the top of a YAML file. `xml` works the same way through its own mechanism: the document type is read from the XML root element (`<Quotation xmlns="...:Quotation-2">`).

`Serializer::serialize()` always wraps `json/yaml` output under that same top-level key (never a bare array), so what it returns for one format is exactly what `Deserializer::deserialize()` accepts back for that same format.

Value and Attribute Representation

Every class also has `toArray()/fromArray()/jsonSerialize()`, used internally by the `json` and `yaml` handlers and available directly to a consumer who wants a plain array with no format involved:

```
$array = $quotation->toArray();
$quotation = Quotation::fromArray($array);
```

Keys are the UBL element and attribute names, not PHP property names. A value with no attributes is a plain scalar:

```
['ID' => 'Q-001', 'IssueDate' => '2026-01-15']
```

A value that carries an attribute, like an amount's currency, uses the `@value/@attributes` form instead:

```
[
    '@value' => '100',
    '@attributes' => ['currencyID' => 'EUR'],
]
```

Optional Dependencies

`symfony/yaml` is not a hard dependency of this package: the `yaml` handler is only autoloaded the first time a caller actually serializes or deserializes with `'yaml'`. A consumer who never touches that format never needs it installed; one who does gets PHP's own "Class not found" error if it is missing,

with no extra exception type or `class_exists()` guard in between.

Last updated on 12/10/2026