

Docs

UBL Project

Derafu UBL

Anonymous · 12/10/2026 · #php

Table of Contents

- UBL Project 3
 - Introduction* 4
 - Document Types* 9
 - Formats* 14
 - Validation* 17
 - Services* 19

Docs » Data Handling Category » UBL Project

UBL Project

Derafu UBL

Anonymous · 12/10/2026 · #php

Derafu UBL

Last updated on 12/10/2026

Introduction

Typed PHP model of UBL 2.1 (Universal Business Language) to read, write and validate business documents against the official XSD

Anonymous · 12/10/2026

One Language for Every Business Document

github derafu/ubl last commit today CI passing code size 3.8 MiB open issues 0 downloads 4

downloads 4 this month

Derafu UBL is a typed PHP model of [UBL](#) (Universal Business Language): one PHP class per document type and per building block defined by the official UBL 2.1 XSD, with services to read and write those documents in multiple formats and validate them against the real schema.

It is deliberately generic: nothing in it knows about any one business (no fixed seller, no fixed tax rate, no fixed currency) and nothing about any e-invoicing profile (no Peppol rules, no country-specific mandatory fields or codelists). That kind of logic belongs in a package built on top of this one, for whichever network or country it targets.

Features

- **One class per UBL 2.1 type:** every document (Quotation, Invoice, CreditNote, ...) and every shared building block (Party, MonetaryTotal, TaxTotal, every *Line type) as a plain, typed PHP class.
- **Three wire formats:** serialize and deserialize the same document as json, yaml or xml, through one consistent API.
- **Schema validation:** validate a document's XML against the official OASIS UBL 2.1 XSD bundled with the package.
- **Array/JSON native:** every class also has toArray()/fromArray()/jsonSerialize(), for working with a document as plain data with no XML involved at all.

Installation

```
composer require derafu/ubl
```

Applying the Bundled Patch to xterr/php-ubl

derafu/ubl wraps xterr/php-ubl for xml, and ships a fix for a few of its known defects as a script that patches it directly in vendor/ (tools/patch-xterr-ubl.php). That script is wired as derafu/ubl's own

Composer `post-install-cmd/post-update-cmd`, so it reapplies automatically whenever `derafu/ubl` itself is installed or updated as the root project.

Composer does not run a dependency's own scripts for the project that requires it - only the root project's own `composer.json` scripts run. A project that depends on `derafu/ubl` must add the same command to its own `composer.json` to actually get the patch; otherwise it never runs, and `xterr/php-ubl` keeps its original, unpatched behavior silently.

`post-install-cmd/post-update-cmd` each accept either a single command or an array of them - use the array form, so this is added alongside whatever else is already there instead of replacing it:

```
{
  "scripts": {
    "post-install-cmd": [
      "php vendor/derafu/ubl/tools/patch-xterr-ubl.php"
    ],
    "post-update-cmd": [
      "php vendor/derafu/ubl/tools/patch-xterr-ubl.php"
    ]
  }
}
```

Run `composer install` (or `composer update`) once more after adding it, so the patch also applies to the `xterr/php-ubl` already sitting in `vendor/`.

Basic Usage

Building a Document

Using an object-oriented API:

```
use Derafu\Ubl\Oasis\Cac\Item;
use Derafu\Ubl\Oasis\Cac\LineItem;
use Derafu\Ubl\Oasis\Cac\MonetaryTotal;
use Derafu\Ubl\Oasis\Cac\Party;
use Derafu\Ubl\Oasis\Cac\PartyName;
use Derafu\Ubl\Oasis\Cac\Price;
use Derafu\Ubl\Oasis\Cac\QuotationLine;
use Derafu\Ubl\Oasis\Cac\SupplierParty;
use Derafu\Ubl\Oasis\Cbc\Amount;
use Derafu\Ubl\Oasis\Cbc\Identifier;
use Derafu\Ubl\Oasis\Cbc\Text;
use Derafu\Ubl\Oasis\Doc\Quotation;

$quotation = (new Quotation())
    ->setId((new Identifier())->setValue('Q-001'))
```

```

->setIssueDate(new DateTimeImmutable('2026-01-15'))
->setSellerSupplierParty(
    (new SupplierParty()->setParty(
        (new Party()->setPartyNames([
            (new PartyName()->setName((new Text()->setValue('Acme Corp'))),
        ])
    ))
)
->setQuotedMonetaryTotal(
    (new MonetaryTotal()->setPayableAmount(
        (new Amount()->setValue('100')->setCurrencyID('EUR'))
    )
)
->setQuotationLines([
    (new QuotationLine()
        ->setId((new Identifier()->setValue('1'))
        ->setLineItem(
            (new LineItem()
                ->setId((new Identifier()->setValue('1'))
                ->setPrice(
                    (new Price()->setPriceAmount(
                        (new Amount()->setValue('100')->setCurrencyID('EUR'))
                    )
                )
                ->setItem((new Item()->setName((new Text()->setValue('Widget'))))
            ),
        ),
    ]));

```

Only ID, IssueDate, SellerSupplierParty, QuotedMonetaryTotal and one QuotationLine (with LineItem.ID and LineItem.Item) are actually required: true in UBL - an Item with no Name, and a line with no Price, are schema-valid too. The seller's and item's names, and the price, are kept here only so the example reads like a real quotation, the same choice behind this package's own committed Quotation fixture.

Or using a plain PHP array - the exact same shape toArray()/jsonSerialize() produce, and json/yaml wrap under the document type's name (see Formats):

```

use Derafu\Ubl\Oasis\Doc\Quotation;

$quotation = Quotation::fromArray([
    'ID' => 'Q-001',
    'IssueDate' => '2026-01-15',
    'SellerSupplierParty' => [
        'Party' => [
            'PartyName' => [
                ['Name' => 'Acme Corp'],
            ],
        ],
    ],
]);

```

```

    'QuotedMonetaryTotal' => [
        'PayableAmount' => [
            '@value' => '100',
            '@attributes' => ['currencyID' => 'EUR'],
        ],
    ],
    'QuotationLine' => [
        [
            'ID' => '1',
            'LineItem' => [
                'ID' => '1',
                'Price' => [
                    'PriceAmount' => [
                        '@value' => '100',
                        '@attributes' => ['currencyID' => 'EUR'],
                    ],
                ],
                'Item' => [
                    'Name' => 'Widget',
                ],
            ],
        ],
    ],
]);

```

Serializing and Deserializing

```

use Derafu\Ubl\Service\Deserializer;
use Derafu\Ubl\Service\Serializer;

$serializer = new Serializer();
$deserializer = new Deserializer();

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

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

```

See **Formats** for the exact shape of each one.

Validating Against the Official Schema

```

use Derafu\Ubl\Exception\ValidationException;
use Derafu\Ubl\Service\SchemaValidator;

try {
    (new SchemaValidator())->validate($xml);
} catch (ValidationException $e) {

```

```
foreach ($e->getErrors() as $error) {  
    echo $error->message, "\n";  
}  
}
```

See **Validation** for how the document type is resolved and what else can be inspected on a failure.

What This Package Is Not

- **Not a profile.** PEPPOL BIS, EN 16931, and any country's CIUS (Core Invoice Usage Specification) restrict or extend this base syntax with their own mandatory fields, codelists and Schematron business rules. Those are separate packages that depend on this one, never the other way around.
- **Not a mapping layer.** Turning your own application's data (prices, customers, products) into one of these documents is the job of your application, not of this package.

Last updated on 12/10/2026

Document Types

The 65 UBL 2.1 document types this package models, one PHP class per document

Anonymous · 12/10/2026

Document Types

Every document type defined by UBL 2.1 is available as a class under `Derafu\Ubl\Oasis\Doc\`, named exactly like the document (`Quotation`, `Invoice`, `CreditNote`, ...). There is no sub-namespace per business domain: most of the underlying building blocks (`Party`, `MonetaryTotal`, `TaxTotal`, every `*Line` type) are shared across nearly all of them regardless of domain, and a document does not belong to a single domain either (an `Invoice` is billing, but it is referenced from other flows through `BillingReference`).

Available Documents

Class	Definition
<code>ApplicationResponse</code>	A document to indicate the application's response to a transaction.
<code>AttachedDocument</code>	A wrapper that allows a document of any kind to be packaged with the UBL document that references it.
<code>AwardedNotification</code>	The document used to communicate a contract award to the winner.
<code>BillOfLading</code>	A document issued by the party who acts as an agent for a transportation carrier or other agents to the party who gives instructions for the transportation services (shipper, consignor, etc.) stating the details of the transportation, charges, and terms and conditions under which the transportation service is provided.
<code>CallForTenders</code>	A document used by a Contracting Party to define a procurement project to buy goods, services, or works during a specified period.
<code>Catalogue</code>	A document that describes items, prices, and price validity.
<code>CatalogueDeletion</code>	A document used to cancel an entire Catalogue.
<code>CatalogueItemSpecificationUpdate</code>	A document used to update information (e.g., technical descriptions and properties) about Items in an existing Catalogue.

Class	Definition
CataloguePricingUpdate	A document used to update information about prices in an existing Catalogue.
CatalogueRequest	A document used to request a Catalogue.
CertificateOfOrigin	A document that describes the Certificate of Origin.
ContractAwardNotice	A document published by a Contracting Party to announce the awarding of a procurement project.
ContractNotice	A document used by a Contracting party to announce a project to buy goods, services, or works.
CreditNote	A document used to specify credits due to the Debtor from the Creditor.
DebitNote	A document used to specify debts incurred by the Debtor.
DespatchAdvice	A document used to describe the despatch or delivery of goods and services.
DocumentStatus	A document used to provide information about document status.
DocumentStatusRequest	A document used to request the status of another document.
ExceptionCriteria	A document used to specify the thresholds for forecast variance, product activity, and performance history beyond which exceptions should be triggered.
ExceptionNotification	A document used to notify an exception in forecast variance, product activity, or performance history.
Forecast	A document used to forecast sales or orders.
ForecastRevision	A document used to revise a Forecast.
ForwardingInstructions	A document issued to a forwarder, giving instructions regarding the action to be taken for the forwarding of goods described therein.
FreightInvoice	A document stating the charges incurred for a logistics service.
FulfilmentCancellation	A document used to cancel an entire fulfilment document (Despatch Advice or Receipt Advice).
GoodsItemItinerary	A document providing details relating to a transport service, such as transport movement, identification of equipment and goods, subcontracted service providers, etc.
GuaranteeCertificate	A document to notify the deposit of a bid bond guarantee.
InstructionForReturns	A document used to initiate a return of goods.

Class	Definition
InventoryReport	A report on the quantities of each item that are, or will be, in stock.
Invoice	A document used to request payment.
ItemInformationRequest	A document used to request product activity, forecast, or performance data.
Order	A document used to order goods and services.
OrderCancellation	A document used to cancel an entire Order.
OrderChange	A document used to specify changes to an existing Order.
OrderResponse	A document used to indicate detailed acceptance or rejection of an Order or to make a counter-offer.
OrderResponseSimple	A document used to indicate simple acceptance or rejection of an entire Order.
PackingList	A document describing how goods are packed.
PriorInformationNotice	A document used by a contracting party to declare the intention to buy goods, services, or works during a specified period.
ProductActivity	A document reporting the movement of goods at specified retail locations for inventory tracking purposes.
Quotation	A document used to quote for the provision of goods and services.
ReceiptAdvice	A document used to describe the receipt of goods and services.
Reminder	A document used to remind a customer of payments past due.
RemittanceAdvice	A document that specifies details of an actual payment.
RequestForQuotation	A document used to request a Quotation for goods and services from a Seller.
RetailEvent	A document used to specify basic information about retail events (such as promotions, product introductions, and community or environmental events) that affect supply or demand.
SelfBilledCreditNote	A credit note created by the debtor in a self billing arrangement with a creditor; Self Billed Credit Note replaces Debit Note in such arrangements.
SelfBilledInvoice	An Invoice document created by the Customer (rather than the Supplier) in a Self Billing relationship.
Statement	A document used to report the status of orders, billing, and payment.

Class	Definition
StockAvailabilityReport	A report on the quantities of each item that are, or will be, in stock.
Tender	A document whereby an economic operator (the tenderer) makes a formal offer (the tender) to a contracting authority to execute an order for the supply or purchase of goods, or for the execution of work, according to the terms of a proposed contract.
TenderReceipt	A document sent by a contracting party to an economic operator acknowledging receipt of a Tender.
TendererQualification	A document declaring the qualifications of a tenderer.
TendererQualificationResponse	A document issued by a procurement organization to notify an economic operator whether it has been admitted to or excluded from the tendering process.
TradeItemLocationProfile	A document specifying trade item attributes relating to replenishment policies.
TransportExecutionPlan	A document used in the negotiation of a transport service between a transport user and a transport service provider.
TransportExecutionPlanRequest	A document sent by a transport user to request a transport service from a transport service provider.
TransportProgressStatus	A document sent from a Transportation Network Manager to a Transport Service Provider giving the status of the whereabouts and schedule of the transport means involved in a transport service.
TransportProgressStatusRequest	A document sent from a transport service provider to a transportation network manager requesting a Transport Progress Status.
TransportServiceDescription	A document sent by a transport service provider to announce the availability of a transport service.
TransportServiceDescriptionRequest	A document requesting a Transport Service Description, sent from a party with a transport demand (transport user) to a party providing transport services (transport service provider).
TransportationStatus	A document to circulate reports of transportation status or changes in status (events) among a group of participants.
TransportationStatusRequest	A document requesting a Transportation Status report.
UnawardedNotification	A document communicating to a tenderer that the contract has been awarded to different tenderer.

Class	Definition
<code>UtilityStatement</code>	A supplement to an Invoice or Credit Note, containing information on the consumption of services provided by utility suppliers to private and public customers, including electricity, gas, water, and telephone services.
<code>Waybill</code>	A transport document describing a shipment.

Working with a Document

A document is a plain object: build it with its setters, or hydrate one from data with `fromArray()`. See **Introduction** for a complete example, and **Formats** for reading and writing documents as `json`, `yaml` or `xml`.

Last updated on 12/10/2026

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

Docs » Data Handling Category » UBL Project » Validation

Validation

Validating a UBL document's XML against the official OASIS UBL 2.1 XSD

Anonymous · 12/10/2026

Validation

`SchemaValidator` validates a document's XML directly against the official OASIS UBL 2.1 XSD bundled with the package - the same schema real UBL consumers (Peppol Access Points, tax authorities) validate against.

Validating a Document

```
use Derafu\Ubl\Service\SchemaValidator;

$validator = new SchemaValidator();

$validator->validate($xml); // Throws Derafu\Ubl\Exception\ValidationException if
invalid.
```

Nothing is returned on success: a document either validates, or the call throws.

Resolving the Document Type

Which XSD file to validate against is resolved automatically from the XML's own root element, the same way `Deserializer` resolves a target class for `xml` (see Self-Describing Content in **Formats**). An explicit class can be passed as a second argument to skip or override that:

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

$validator->validate($xml, Quotation::class);
```

Handling Validation Errors

A failed validation throws `Derafu\Ubl\Exception\ValidationException`, which exposes the real list of `LibXML_Error` objects `libxml` reported, not only a flat message:

```
use Derafu\Ubl\Exception\ValidationException;
```

```
try {  
    $validator->validate($xml);  
} catch (ValidationException $e) {  
    foreach ($e->getErrors() as $error) {  
        printf("Line %d: %s\n", $error->line, trim($error->message));  
    }  
}
```

A malformed XML string, or a well-formed one whose root element matches no known UBL document, also throws `ValidationException` - with `getErrors()` empty in both cases, since neither one ever reaches schema validation itself: the document type could not be resolved to begin with.

Last updated on 12/10/2026

Docs » Data Handling Category » UBL Project » Services

Services

The services this package provides, their interfaces, exceptions, and how to wire them with Symfony's dependency injection

Anonymous · 12/10/2026

Services

Available Services

Interface	Class	Purpose
<code>Contract\SerializerInterface</code>	<code>Service\Serializer</code>	Turn a document into a <code>json/yaml/xml</code> string.
<code>Contract\DeserializerInterface</code>	<code>Service\Deserializer</code>	Turn a <code>json/yaml/xml</code> string into a document.
<code>Contract\ValidatorInterface</code>	<code>Service\SchemaValidator</code>	Validate a document's XML against the official XSD.
<code>Contract\FormatHandlerFactoryInterface</code>	<code>Service\FormatHandlerFactory</code>	Resolve a format name to the handler that implements it; register a custom one.

Every class is namespaced under `Derafu\Ubl\`. See **Formats** for `Serializer/Deserializer` usage and **Validation** for `SchemaValidator`.

Exceptions

Every exception this package throws extends `Derafu\Ubl\Exception\UblException`, itself a `Derafu\Translation\Exception\Core\TranslatableException`:

Exception	Thrown when
<code>UnsupportedFormatException</code>	A format name is not <code>json</code> , <code>yaml</code> or <code>xml</code> (or a custom registered one).
<code>SerializationException</code>	A document cannot be turned into the requested format.
<code>DeserializationException</code>	Content cannot be turned into a document (invalid JSON/YAML, an unknown document type, ...).
<code>ValidationException</code>	A document's XML does not validate against the schema. Carries the real <code>LibXML_Error</code> list, see Validation .

Catching the base `UblException` catches any of them:

```
use Derafu\Ubl\Exception\UblException;

try {
    $document = (new Deserializer())->deserialize($content, 'yaml');
} catch (UblException $e) {
    // SerializationException, DeserializationException, ValidationException, ...
}
```

Dependency Injection

For an application using `symfony/dependency-injection`, `resources/config/ubl-services.yaml` wires every service above to its interface, ready to autowire:

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

With that imported, a service can simply type-hint the interface it needs:

```
use Derafu\Ubl\Contract\SerializerInterface;

final class QuotationController
{
    public function __construct(
        private readonly SerializerInterface $serializer,
    ) {
    }
}
```

Last updated on 12/10/2026