Signature

Signing and validating UBL documents with the UBL Digital Signature Profile of OASIS, enveloped and detached, with Signer and SignatureValidator

Anonymous · 09/11/2026

Signature

`Signer` signs a UBL document with the UBL Digital Signature Profile of OASIS, and `SignatureValidator` validates the signatures of a document and checks that it follows what the profile requires. The text of the profile is the same for UBL 2.1 to 2.5, so they work with every version of the package.

The signatures are made and validated by `derafu/signature`, an implementation of XML Signature that knows nothing about UBL. This package adds what is of UBL: where the signature goes in the document, the elements that describe it and the transforms that the profile defines.

Signing a Document

```
use Derafu\Signature\Service\SignatureGenerator;
use Derafu\Signature\Service\SignatureValidator as XmlSignatureValidator;
use Derafu\Ubl\Service\SignatureValidator;
use Derafu\Ubl\Service\Signer;
use Derafu\Xml\Service\XmlDecoder;
use Derafu\Xml\Service\XmlEncoder;
use Derafu\Xml\Service\XmlService;
use Derafu\Xml\Service\XmlValidator;

$xmlService = new XmlService(new XmlEncoder(), new XmlDecoder(), new XmlValidator());
$generator = new SignatureGenerator($xmlService);

$signer = new Signer($generator);
$validator = new SignatureValidator(new XmlSignatureValidator($generator,
$xmlService));

// $certificate is a Derafu\Certificate\Contract\CertificateInterface.
$signed = $signer->sign($xml, $certificate, ['version' => '2.1']);

$report = $validator->validate($signed);
$report->isValid();
```

The document is an object of the model, an `XmlDocument` of `derafu/xml` or its XML as a string. An

`XmlDocument` is signed as the XML that it saves, with its own format settings: its format is not changed. The result is the XML of the signed document, valid against the schema of its version.

What the Signer Writes

The signature goes in an `ext:UBLExtension` of the document, in the way that the profile says:

```
<ext:UBLExtensions>
  <ext:UBLExtension>
<ext:ExtensionURI>urn:oasis:names:specification:ubl:dsig:enveloped</ext:ExtensionURI>
    <ext:ExtensionContent>
      <sig:UBLDocumentSignatures>
        <sac:SignatureInformation>
          <cbc:ID>urn:oasis:names:specification:ubl:signature:1</cbc:ID>
<sbc:ReferencedSignatureID>urn:oasis:names:specification:ubl:signature:Invoice</sbc:Re
ferencedSignatureID>
          <ds:Signature Id="dsig-1">...</ds:Signature>
        </sac:SignatureInformation>
      </sig:UBLDocumentSignatures>
    </ext:ExtensionContent>
  </ext:UBLExtension>
</ext:UBLExtensions>
...
<cac:Signature>
  <cbc:ID>urn:oasis:names:specification:ubl:signature:Invoice</cbc:ID>
<cbc:SignatureMethod>urn:oasis:names:specification:ubl:dsig:enveloped</cbc:SignatureMe
thod>
  <cac:SignatoryParty>
    <cac:PartyName><cbc:Name>Common name of the certificate</cbc:Name></cac:PartyName>
  </cac:SignatoryParty>
</cac:Signature>
```

The `ext:UBLExtensions` is the first element of the document and the `cac:Signature` is where the schema of each document type has it (it is not the same in an invoice and in an order): the document is built with the model, so the position is the one of the schema. What is signed is the text of the XML that results, and it is not written again afterwards.

The `ds:Signature` signs the whole document except the `sig:UBLDocumentSignatures`, with SHA-256 and the algorithm of the key of the certificate (RSA, ECDSA or DSA), the elements with the prefix `ds`, and an `Id` in the signature and in its value, so another signature can countersign it.

Options

The third argument of `sign()` is an array of options, or an object that implements `Derafu\Config\Contract\OptionsInterface`. They are valid only for that signature: the ones that are not given have their default value.

Option	Default	Description
`version`	the default of the registry	The version of UBL of the XML of the document.
`xpath`	`flexible`	The transform of the signature: `flexible`, `final` or `enveloped` (see below).
`include_signature`	`true`	Whether the document has the `cac:Signature` of the signature.
`signature_id`	`urn:oasis:names:specification:ubl:signature:` and the name of the document	The `cbc:ID` of the `cac:Signature`, that the `sbc:ReferencedSignatureID` has.
`information_id`	`urn:oasis:names:specification:ubl:signature:` and its position	The `cbc:ID` of the `sac:SignatureInformation`.
`extension_id`	none	The `cbc:ID` of the `ext:UBLExtension`.
`extension_uri`	`urn:oasis:names:specification:ubl:dsig:enveloped`	The `ext:ExtensionURI`.
`signatory`	none	The `cac:SignatoryParty`: `id` and `name`. Without them the party has no identification and its name is the common name of the certificate.
`countersign`	none	The `Id` of the signatures or signature values that this signature signs too.
`references`	none	More references of the signature, as the ones of `derafu/signature`.
`signature`	none	The options of the signature of `derafu/signature`: the algorithms, the prefix, the `Id` of the signature, the `Object` elements... but `reference`, `references`, `signature_location` and `resource_resolver`, that the signer decides.

An option that is not valid is a `UblException`: `The options of the signature are not valid: ...`.

The Transform

The profile recommends one of two transforms for the reference to the document:

- `flexible` (default): leaves out every `sig:UBLDocumentSignatures` of the document, so other signatures can be added afterwards without invalidating this one.
- `final`: leaves out only this signature, so another signature added afterwards invalidates it.

`enveloped` is the enveloped signature transform of XML Signature (the whole document without the signature), that is not one of the profile. A change in the document that is outside the `sig:UBLDocumentSignatures` invalidates the signature with any of them.

Other Signatures

A document that has signatures is given as XML: it is signed as it is, adding a `sac:SignatureInformation` to the `sig:UBLDocumentSignatures` that it has. Nothing else of the document changes, so the signatures that it has stay valid if their transform is `flexible`. A document that the model holds and that has signatures inside is not signed from the model: its text would change.

```
// A co-signature.
$both = $signer->sign($signed, $otherCertificate);

// A countersignature: it signs the document and the value of the first signature.
$countersigned = $signer->sign($signed, $otherCertificate, ['countersign' => 'dsig-1-
value']);
```

The `cac:Signature` of the new signature is not added, because adding it changes the document and invalidates the signatures that it has. To plan a co-signature, add the `cac:Signature` elements to the document before the first signature, and reference one of them with `signature_id`.

Detached Signatures

`signDetached()` signs the document with a signature that is in another document. The document gets a `cac:Signature` that says so and where the signature is:

```
$detached = $signer->signDetached($xml, $certificate, [
    'version' => '2.1',
    'document_uri' => 'invoice.xml',
    'signature_uri' => 'invoice-signature.xml',
]);

file_put_contents('invoice.xml', $detached->document);
file_put_contents('invoice-signature.xml', $detached->signature);
```

`document_uri` is the `URI` that the reference of the signature has for the document, and `signature_uri` is the one that the `cac:Signature` of the document has for the signature. The document that is returned is what the signature signs, so it has to be kept as it is. The signature leaves out the `sig:UBLDocumentSignatures` of the document (`count(ancestor-or-self::sig:UBLDocumentSignatures)=0`) and canonicalizes it.

A document that has signatures inside can not get a detached one. It takes the same options as `sign()`, but `countersign`.

Validating

```
$report = $validator->validate($signed);

$report->getSignatures();    // The result of each signature (of derafu/signature).
$report->areSignaturesValid();
$report->getIssues();        // What the document does not follow of the profile.
$report->isConformant();
$report->isValid();          // Valid signatures and conformant.

// A detached signature, with the document it signs.
$report = $validator->validateDetached($document, $signature, ['document_uri' =>
'invoice.xml']);
```

The XML is validated as it is given, as an `XmlDocument` (as the XML that it saves) or as a string. A signature that is not valid is a result, not an exception: `validate()` throws only if the XML is malformed or has no signatures (`Derafu\Signature\Exception\SignatureException`) or an option is not valid (`UblException`).

The document is checked against what the profile requires of it, not against what it recommends. These are its issues, each one a `UblException`:

- An extension with signatures has no `ext:ExtensionURI`, or it is not one of the profile (`urn:oasis:names:specification:ubl:dsig:enveloped` or `...:enveloped:xades`).
- A `sbc:ReferencedSignatureID` is not the `cbc:ID` of a `cac:Signature` of the document.
- The `cbc:SignatureMethod` of the `cac:Signature` that a signature references is not one of the enveloped profile.
- The `Id` of a `ds:Signature` or of its `ds:SignatureValue` (the one that lets another signature countersign it) is repeated among the `Id` of the document.
- In a detached signature, the `cbc:SignatureMethod` of a `cac:Signature` with an attachment is not one of the detached profile (`urn:oasis:names:specification:ubl:dsig:detached` or `...:detached:xades`).

A signature is never rejected for not following what the profile recommends, like the transform that it uses.

Validate the Text that Was Signed

The signature covers the text of the whole document. A document that is read into the model and written again is not that text (the model does not keep the spaces between the elements, for example), so its signature is not valid. A signed document is validated from its XML, as it was signed, and not from what the model writes from it. The signature that the document has inside is kept as it is when the model reads and writes it.

XAdES

The XAdES profile (`urn:oasis:names:specification:ubl:dsig:enveloped:xades`) is not made by this package. What is needed to make it with another one is in the options: `extension_uri` for the URI, `signature.objects` for the `Object` with the qualifying properties, and `references` for the reference to its signed properties.

Dependency Injection

`resources/config/ubl-signature-services.yaml` wires the signer and the validator. Import it after `ubl-services.yaml`; the app has to have the services of `derafu/signature` (`SignatureGeneratorInterface` and `SignatureValidatorInterface`):

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

Interface	Class
`Contract\SignerInterface`	`Service\Signer`
`Contract\SignatureValidatorInterface`	`Service\SignatureValidator`

Last updated on 09/11/2026