

# Signature Class

Signature Class

Anonymous · 21/10/2026

## Signature Class

The `Signature` class is the core component of the Derafu Signature library, representing an XML digital signature node according to the XML-DSIG standard.

### Overview

The `Signature` class implements the `SignatureInterface` and represents the XML `<Signature>` element that contains all the information related to a digital signature in an XML document:

- The digest value of the signed content.
- The signature value.
- The public key information for signature verification.

This class is used both when creating new signatures and when validating existing ones.

### Usage

#### Creating a New Signature Node

```
use Derafu\Signature\Signature;
use Derafu\Certificate\Service\CertificateLoader;

// Load a certificate.
$certificateLoader = new CertificateLoader();
$certificate = $certificateLoader->loadFromFile(
    '/path/to/certificate.p12',
    'password'
);

// Create a signature node.
$signatureNode = new Signature();

// Configure with digest value, certificate, and optional reference.
$signatureNode->configureSignatureData(
    digestValue: 'bWFpbkRpZ2VzdFZhbHVlQmFzZTY0',
```

```
certificate: $certificate,  
reference: 'documentId' // Optional, specify to sign a specific element.  
);
```

## Working with an Existing Signature Node

```
// Typically you'd use the SignatureValidator to create a signature node from XML.  
$signatureNode = $signatureService->createSignatureNode($signatureXml);  
  
// Access signature properties.  
$reference = $signatureNode->getReference();  
$digestValue = $signatureNode->getDigestValue();  
$signatureValue = $signatureNode->getSignatureValue();  
$x509Certificate = $signatureNode->getX509Certificate();
```

## API Reference

---

### Setting and Getting Data

```
// Set raw data structure.  
$signatureNode->setData($dataArray);  
  
// Get the current data structure.  
$dataArray = $signatureNode->getData();
```

### Configuring the Signature

```
// Configure all signature components at once.  
$signatureNode->configureSignatureData(  
    digestValue: 'base64DigestValue',  
    certificate: $certificate,  
    reference: 'elementId'  
);
```

### Working with XML

```
// Set the XML representation of the signature.  
$signatureNode->setXml($xmlDocument);  
  
// Get the XML representation of the signature.  
$xmlDocument = $signatureNode->getXml();
```

## Setting Signature Value

```
// Set the calculated signature value (after signing the SignedInfo element).
$signatureNode->setSignatureValue('base64SignatureValue');
```

## Getting Signature Components

```
// Get the reference URI (without # prefix).
$reference = $signatureNode->getReference();

// Get the digest value.
$digestValue = $signatureNode->getDigestValue();

// Get the X.509 certificate (without headers/footers).
$certificate = $signatureNode->getX509Certificate();

// Get the signature value.
$signatureValue = $signatureNode->getSignatureValue();
```

## Data Structure

The Signature class maintains an internal data array that represents the XML structure of the signature. This structure follows the XML-DSIG standard:

```
[
  'Signature' => [
    '@attributes' => [
      'xmlns' => 'http://www.w3.org/2000/09/xmldsig#',
    ],
    'SignedInfo' => [
      '@attributes' => [
        'xmlns:xsi' => 'http://www.w3.org/2001/XMLSchema-instance',
      ],
      'CanonicalizationMethod' => [
        '@attributes' => [
          'Algorithm' => 'http://www.w3.org/TR/2001/REC-xml-c14n-20010315',
        ],
      ],
      'SignatureMethod' => [
        '@attributes' => [
          'Algorithm' => 'http://www.w3.org/2000/09/xmldsig#rsa-sha1',
        ],
      ],
      'Reference' => [
        '@attributes' => [
          'URI' => '', // Reference URI, empty for entire document.
        ],
      ],
    ],
  ],
]
```

```

        'Transforms' => [
            'Transform' => [
                '@attributes' => [
                    'Algorithm' =>
'http://www.w3.org/2000/09/xmldsig#enveloped-signature',
                ],
            ],
        ],
        'DigestMethod' => [
            '@attributes' => [
                'Algorithm' => 'http://www.w3.org/2000/09/xmldsig#sha1',
            ],
        ],
        'DigestValue' => '', // Will contain the digest value.
    ],
],
'SignatureValue' => '', // Will contain the signature value.
'KeyInfo' => [
    'KeyValue' => [
        'RSAKeyValue' => [
            'Modulus' => '', // Will contain the certificate modulus.
            'Exponent' => '', // Will contain the certificate exponent.
        ],
    ],
    'X509Data' => [
        'X509Certificate' => '', // Will contain the certificate.
    ],
],
],
]

```

## Implementation Details

### XML Invalidation

The `Signature` class automatically invalidates the XML representation when data is modified:

```

// This will cause the internal XML to be invalidated.
$signatureNode->setData($newData);

// This will also invalidate the XML.
$signatureNode->setSignatureValue($newSignatureValue);

```

After invalidation, the XML must be regenerated (typically by the `SignatureGenerator` class) before `getXml()` can be called again.

## Reference URIs

Reference URIs are handled according to the XML-DSIG standard:

- An empty URI ("") means the entire document is signed.
- A URI starting with # refers to an element with the specified ID.
- The `getReference()` method returns the reference without the # prefix.
- The `configureSignatureData()` method automatically adds the # prefix if not present.

## Transformation Algorithm

The transformation algorithm changes based on whether a reference is provided:

- With a reference: `http://www.w3.org/TR/2001/REC-xml-c14n-20010315` (standard C14N).
- Without a reference: `http://www.w3.org/2000/09/xmlsig#enveloped-signature` (enveloped signature transformation).

This ensures that the signature is correctly calculated for both whole-document signatures and element signatures.

---

Last updated on 21/10/2026