

OpenSSL Service

The service that calls OpenSSL, the only one in the package, the errors that it takes from each call and how to replace it

Anonymous · 17/10/2026

OpenSsl Service

Derafu\Certificate\Service\OpenSsl is the only code of the package that calls the functions of OpenSSL. It implements Derafu\Certificate\Contract\OpenSslInterface. The loader, the faker, the certificate and the self-signed certificate use it, so what is common to all the calls is done once.

The Methods

Each method is the function of OpenSSL of the same name, without the `openssl_` prefix:

Method	Function	Gives
<code>pkeyNew()</code>	<code>openssl_pkey_new()</code>	A new private key.
<code>pkeyGetPrivate()</code>	<code>openssl_pkey_get_private()</code>	A private key, from its PEM.
<code>pkeyGetPublic()</code>	<code>openssl_pkey_get_public()</code>	The public key of a certificate.
<code>pkeyGetDetails()</code>	<code>openssl_pkey_get_details()</code>	The details of a key.
<code>x509Parse()</code>	<code>openssl_x509_parse()</code>	The data of a certificate.
<code>csrNew()</code>	<code>openssl_csr_new()</code>	A certificate signing request.
<code>csrSign()</code>	<code>openssl_csr_sign()</code>	A certificate, signed.
<code>pkcs12Read()</code>	<code>openssl_pkcs12_read()</code>	The certificate, the private key and the chain of a PKCS#12.
<code>pkcs12Export()</code>	<code>openssl_pkcs12_export()</code>	The content of a PKCS#12.
<code>sign()</code>	<code>openssl_sign()</code>	The signature of data (binary).
<code>verify()</code>	<code>openssl_verify()</code>	<code>true</code> if the signature is correct, <code>false</code> if it is not.

A call that fails (`false`, or `-1` when verifying) is a `CertificateException`, and is never a value that is returned: what the methods give is never a failure. A signature that is not valid is a result (`false`), not an error.

Why It Exists

The queue of errors of OpenSSL belongs to the process, not to the call. Another call can leave errors in it, even a call that did not fail: a signature that is not valid is a result, and leaves errors. If an exception took the errors that are in the queue, it would show errors of a call that has nothing to do with it, and with a process that lives for many requests, errors of the request of another user.

So each call is done in its own scope:

1. The queue is emptied **before** the call: what was there is not of this call.
2. The warnings of PHP that OpenSSL emits are taken, and are not emitted.
3. The queue is emptied **after** the call, whether it failed or not: nothing is left behind.

What is in the errors of a `CertificateException` is what its call left. The exception itself does not read the queue: it has the errors that it is given (see [Exceptions](#)).

```
// A signature that is not valid leaves errors in the queue of OpenSSL...
$openSsl->verify('tampered', $signature, $certificate, OPENSSL_ALGO_SHA256); // false

// ...and they are not in the errors of the next exception.
try {
    $openSsl->pkcs12Read('invalid', 'password');
} catch (CertificateException $e) {
    $e->getErrors(); // Only the error of this call: "OpenSSL: error:0680000E:asn1
encoding routines::not enough data"
}
```

The Errors

The errors of OpenSSL are translatable messages (see [Exceptions](#)). Ten are known and have their own message in the catalogue, with the code of the error. Any other error, and the warnings of PHP, are the text as OpenSSL or PHP say it (`OpenSSL: {text}`).

Using It and Replacing It

`Certificate`, `SelfSignedCertificate`, `CertificateLoader` and `CertificateFaker` receive an `OpenSslInterface` as the last argument of their constructor, and it is optional: without it they use `new OpenSsl()`.

```
$loader = new CertificateLoader(); // Uses OpenSsl.
$loader = new CertificateLoader(new MyOpenSsl()); // Uses another
implementation.
```

In an application with the Symfony container, `resources/config/certificate-services.yaml`

registers `OpenSslInterface` with `OpenSsl`.

Another implementation is useful for a test that needs OpenSSL to fail, which with a real one is hard to do:

```
$openSsl = $this->createStub(OpenSslInterface::class);  
$openSsl->method('x509Parse')->willThrowException(new CertificateException('OpenSSL  
could not read the data of the certificate.'));  
  
$certificate = new Certificate($pem, null, [], $openSsl);  
$certificate->getData(); // Throws the exception.
```

Only It Calls OpenSSL

A test of the package fails if a function `openssl_*` is called anywhere in `src/` that is not this service. The code that is not the service does not have the scope of the calls, and an error of OpenSSL that it caused would be left in the queue.

Last updated on 17/10/2026