1 Introduction

This document is a technical introduction to the Signaturgruppen SPS PHP client library.

For the rest of this document the SPS PHP client will be referred to as "the client".

2 Requirements

PHP version 5.6 or greater and basic knowledge of the Composer tool : https://getcomposer.org/.

3 Configuration

The client is configured through a .json snippet which determines the certificates, client-ID and endpoint of the SPS services backend.

In the demo this is already preconfigured for our test setup out-of-the-box.

3.1 Load configuration

The configuration is loaded through

```
using Signaturgruppen\SPS\Configuration\Config;
Config::setDefaultConfigJson(config);
```

The config parameter is the JSON configuration snippet used in the setup. Note that this snippet is preconfig-uret for the testenvironment, and you are not required to change this before changing environment.

3.2 NemIDConfig

The static Signaturgruppen \ SPS \ Configuration \ NemIDConfig class holds settings for configuring the NemID client

Currently the following is supported:

Option	Values	Description
RememberUserId	true / false	Allows user to check "remember me"

3.2.1 NemIDConfig usage

Example of how to set the NemIDConfig settings, see Configuration.php in the demo.

Setting RememberUserId is done like this

```
NemIDConfig::$RememberUserId = true;
```

3.3 Configuration.php

Use the Configuration.php as an example for how the configuration could be loaded and initialized. This enables both static and dynamic loading of the configuration.

4 HTML 5 Web Messaging and iframes

The flow works by setting up an iframe and then communicating with the backend using the HTML 5 Web Messaging JavaScript API through an iframe.

The client library generates the code needed to enable communication with the backend through an iframe. As an example, the following code snippet creates the signed parameters for the backend as well as the JavaScript needed for the communication

```
$nemid = new \Signaturgruppen\SPS\Frame\NemIDFrameBuilder();
echo $nemid->buildScript("receive.php");

<iframe src="<?php echo $nemid->getFrameUrl() ?>" scrolling="no"
style="position:relative; width: 200px; height: 250px; border: none; margin: 0; padding: 0;"></iframe>
```

The argument for the buildScript function, is the postback-url, to which the response from the backend is posted when the client flow is completed or fails inside the iframe.

The ifrane is setup in the wanted size, minimum 200x250px, and using the src as showed in the demo.

The response is handled at "receive.php" by using our client:

```
\Signaturgruppen\SPS\Demo\Configuration::loadConfig();
$postResult = $_POST["result"];

$validator = new \Frame\NemIDFlowValidator();
try{
  $flowResult = $validator->validate($postResult);
  echo "<pre>" . json_encode($flowResult, JSON_PRETTY_PRINT) . "</pre>";
}catch (\Signaturgruppen\SPS\Frame\ValidationException $exeption){
  echo "Error: " .$exeption->errorMessage .
       ", Status: " . $exeption->status .
       ", User message (if set): " . $exeption->userMessage;
}
```

The $flowResult- > AuthenticationInfo object contains the successful NemID authentication info like PID, RID, certificate subject, the signed XMLDSig etc.

The validation function throws an exception if the result is not as expected or if the result does not successfully validate as a valid and expected NemID response. If a result object is returned, the user can be logged in or the document has been signed successfully.

4.1 CPR Services

The client library supports the CPR services available through the NemID infrastructure. If you are a private service provider you are restricted to the CPR Match service, which lets you validate a given CPR paired with the Personal ID (PID) from a successfull NemID authentication flow.

4.1.1 CPR (non-private service providers)

If you have a CPR agreement with DanID it is a simple configuration setting at the backend which enables the CPR service and sets the CPR in the response from the backend and makes it available through

```
FlowResult->AuthenticationInfo->Cpr.
```

The CPR is encrypted and decrypted transparently using RSA-SHA256.

4.1.2 CPR Match (private service providers)

The CPR Match service is available at the backend as a RESTful service available transparently through the client library. A CPR Match agreement with DanID is required.

The CPR service is called using the following:

```
$cprService = new \Signaturgruppen\SPS\Services\PidMatchService();
$cprMatched = $cprService->pidMatchesCpr($flowResult->AuthenticationInfo->Pid, "CPR-FROM-USER");
```

The pidMatchesCpr(string pid, string cpr) method returns true or false. All communication is secured by https and signed and encrypted using RSA-SHA256.

5 Backend response error

In case of errors, a ValidationException is thrown, with the following properties:

```
public string errorMessage;
public string userMessage;
public string flowErrorCode;
public string status;
```

5.0.1 ErrorMessage

Contains an errormessage sutable for your log.

5.0.2 UserMessage

Contains a user friendly message which can be safely displayed as html-text. No dynamic html content will be returned and in case of links, the link will always be a target=" _ blank" link.

If the UserMessage is empty, we recommend a generic error page for your users. We will try to have all errors where meaningful error messages can be displayed for the user covered.

5.0.3 FlowErrorCode

Contains a flow specific errorcode, which when errors occur in the NemID client, contains the NemID client errorcode.

6 Flow overview

To illustrate how the different parts work together, we provide a short overview of the overall flow here.

All communication are done via https and are thus encrypted.

1. The client website opens an iframe to the backend and sends the generated and signed parameters to the backend utilizing the HTML 5 Web Messaging JavaScript API.
2. The backend validates the parameters received.
3. The requested flow/client is started and generated parameters used. The user is interacting directly with the service inside the iframe using a secure connection.
4. As an example: The NemID client is started inside the iframe which transparently lets the user interact with the NemID client, as-if the client was hosted directly on the client website.
5. When the user has entered his credentials the response is validated and all required logs created at the backend.

6. A response is created and sent to the client website through the iframe using the HTML 5 web massaging API.
7. The response is validated using the client library, after which all required information is available through the client library API.
8. All messages are signed and sensitive information is encrypted. Our library transparently validates and de-/encrypts the messages.

7 Missing features

We try to keep all of our clients up-to-date with the features available through the NemID client.

If you miss a feature in the PHP client dont hesitate to contact us.

8 Contact information

Info	
Mail	support@signaturgruppen.dk
Phone	+45 70256425
Website	http://www.signaturgruppen.dk