**Payment Initiation Sandbox API**

At Rabobank, we are opening up payment initiation API for third party payment providers (TPPs). For testing purpose, we also provide a sandbox environment to TPPs. The API definition is provided by the Open API format (see [api-definition.yaml] (./api-definition.yaml). TPPs shall implement their application according to the definition, so the application must be compliant to it.

**Your task**

You need to implement a web application which confirms the API definition. The application doesn't create actual payments but validating the incoming requests from TPPs, check amount limit and respond back the defined response.

When the validation failed, then the application must return HTTP status code `400`. If the amount limit check failed, then HTTP status code `422` will be returned. If the request passes the all validations and amount check, then the application must return HTTP code `201`. The `paymentId` must be generated on the application.

**Validations**

These are the validations need to be in place.

1. White listed certificates validation
2. Signature validation
3. Request validation

**Details of the validations are described below sections.**

**Amount limit exceeded**

In the production environment, TPPs may face an error of `LIMIT\_EXCEEDED` due to the balance of the account or other reasons. We need to emulate this situation for recovery scenarios of TPPs.

A limit exceeded error is returned when the given request is met with the following formula:

```

Amount > 0 && Sum (DebtorAccountIBAN) mod Length (DebtorAccountIBAN) == 0

```

Where Sum is the sum of the digits of the `DebtorAccountNumber`. The alphabets needs to be ignored in the sum calculation.

**Response**

The application must always respond with `Signature` and `Signature-Certificate` HTTP headers. The `Signature` header contains the signature of the response.

**Validation details**

1. **White listed certificates validation**

We only expose this API to selected TPPs at this moment. This is done by white listing incoming certificates. The way we are white listing is checking the common name (CN) of the subject name of incoming certificate. If a CN start with `Sandbox-TPP`, then the certificate is known. Otherwise, the application must respond

`UNKNOWN\_CERTIFICATE` error response code.

1. **Signature validation**

The below pseudo code describes how to compute a signature:

```

SHA256WithRSA (privateKey, headers['X-Request-Id'] + digest(SHA-256, requestBody))

```

The application must verify the signature with the incoming certificate, the `Signature-Certificate` header value, if the verification failed, then it must respond `INVALID\_SIGNATURE` error response code.

You can find an example signature in

[example-signature.md](./example-signature.md) file.

1. **Request validation**

The request body must be validated according to the format defined by in the Open API file. For IBANs, The application doesn't need to check the check digits of the IBAN (it is nice to have, though).

If the IBAN validation is failed, then the application must respond

`INVALID\_REQUEST` error response code.

**GENERAL\_ERROR**

When something went wrong on the application, it must respond HTTP status code `500` and `GENERAL\_ERROR` error response code. This includes an error case which isn't applicable of the above validation error situation (e.g. missing certificate).