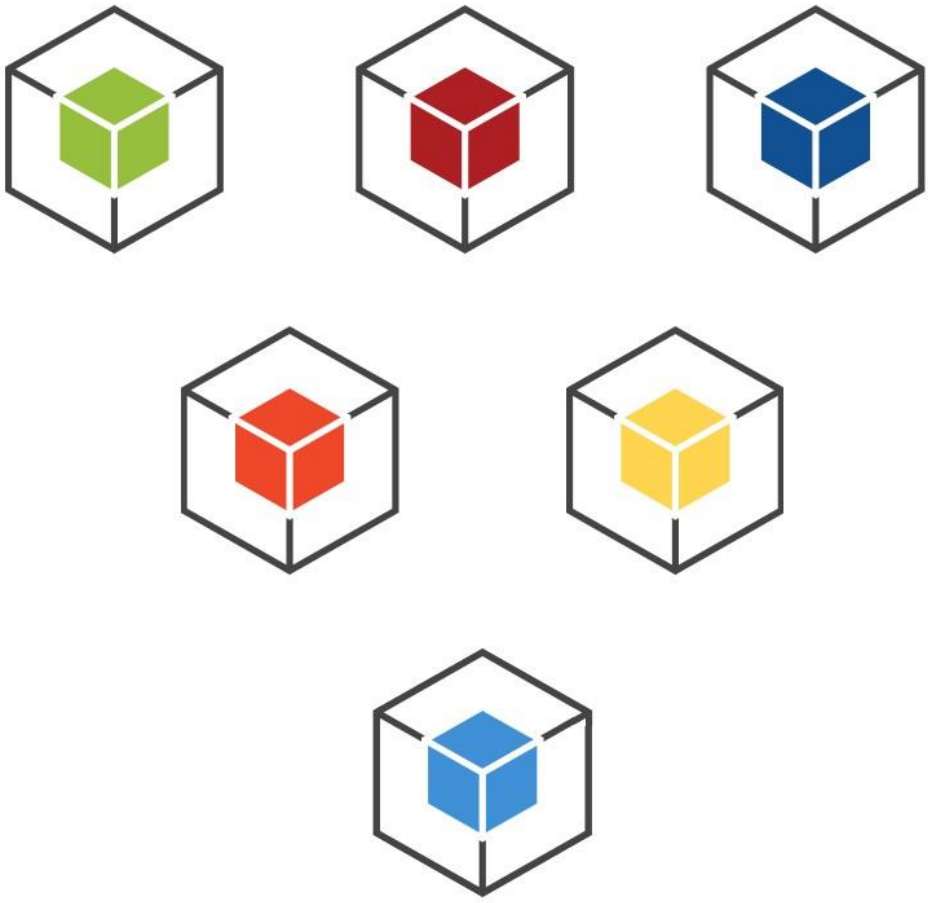
API

PSD2 Payments API

Version 1.3 ● 17 August 2021



BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

Address: 66 Unirii Blvd.K3 Block, 3rd District Bucharest, Romania

Email: psd2.sandbox@banorientfrance.com

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

Trademarks

EUBank are registered trademarks of Advahoo SRL Company. All other trademarks or registered trademarks are the property of their respective owners.

Disclaimer

The information provided in this document is provided "as is" without warranty of any kind. BANORIENT Bank disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall BANORIENT Bank be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if BANORIENT Bank or its suppliers have been advised of the possibility of such damages.

Document Lifetime

BANORIENT Bank may occasionally update online documentation between releases of the related software. Consequently, if this document was not downloaded recently, it may not contain the most up-to-date information. Please refer to https://www.banorientfrance.com for the most current information.

From the Web site, you may also download and refresh this document if it has been updated, as indicated by a change in this date: 14-03-2019.

Where to get help

BANORIENT Bank support, product, and licensing information can be obtained as follows.

Product information

For documentation, release notes, software updates, or for information about BANORIENT Bank products, licensing, and service, go to the following website: https://www.banorientfrance.com/romanian/romania **Technical support**

For technical support, use the email address psd2.sandbox@banorientfrance.com.

Note that to open a service request, you must have a valid support agreement.

Your comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications. Please send your opinion of this document to: psd2.sandbox@banorientfrance.com

If you have issues, comments, or questions about specific information or procedures, please include the title and, if available, the part number, the revision, the page numbers, and any other details that will help us locate the subject that you are addressing.

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

Preface

Intended Audience

This guide is part of the PSD2 Payments API documentation set. It is intended for use by System Administrator, Application Developers from Third Party Provider during integration of the PSD2 services offered by BANORIENT Bank.

Readers should be familiar with the following API specifications defined by Berlin Group.

01. NextGenPSD2 Access to Account Interoperability Framework - General Introduction Paper V2_20181120.pdf

02. NextGenPSD2 Access to Account Interoperability Framework - Operational Rules V1_20180208.pdf

03. NextGenPSD2 Access to Account Interoperability Framework - Implementation Guidelines V1.3_20181019.pdf

04. NextGenPSD2 Access to Account Interoperability Framework - ChangeLog V12 V13 20181019.pdf

Style Conventions

The following style conventions are used in this document:

Bold

- Names of commands, options, programs, processes, services, and utilities
- Names of interface elements (such windows, dialog boxes, buttons, fields, and menus)
- Interface elements the user selects, clicks, presses, or types

Italic

- Publication titles referenced in text
- Emphasis (for example a new term)
- Variables

Courier

- System output, such as an error message or script
- URLs, complete paths, filenames, prompts, and syntax

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

Courier italic

- Variables on command line

User input variables

- < > Angle brackets enclose parameter or variable values supplied by the user
- [ ] Square brackets enclose optional values
- | Vertical bar indicates alternate selections - the bar means “or”
- { } Braces indicate content that you must specify (that is, x or y or z)

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

Table of Contents

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

Document History

Paper copies are valid only on the day they are printed. Contact the author if you are in any doubt about the accuracy of this document.

Revision History

This document has been revised by:

Revision Number	Revision Date	Summary of Changes	Author
v1.1	26 March 2020	Initial version	BANORIENT Bank
v1.2	14 July 2020	Second version	BANORIENT Bank
v1.3	17 August 2021	Third version	BANORIENT Bank

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

1. Overview

This guide presents the PSD2 Payments API services offered by BANORIENT Bank.

The services are protected by OAuth2 protocol. Order of presentation for the services will follow the logic access, including authentication step, token exchanges, and status.

This will help users to standalone test the services without developing a specific application for this purpose.

All services are documented using Open API 3.0 version.

For details please follow https://sandbox.banorientfrance.com/eubank/openapi-payments

1.1. Conventions

We use the following conventions in this document:

- Responses are listed under 'Responses' for each method.
- Responses are in JSON format.
- Request parameters are mandatory unless explicitly marked as Optional.
- The type of values accepted for a request parameter are shown the **values** column.
- The | symbol means OR.

1.2. Current Version

Version specification follow Berlin Group recommendations and are present in the URL form [/v1/]

1.3. Schema

All API access is over HTTPS, and accessed from the address: https://sandbox.banorientfrance.com/DVHPSD2PaymentsAPI/

All data is sent and received as JSON.

All timestamps are returned using the ISO 8601 format: YYYY-MM-DDTHH:MM:SS

Summary Representations - When you fetch a list of resources, the response includes a subset of the attributes for that resource. This is the "summary" representation of the resource.

Detailed Representations - When you fetch an individual resource, the response typically includes all attributes for that resource. This is the "detailed" representation of the resource.

1.4. HTTP Redirects

Redirection are used by OAuth2 protocol in order to deliver access code to TPP.

Receiving an HTTP redirection is not an error and clients should follow that redirect. Redirect responses will have a Location header field which contains the URI of the resource to which the client should repeat the requests.

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

Status Code	Description
302	Temporary redirection. The request should be repeated verbatim to the URI specified in the Location header field but clients should continue to use the original URI for future requests.

1.5. Communication security

PSD2 Directive defines requirements on communication among payment service providers and account servicing institutions.

The Regulatory Technical Standards defines requirements on the use of qualified certificates (as defined in eIDAS) for website authentication and qualified certificates for electronic seal for communication among payment and bank account information institutions.

The ETSI TS 119 495 defines a standard for implementing the requirements of the RTS for use of qualified certificates as defined in eIDAS (Regulation (EU) No 910/2014) to meet the regulatory requirements of PSD2.

Regulatory Technical Standards mandates the use of certificates according to Article 34. The article restricts the use of certificates to "qualified certificates for electronic seals as referred to in Article 3(30) of Regulation (EU) No 910/2014 or for website authentication as referred to in Article 3(39) of that Regulation".

EUBank will encrypt the communication between Bank and TPP by using a SSL extended validation certificate. No mutual TLS authentication and encryption will be used.

The TPP request and Bank responses will be authenticated and protected by the usage of QSealC certificates. Both TPP and Banks will sign the corresponding requests and responses using qualified seal certificates.

Message Signing

Each request initiated by a TPP must contain a JSON Web Signature as a header. This header signs the payload of the request, using the private key of the TPP's compliant certificate. The responses are also signed using the Bank's certificate, using the same technique. Both the Bank and the TPP must validate requests and responses using the appropriate public keys.

The present documentation details the signing procedure for the TPP requests. The bank will sign responses using the same methodology.

We assume TPP software will compute and attach the header signature for each API requests.

For procedure verification only the document include detailed examples for building signatures using only Windows command line.

References

In order to build and check the signature on requests please considers the following references:

JSON Web Signature Documentation

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

Base64URL Documentation

OpenSSL

Certificates

The sandbox contains for testing a pre-registered TPP SC_EXEMPLU_SRL. The certificate and the private key for SC_EXEMPLU_SRL are available for download on sandbox page.

The Bank's public certificate is available for download on Bank website and sandbox page.

OAuth2 Protocol Implementation

The initiation of transactions is controlled using Authorization code grant type protocol. For SC_EXEMPLU_SRL TPP an application was previously registered with following parameters:

Client Id: LrcL4ywuHuLtyf34g40LNf14RFfDJ4SL

Client Secret: N9Vt3Jm9Bx3MCDByyclwXcbliyqxXzGk

Request header parameter signature format

Each request coming from the TPP will include a special header parameter x-jws-signature.

The signature includes three sections:

1. JWS Header
2. JWS Payload
3. JWS Signature

The three sections are finally assembled in the parameter x-jws-signature.

x-jws-signature= Base64URL (JWS Header)‘.'Base64URL (JWS Payload)‘.'Base64URL (JWS Signature)

1. The JWS Header

The JWS Header will contain specific information:

- alg: the algorithm to sign - RS256
- typ: type of the encoded object –JOSE
- kid: certificate thumbprint for SC_EXEMPLU_SRL the value is:

  133c11470740d7ed33c86c3501e3ac8221fece03

Consequently, the JWS Header will be:

```
JWS Header = {"alg":"RS256","typ":"JOSE","kid":"133c11470740d7ed33c86c3501e3ac8221fece03"}
```

For obtaining Base64URL(JWSHeader) the steps are:

- Compute base64 for JWS Header

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

- Replace any occurrences of '+' character with '-' and any occurrences of '/' character with '_'. Also, delete every '=' from the resulted string.

The result in our test case is:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZjMzUwMWUzYWM4MjIxZmVjZTAzIn0
```

2. *The JWS Payload*

The JWS Payload is specific for each request; an example is provided for each request type within the document, starting from the general template of the JSON request which includes the headers information and the body information, altogether on a single line, trim spaces:

```
{{"headers":{all not null headers properties as they occur in request },"payload":{request body}}
```

3. *The JWS Signature*

The process of computing the *JWS Signature* component includes the following steps:

1. Concatenate the first two parts separated by a *'.'*:

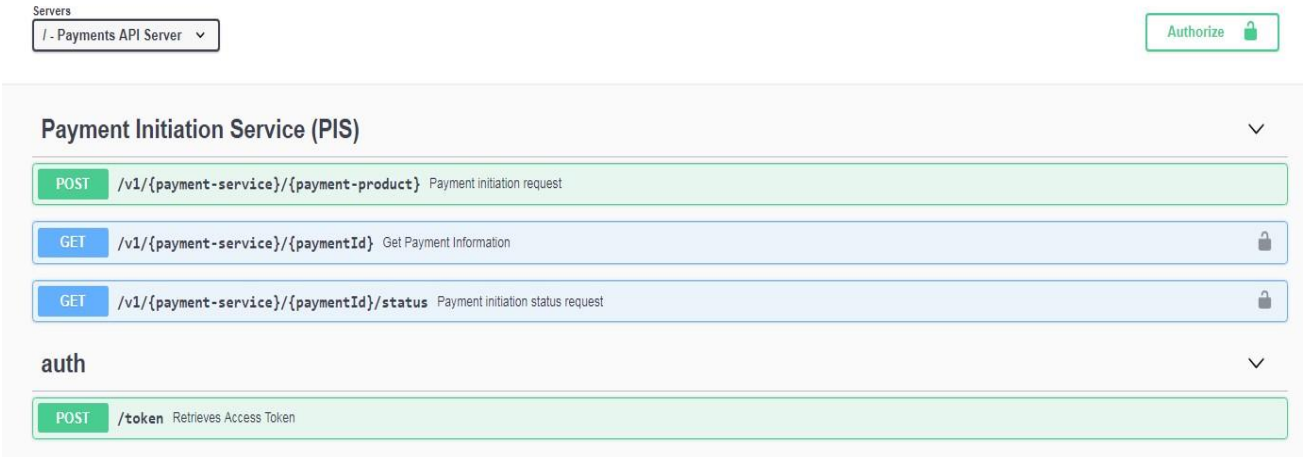
```
Base64URL(JWS Header) '.' Base64URL(JWS Payload)
```

2. Sign the resulted string using the TPP private key and then apply Base64 encoding.
3. Replace any occurence of “+" character with “-" and any occurence of "/" character with “_". Also, delete every “=" from the resulted string.

2. API Reference Documentation

PSD2 Payments API

Under the Payment Initiation Service, the following set of methods is available (Figure1):



© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

Figure 1

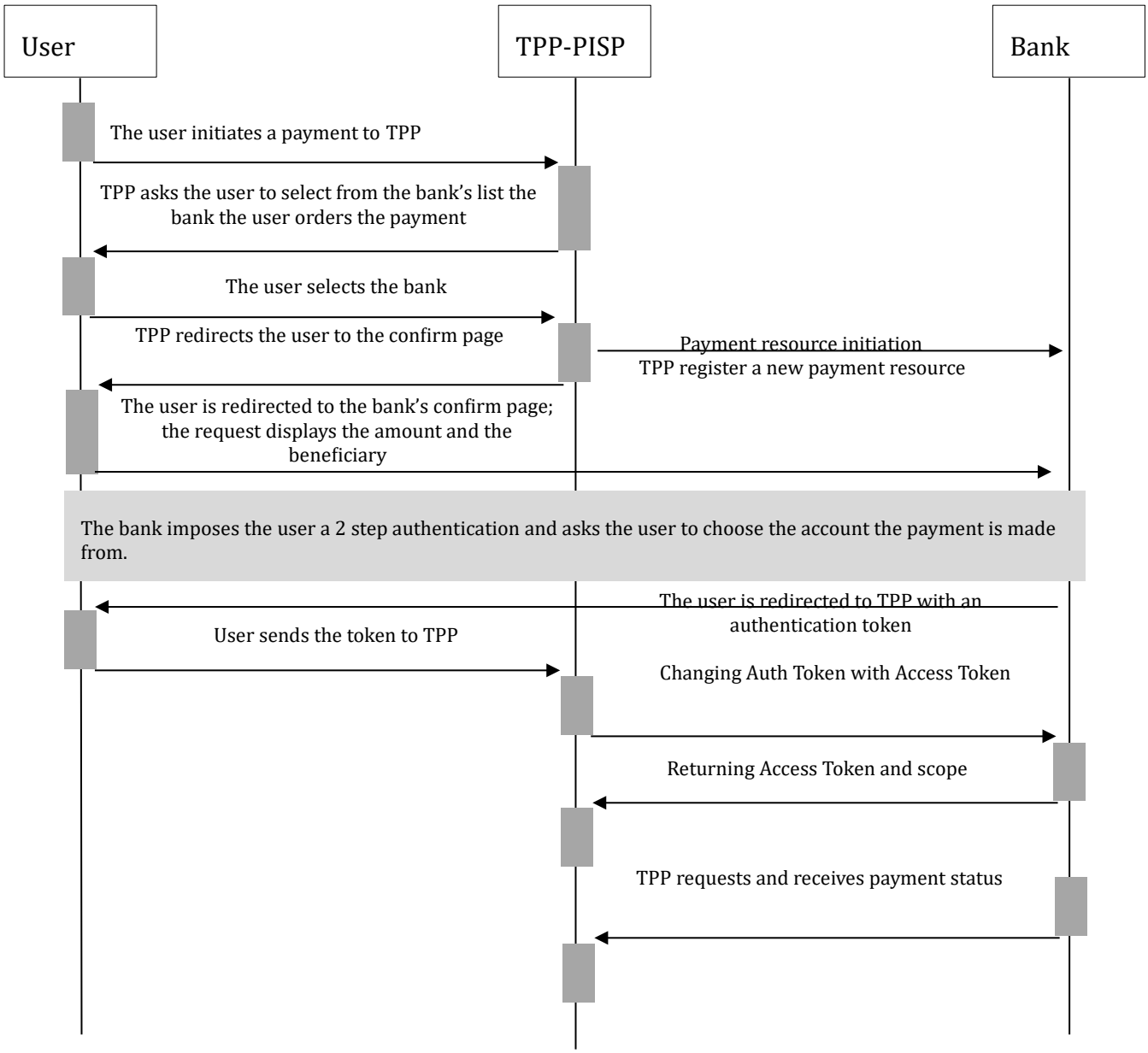
2.1. Overview

Method	Purpose
/v1/{payment-service}/{payment-product}	Payment initiation request
/v1/{payment-service}/{paymentId}	Get payment information
/v1/{payment-service}/{paymentId}/status	Check the status of a payment resource
/token	Retrieve access token for a specific payment resource

2.2. Specific flow

Under PSD2 rules making a payment follow a specific flow of API calls. The calls are protected by OAuth2 authentication and authorization protocol using authorization code flow.

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA



3. Testing a payment flow example

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

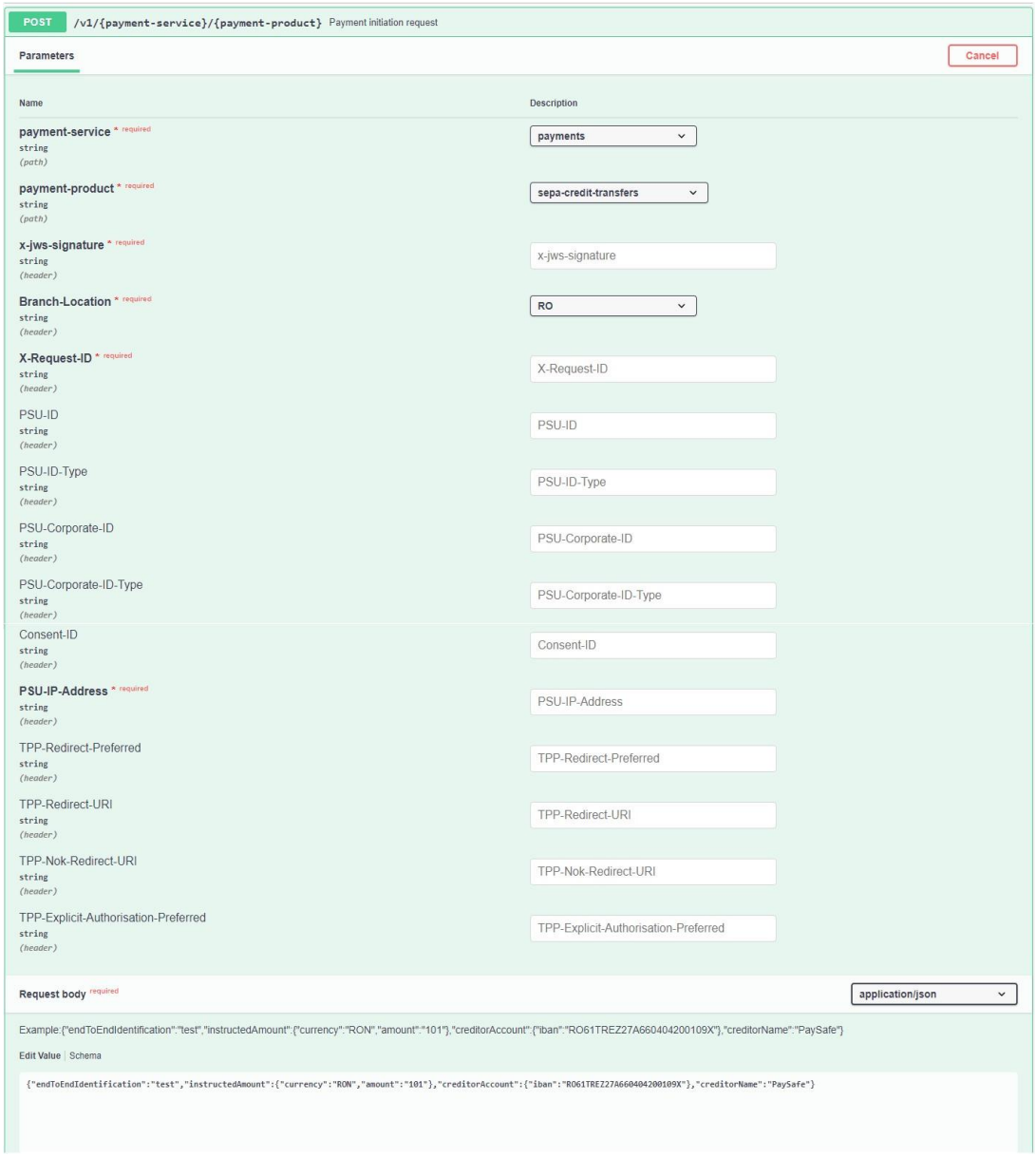


Figure 2

This section presents an example of complete payment registration. The scenario assumes one customer [PSU] initiated a payment on an e-Commerce site [TPP]. In order to make a payment using PSD2 standard, the user is prompted to choose the bank where he owns a checking account. After the bank account selection, the e-Commerce site initiates a payment resource registration. This action is done through the API call: Payment resource initiation on bank side [ASPSP].

For testing this service please launch the call from the BANORIENT Bank sandbox UI (Figure 2).

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

The example provided assumes that the user should pay 101 RON to the e-Commerce site. The request body of the API call contains information related to the amount, currency, creditor name [e-Commerce merchant] and his account.

Steps

The *Request Body must be* on a single line without spaces:

```
{"endToEndIdentification":"test","instructedAmount":{"currency":"RON","amount":"101"},"c
reditorAccount":{"iban":"RO61TREZ27A660404200109X"},"creditorName":"PaySafe"}
```

Compute the ***Base64URL (JWS Payload):***

1. Create the following JSON on a single line without spaces:

```
{"headers":{"Branch-Location":"RO","X-Request-ID":"b7d96357-b320-4f54-
bb021c4511e4b772","PSU-IP-
Address":"127.0.0.1"},"payload":{"endToEndIdentification":"test","instructedAmount":{"curr
ency":"RON","amount":"101"},"creditorAccount":{"iban":"RO61TREZ27A660404200109X"},"c
reditorName":"PaySafe"}}
```

2. Apply SHA-256 on the JSON from step 1 using the following command (on Windows OS):

```
echo|set /p="{"headers":{"Branch-Location":"RO","X-Request-ID":"b7d96357-b320-4f54bb02-
1c4511e4b772","PSU-IP-
Address":"127.0.0.1"},"payload":{"endToEndIdentification":"test","instructedAmount":{"curr
ency":"RON","amount":"101"},"creditorAccount":{"iban":"RO61TREZ27A660404200109X"},"c
reditorName":"PaySafe"}}" | openssl dgst -sha256
```

   The result will be:

```
986b4b3fb73b0d96e777799fcf439d675850c73021c3482f95da1a2b3cea75ab
```

3. Create the following JSON with the result:

```
{"SHA256":"986b4b3fb73b0d96e777799fcf439d675850c73021c3482f95da1a2b3cea75ab"}
```

4. Compute Base64 encoding on the later JSON using the following command (on Windows OS):

```
echo | set
/p="{"SHA256":"986b4b3fb73b0d96e777799fcf439d675850c73021c3482f95da1a2b3cea75a
b"}" | openssl base64 -e -A
```

5. Replace any occurence of “+" character with “-" and any occurence of "/" character with “_". Also, delete every “=" from the resulted string. The result for our test case is:

```
eyJTSEEyNTYiOiI5ODZiNGIzZmI3M2IwZDk2ZTc3Nzc5OWZjZjQzOWQ2NzU4NTBjNzMwMjFjMzQ
4MmY5NWRhMWEyYjNjZWE3NWFiIn0
```

Compute the ***JWS-signature:***

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

1. Compute the following string: *Base64URL(JWS Header) '.' Base64URL(JWS Payload)*

   The result will be:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZjMzUwMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiI5ODZiNGIzZmI3M2IwZDk2ZTc3Nzc5OWZjZjQzOWQ2NzU4NTBjNzMwMjFjMzQ4MmY5NWRhMWEyYjNjZWE3NWFiIn0
```

2. Sign the string using the TPP private key and apply Base64 encoding using the following command (on Windows OS):

```
echo | set /p="eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZjMzUwMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiI5ODZiNGIzZmI3M2IwZDk2ZTc3Nzc5OWZjZjQzOWQ2NzU4NTBjNzMwMjFjMzQ4MmY5NWRhMWEyYjNjZWE3NWFiIn0"| openssl dgst -sha256 -sign SC_EXEMPLU_SRL.key | openssl base64 -e –A
```

   The result will be:

```
g4yWSERVKtrmPT9cv-clycjr7ZQqFbQWLZ2rrqJM4jPELT7Nyf2syFDyAxgxjp8IVhhJvKsdLclkBaCs3YHB-irFGR2xJ0hj1maU4P1DbCL_kraEdY_5MGY-B35GgtVaER61t170kkMUEbIUVm_C2meiE3FXpIfNVh7kKYWOcGQOJ353cV3IzuFBZXle8b5pFf8rwMcrMJfgycfFIdiUCW1e6xMgwNWdbqgW0VdiniXNJl3SWfi1JY8ly4RI4p71tWDHqaknZkij__otSYv3vFORkeWPmvi_OfJluRcGtjbVKJLQosPS9sxuL8-MNZXJUKWEaZz0NLv07Nzk-g
```

3. Finally, the Header JWS-signature will be:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZjMzUwMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiI5ODZiNGIzZmI3M2IwZDk2ZTc3Nzc5OWZjZjQzOWQ2NzU4NTBjNzMwMjFjMzQ4MmY5NWRhMWEyYjNjZWE3NWFiIn0.g4yWSERVKtrmPT9cv-clycjr7ZQqFbQWLZ2rrqJM4jPELT7Nyf2syFDyAxgxjp8IVhhJvKsdLclkBaCs3YHBirFGR2xJ0hj1maU4P1DbCL_kraEdY_5MGY-B35GgtVaER61t170kkMUEbIUVm_C2meiE3FXpIfNVh7kKYWOcGQOJ353cV3IzuFBZXle8b5pFf8rwMcrMJfgycfFIdiUCW1e6xMgwNWdbqgW0VdiniXNJl3SWfi1JY8ly4RI4p71tWDHqaknZkij__otSYv3vFORkeWPmvi_OfJluRcGtjbVKJLQosPS9sxuL8-MNZXJUKWEaZz0NLv07Nzk-g
```

The bank successful response to the e-Commerce site returns the paymentId and the link for authentication and authorization of the customer.

```
{
 "transactionStatus": "RCVD",
 "paymentId": "c806a400-4671-4ee9-b712-d45262df6d1b",
 "_links": {
  "scaOAuth": "https://sandbox.banorientfrance.com/services/startAuthorize"
 }
```


© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

```
}
```

Request Response

Following the successful response, the e-Commerce site redirects the user to the SCA authentication and authorization page according to OAuth2 authorization code flow.

In order to test this step from sandbox, please press Authorize button. From the dialog box choose OAuth2 authorizationCode method, fill in the clientId field and choose the payment scope [make payment].

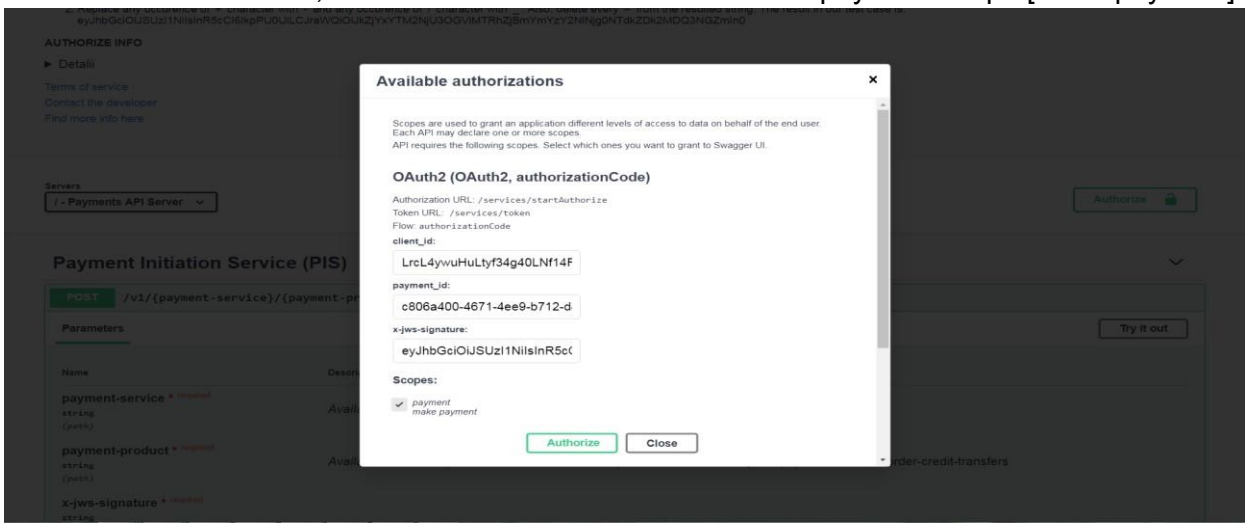


Figure 3

Steps

Compute the ***Base64URL (JWS Payload):***

1. Create the following JSON on a single line without spaces:

```
{"headers":{},"payload":{"payment_id":"c806a400-4671-4ee9-b712-d45262df6d1b","scope":"payment","response_type":"code","state":"state","client_id":"LrcL4ywuHuLtyf34g40LNf14RFfDJ4SL"}}
```

2. Apply SHA-256 on the JSON from step 1 using the Windows command:

```
echo|set /p="{"headers":{},"payload":{"payment_id":"c806a400-4671-4ee9-b712d45262df6d1b","scope":"payment","response_type":"code","state":"state","client_id":"LrcL 4ywuHuLtyf34g40LNf14RFfDJ4SL"}}" | openssl dgst -sha256
```

The result will be:

```
f1b44d6c1cbecd2afff378cb31572783207c9fd2e97a5ecae952a95a922c669b
```

3. Create the following JSON with the result:

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

```
{"SHA256":"f1b44d6c1cbecd2afff378cb31572783207c9fd2e97a5ecae952a95a922c669b"}
```

4. Compute Base64 encoding on the last JSON using the following command (on Windows OS):

```
echo|set
/p="{"SHA256":"f694764a36ecfeeef63df0c8accae60266edfd376c80d0d5dcd20b4717f94ae3"
}" | openssl base64 -e –A
```

5. Replace any occurence of “+" character with “-" and any occurence of "/" character with “_". Also, delete every “=" from the resulted string. The result for our test case is:

```
eyJTSEEyNTYiOiJmMWI0NGQ2YzFjYmVjZDJhZmZmMzc4Y2IzMTU3Mjc4MzIwN2M5ZmQyZTk3
YTVlY2FlOTUyYTk1YTkyMmM2NjliIn0
```

Compute the ***JWS-signature:***

1. Compute the following string: *Base64URL(JWS Header) '.' Base64URL(JWS Payload)*

   The result will be:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZjMz
UwMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiJmMWI0NGQ2YzFjYmVjZDJhZmZmMzc4Y2
IzMTU3Mjc4MzIwN2M5ZmQyZTk3YTVlY2FlOTUyYTk1YTkyMmM2NjliIn0
```

2. Sign the string using the TPP private key and apply Base64 encoding using the following command:

```
echo | set
/p="eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZj
MzUwMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiJmMWI0NGQ2YzFjYmVjZDJhZmZmMzc4Y
2IzMTU3Mjc4MzIwN2M5ZmQyZTk3YTVlY2FlOTUyYTk1YTkyMmM2NjliIn0"| openssl dgst
sha256 -sign SC_EXEMPLU_SRL.key | openssl base64 -e –A
```

   The result will be:

```
HNZgM29AvGZP72uyTe3f29Nt1Qn7nnuohl3WvVoqq8fmpQQVElNBawhrQ1ahfxo3VQC1BWu
NyDSw1Ba7kcy4fQsJ6cfRZuOJMyHOv0TPfEpiZwXRkZQ--JROrW-
vnYjWIi8_oHcYN1EkYizSPXlmcZPSSW6hN-
WxkUhFIUhgtnAWF9PTOfVGtoiJ0ID1Ag9ImOZPFW5IkIXzElBXkmu3nmacBvxD2VuiUPbqwz6RZO
UkCsrAf96nkhsv9YP4bXXmPeThhLef6hf4dkk9_2Rj
5D_aMwIvX9Q0YsnsP4DZL8318CetOQ-KiPw64-qhxprSkT4oI8_LY0dVUIfooaNg
```

3. Finally, the Header JWS-signature will be:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZjMz
UwMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiJmMWI0NGQ2YzFjYmVjZDJhZmZmMzc4Y2I
```


© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

zMTU3Mjc4MzIwN2M5ZmQyZTk3YTVlY2FlOTUyYTk1YTkyMmM2NjliIn0.HNZgM29AvGZP72uyTe3f29Nt1Qn7nnuohl3WvVoqq8fmpQQVElNBawhrQ1ahfxo3VQC1BWuNyDSw1Ba7kcy4fQsJ6cfRZuOJMyHOv0TPfEpiZwXRkZQ--JROrW-vnYjWIi8_oHcYN1EkYizSPXlmcZPSSW6hNWxkUhFIUhgtnAWF9PTOfVGtoiJ0-ID1Ag9ImOZPFW5I-kIXzElBXkmu3nmacBvxD2VuiUPbqwz6RZOUkCsrAf96nkhsv9YP4bXXmPeThhLef6hf4dkk9_2Rj5D_aMwIvX9Q0YsnsP4DZL8318CetOQ-KiPw64-qhxprSkT4oI8_LY0dVUIfooaNg

Based on the SCA [strong customer authentication] implementation, the user will be redirected to the *Sign in* page, where the value for username is “user1” and the value for password is “Parola1234” (Figure 4).

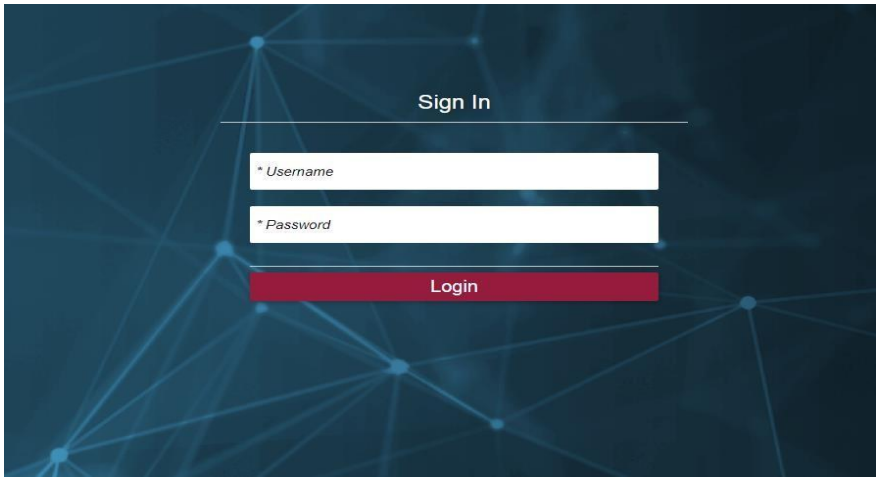


Figure 4

Finally, the user is asked to input the OTP – in this example a SMS code. Please use *123456* to test this scenario (Figure 5).

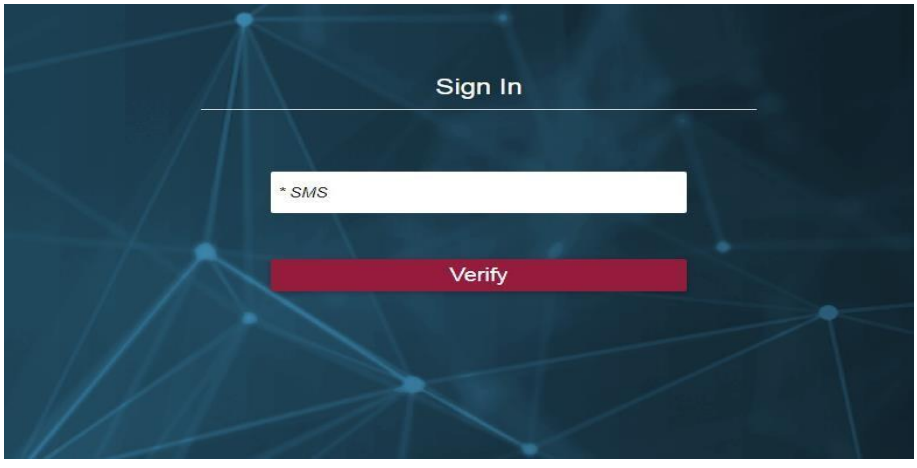


Figure 5

Following a successful SCA for payment intent, the EUBank Auth/Authz Server will present to the user the details of payment, the list of the payment accounts and the option to confirm the intent (Figure 6).

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

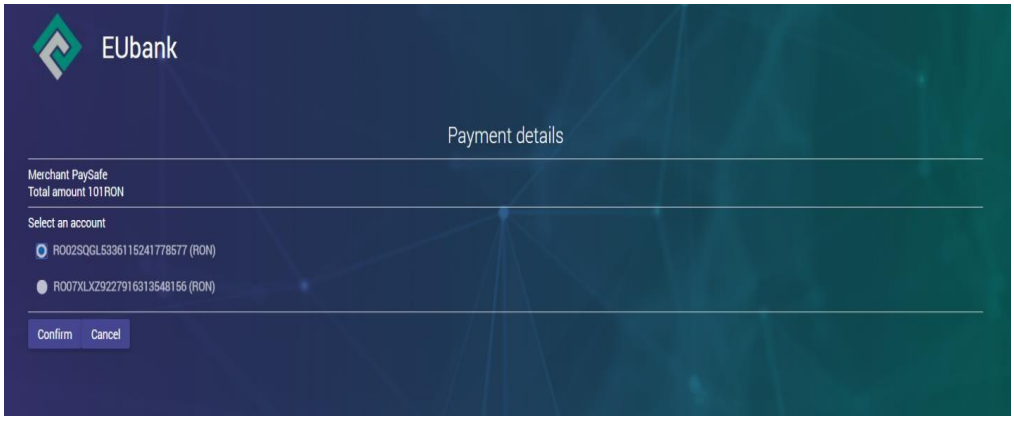


Figure 6

After selecting an account from the list, a dialog will be displayed where a SMS will be sent. For testing purpose, use the value “123456” (Figure 7).

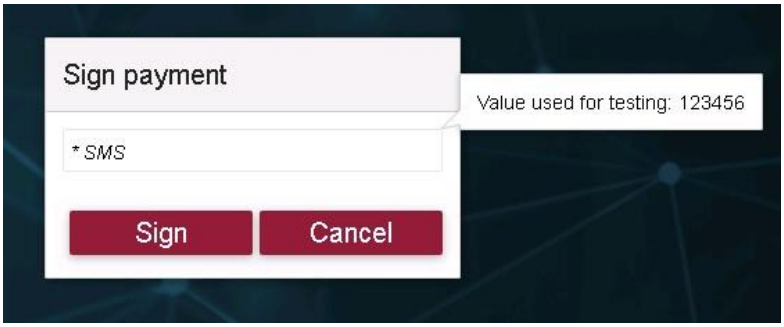


Figure 7

Following a user confirmation, the Auth/Authz Server will update the status of the payment operation on the bank’s side and the user will be redirected to the e-Commerce site with the appropriate authorization code (Figure 8).

In our testing scenario the e-Commerce site is not present. That’s why the redirected action will not succeed, but in our case will allow us to copy the authorization code from the browser URL.

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

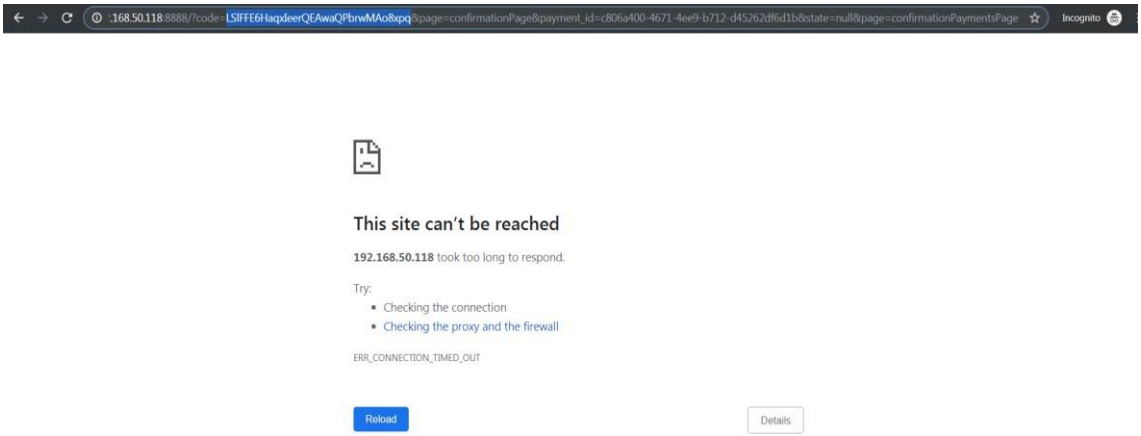


Figure 8

The next step in the OAuth2 flow is to exchange the *authorization code* for the *access token*. For this operation the e-Commerce site application will call Retrieve Access Token specific API on the bank's side. In order to test the Retrieve Access Token service from sandbox first step is to select it (Figure 9).

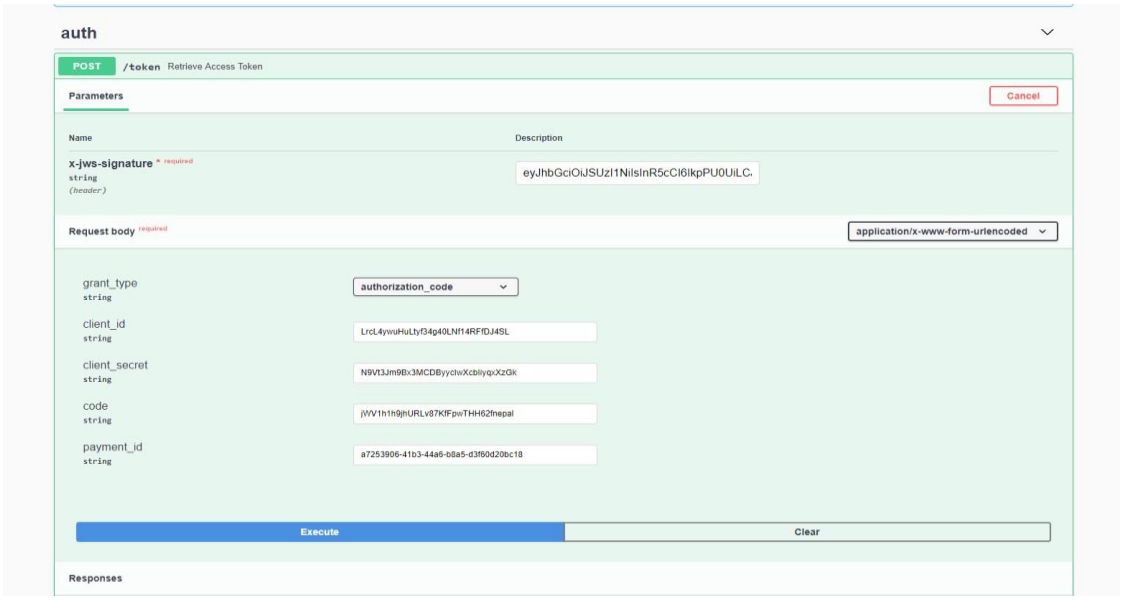


Figure 9

It is necessary to fill in the client Id and client secret (see the constant values in 1.5 Message Signing - Oauth2), authorization code (retrieved from the previous step) and the payment Id (retrieved from the response body of Payment Initiation Request service).

Steps

Compute the ***Base64URL (JWS Payload):***

1. Create the following JSON on a single line without spaces:

```
{"headers":{},"payload":{"grant_type":"authorization_code","client_id":"LrcL4ywuHuLtyf3
```


© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

```
4g40LNf14RFfDJ4SL","client_secret":"N9Vt3Jm9Bx3MCDByycIwXcbIiyqxXzGk","code":"jW
V1h1h9jhURLv87KfFpwTHH62fnepal","payment_id":"a7253906-41b3-44a6-b8a5d3f60d20bc18"}}
```

2. Apply SHA-256 on the JSON from step 1 using the following command (on Windows OS):

```
echo|set
/p="{"headers":{},"payload":{"grant_type":"authorization_code","client_id":"LrcL4ywuHuLtyf
34g40LNf14RFfDJ4SL","client_secret":"N9Vt3Jm9Bx3MCDByycIwXcbIiyqxXzGk","code":"jWV1
h1h9jhURLv87KfFpwTHH62fnepal","payment_id":"a7253906-41b3-44a6-
b8a5d3f60d20bc18"}}" | openssl dgst -sha256
```

The result will be:

```
832e0123e0094042d449b2938fbf90fcac8d048960452163c79d7bfdd003bb8a
```

3. Create the following JSON with the result:

```
{"SHA256":" 832e0123e0094042d449b2938fbf90fcac8d048960452163c79d7bfdd003bb8a"}
```

4. Compute Base64 encoding on the last JSON using the following command (on Windows OS):

```
echo|set
/p="{"SHA256":"832e0123e0094042d449b2938fbf90fcac8d048960452163c79d7bfdd003bb8
a"}" | openssl base64 -e –A
```

5. Replace any occurence of “+" character with “-" and any occurence of "/" character with “_". Also, delete every “=" from the resulted string. The result in our test case is:

```
eyJTSEEyNTYiOiI4MzJlMDEyM2UwMDk0MDQyZDQ0OWIyOTM4ZmJmOTBmY2FjOGQwNDg5
NjA0NTIxNjNjNzlkN2JmZGQwMDNiYjhhIn0
```

Compute the ***JWS-signature:***

1. Compute the following string: *Base64URL(JWS Header) '.' Base64URL(JWS Payload)*

The result will be:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZjMz
UwMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiI4MzJlMDEyM2UwMDk0MDQyZDQ0OWIy
OTM4ZmJmOTBmY2FjOGQwNDg5NjA0NTIxNjNjNzlkN2JmZGQwMDNiYjhhIn0
```

2. Sign the string using the TPP private key and apply Base64 encoding using the following command:

```
echo|set
/p="eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjOD
```


© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

```
ZjMzUwMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiI4MzJlMDEyM2UwMDk0MDQyZDQ0
OWIyOTM4ZmJmOTBmY2FjOGQwNDg5NjA0NTIxNjNjNzlkN2JmZGQwMDNiYjhhIn0" | openssl
dgst -sha256 -sign SC_EXEMPLU_SRL.key | openssl base64 -e -A
```

The result will be:

```
Ke6EaOqYm-
Phx0wU47uNARA9iOpzzGge_nHk0QUF_Csvdlbq0Pvz8_HKS1KBNkAEYnd_AThLKR5sVASUQ60
nTHuYSk7Lo0M8bKBChic4yBqZP2yqURuMbKeH_Vdjk1CkQIK6wpMFLj6TabVVq8QV3GUY3oN
FV5K-
kOQnOyxyJgZCsUD6akmCVQy5RykQB8AsmZ8ujtozDqEyxB37H7RD5WRWezvBn6qYd7PQG1nxa
2MLiBXfVOwx0jTwDIJHNjUhq8O2fDV37d25FS 7anakD-
qEQbNRCFIHs2A0fM0qkWgHqdTSmvbxjxUvimiHgdc6OKr-9JFYmXp1MON38hLcIg
```

3. Finally, the Header JWS-signature will be:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZjMzU
wMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiI4MzJlMDEyM2UwMDk0MDQyZDQ0OWIyOT
M4ZmJmOTBmY2FjOGQwNDg5NjA0NTIxNjNjNzlkN2JmZGQwMDNiYjhhIn0.Ke6EaOqYmPhx0w
U47uNARA9iOpzzGge_nHk0QUF_Csvdlbq0Pvz8_HKS1KBNkAEYnd_AThLKR5sVASUQ60n
THuYSk7Lo0M8bKBChic4yBqZP2yqURuMbKeH_Vdjk1CkQIK6wpMFLj6TabVVq8QV3GUY3oNFV
5K-
kOQnOyxyJgZCsUD6akmCVQy5RykQB8AsmZ8ujtozDqEyxB37H7RD5WRWezvBn6qYd7PQG1nxa
2MLiBXfVOwx0jTwDIJHNjUhq8O2fDV37d25FS7 anakD-
qEQbNRCFIHs2A0fM0qkWgHqdTSmvbxjxUvimiHgdc6OKr-9JFYmXp1MON38hLcIg
```

```
{
 "refresh_token": "PcHoHIcfb5ytbhV2OGZK5TPZGNrLpJAP",
 "token_type": "bearer",
 "access_token": "8714WYCTnTLDxyZtyRWNy7FaAhART4zH",
 "expires_in": 7776000
}
```

Request Response

Upon successful exchange of the authorization code for the access token, the e-Commerce site will be able to call the API for checking the status of the payment or verify the payment instruction.

The e-Commerce application should build the next API request with the presence of the access token in the header of HTTP request.

For testing with the sandbox it is necessary to use the bearer Auth token service (Figure 10). From the available authorization list we choose the bearer Auth service which will have the value of the access_token from the request response (8714WYCTnTLDxyZtyRWNy7FaAhART4Zh).

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

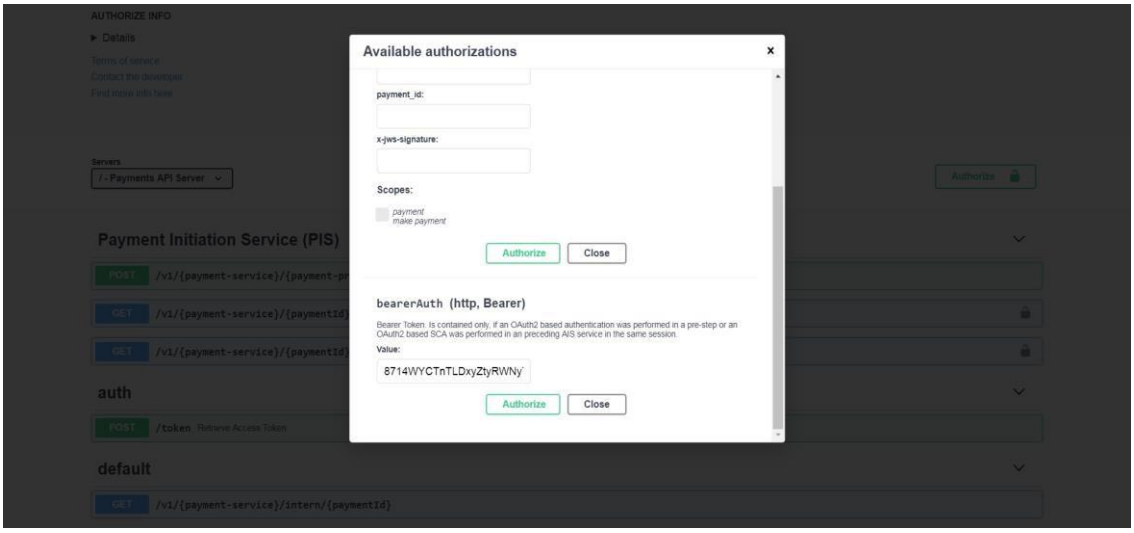


Figure 10

Fill the value with the access token value received in the previous call. This operation will ensure the presence of access token in the HTTP header of the subsequent API requests (Figure 11).

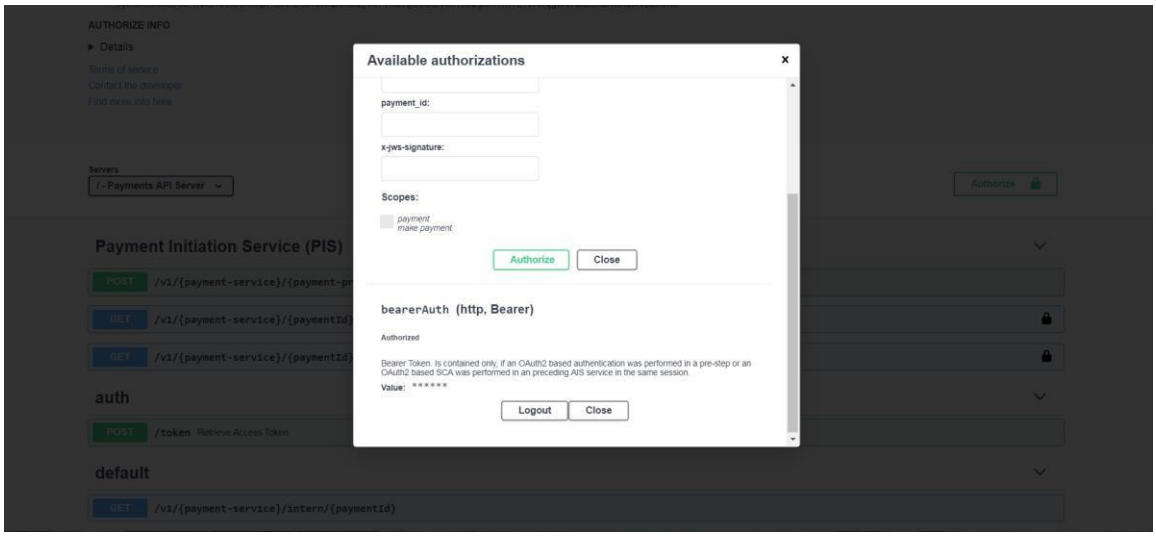


Figure 11

Checking the status of payment

Service 1 (Payment initiation status request)

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

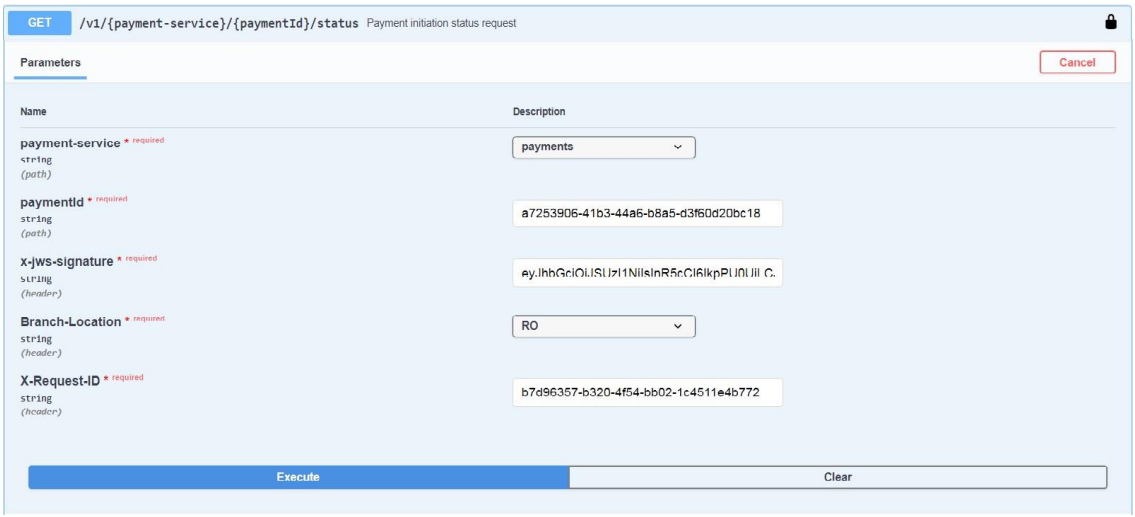


Figure 12

Steps

Compute the ***Base64URL (JWS Payload):***

1. Create the following JSON on a single line without spaces:

```
{"headers":{"Branch-Location":"RO","X-Request-ID":"b7d96357-b320-4f54-
bb021c4511e4b772"},"payload":{"payment_id":" a7253906-41b3-44a6-b8a5-d3f60d20bc18"}}
```

2. Apply SHA-256 on the JSON from step 1 using the following command (on Windows OS):

```
echo|set  /p="{"headers":{"Branch-Location":"RO","X-Request-ID":"b7d96357-b320-4f54-
bb021c4511e4b772"},"payload":{"payment_id":"a7253906-41b3-44a6-b8a5-d3f60d20bc18"}}"
   | openssl dgst -sha256
```

   The result will be:

```
1c8e9e202a4579b7056115d11b083e820ee0b9a90006f2bd61f1ccefb556b5eb
```

3. Create the following JSON with the result:

```
{"SHA256":"1c8e9e202a4579b7056115d11b083e820ee0b9a90006f2bd61f1ccefb556b5eb"}
```

4. Compute Base64 encoding on the later JSON using the following command (on Windows OS):

```
echo|set
/p="{"SHA256":"1c8e9e202a4579b7056115d11b083e820ee0b9a90006f2bd61f1ccefb556b5eb"}"
| openssl base64 -e –A
```

5. Replace any occurrence of “+" character with “-" and any occurrence of "/" character with “_". Also, delete every “=" from the resulted string. The result in our test case is:

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

```
eyJTSEEyNTYiOiIxYzhlOWUyMDJhNDU3OWI3MDU2MTE1ZDExYjA4M2U4MjBlZTBiOWE5MDAwN
mYyYmQ2MWYxY2NlZmI1NTZiNWViIn0
```

Compute the ***JWS-signature:***

1. Compute the following string: *Base64URL(JWS Header) '.' Base64URL(JWS Payload)*

The result will be:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZjMz
UwMWUzYWM4MjIxZmVjZTAzIn0.
eyJTSEEyNTYiOiIxYzhlOWUyMDJhNDU3OWI3MDU2MTE1ZDExYjA4M2U4MjBlZTBiOWE5MDA
wNmYyYmQ2MWYxY2NlZmI1NTZiNWViIn0
```

2. Sign the string using the TPP private key and apply Base64 encoding using the following command:

```
echo|set
/p="eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjOD
ZjMzUwMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiIxYzhlOWUyMDJhNDU3OWI3MDU2M
TE1ZDExYjA4M2U4MjBlZTBiOWE5MDAwNmYyYmQ2MWYxY2NlZmI1NTZiNWViIn0" | openssl
dgst -sha256 -sign SC_EXEMPLU_SRL.key | openssl base64 -e -A
```

The result will be:

```
gB44MECSWqWTQH44zd3PC0S-JczV8UT8IFsIXg8AEpI8SpNuSDNRm-
KAaL2MKA69yfqQT_OPXO8aUAgZxCpcHcRLVFuiulo6MPi5HkmEoCUubVBceDUqMAzI4L3H0Aiq
TicL7Qio5NWG-V-
066PQaUNvaTr7iLU7Sn1TiBW_kg1a4FiWDabuFokp5SOiBdBHhDQ_d4cbfPj927b97_vOxqVW2
wiJ1z1Bh6worO051AWvgqW7moDxCv_L5PFcemlV_75uRGOUVX9nGdHulSL6wB83Mi5gbnLG
agGif9MKy1KPa8mAaV-wLUkmZHx_GJpayp7Tcfs53GnzA3Xabw7OA
```

3. Finally, the Header JWS-signature will be:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZjMz
UwMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiIxYzhlOWUyMDJhNDU3OWI3MDU2MTE1Z
DExYjA4M2U4MjBlZTBiOWE5MDAwNmYyYmQ2MWYxY2NlZmI1NTZiNWViIn0.gB44MECSWq
WTQH44zd3PC0S-JczV8UT8IFsIXg8AEpI8SpNuSDNRm-
KAaL2MKA69yfqQT_OPXO8aUAgZxCpcHcRLVFuiulo6MPi5HkmEoCUubVBceDUqMAzI4L3H0Aiq
TicL7Qio5NWG-V-
066PQaUNvaTr7iLU7Sn1TiBW_kg1a4FiWDabuFokp5SOiBdBHhDQ_d4cbfPj927b97_vOxqVW2
wiJ1z1Bh6worO051AWvgqW7moDxCv_L5PFcemlV_75uRGOUVX9nGdHulSL6wB83Mi5gbnLG
agGif9MKy1KPa8mAaV-wLUkmZHx_GJpayp7Tcfs53GnzA3Xabw7OA
```

```
{
 "transactionStatus": "ACSC"
}
```

Request Response

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

Service 2 (Get Payment information)

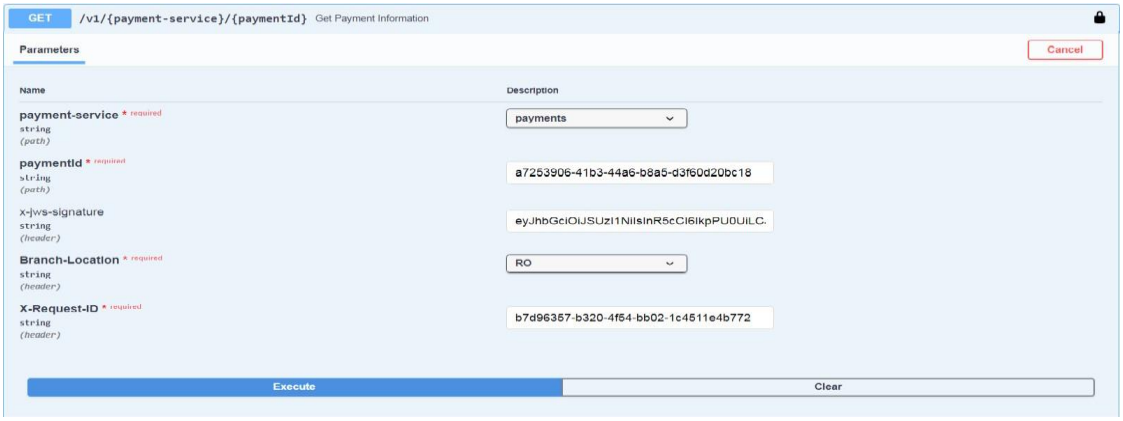


Figure 13

Steps

Compute the ***Base64URL (JWS Payload):***

1. Create the following JSON on a single line without spaces:

```
{"headers":{"Branch-Location":"RO","X-Request-ID":"b7d96357-b320-4f54-
bb021c4511e4b772"},"payload":{"payment_id":"a7253906-41b3-44a6-b8a5-d3f60d20bc18"}}
```

2. Apply SHA-256 on the JSON from step 1 using the following command (on Windows OS):

```
echo|set    /p="{"headers":{"Branch-Location":"RO","X-Request-ID":"b7d96357-b320-4f54-
bb021c4511e4b772"},"payload":{"payment_id":"a7253906-41b3-44a6-b8a5-d3f60d20bc18"}}"
    | openssl dgst -sha256
```

The result will be:

```
1c8e9e202a4579b7056115d11b083e820ee0b9a90006f2bd61f1ccefb556b5eb
```

3. Create the following JSON with the result:

```
{"SHA256":" 1c8e9e202a4579b7056115d11b083e820ee0b9a90006f2bd61f1ccefb556b5eb"}
```

4. Compute Base64 encoding on the later JSON using the following command (on Windows OS):

```
echo|set
/p="{"SHA256":"1c8e9e202a4579b7056115d11b083e820ee0b9a90006f2bd61f1ccefb556b5eb"
}" | openssl base64 -e –A
```

5. Replace any occurence of “+" character with “-" and any occurence of "/" character with “_". Also, delete every “=" from the resulted string. The result in our test case is:
eyJTSEEyNTYiOiIxYzhlOWUyMDJhNDU3OWI3MDU2MTE1ZDExYjA4M2U4MjBlZTBiOWE5MDAwNmYyYmQ2MWYxY2NlZmI1NTZiNWViIn0

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

Compute the ***JWS-signature:***

1. Compute the following string: *Base64URL(JWS Header) '.' Base64URL(JWS Payload)*

   The result will be:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZjMzUwMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiIxYzhlOWUyMDJhNDU3OWI3MDU2MTE1ZDExYjA4M2U4MjBlZTBiOWE5MDAwNmYyYmQ2MWYxY2NlZmI1NTZiNWViIn0
```

2. Sign the string using the TPP private key and apply Base64 encoding using the following command:

```
echo|set /p="eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZjMzUwMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiIxYzhlOWUyMDJhNDU3OWI3MDU2MTE1ZDExYjA4M2U4MjBlZTBiOWE5MDAwNmYyYmQ2MWYxY2NlZmI1NTZiNWViIn0" | openssl dgst sha256 -sign SC_EXEMPLU_SRL.key | openssl base64 -e -A
```

The result will be:

```
gB44MECSWqWTQH44zd3PC0S-JczV8UT8IFsIXg8AEpI8SpNuSDNRm-KAaL2MKA69yfqQT_OPXO8aUAgZxCpcHcRLVFuiulo6MPi5HkmEoCUubVBceDUqMAzI4L3H0AiqTicL7Qio5NWG-V-066PQaUNvaTr7iLU7Sn1TiBW_kg1a4FiWDabuFokp5SOiBdBHhDQ_d4cbfPj927b97_vOxqVW2wiJ1z1Bh6worO051AWvgqW7moDxCv_L5PFcemlV_75uRGOUVX9nGdHulSL6wB83Mi5gbnLGagGif9MKy1KPa8mAaV-wLUkmZHx_GJpayp7Tcfs53GnzA3Xabw7OA
```

3. Finally, the Header JWS-signature will be:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJraWQiOiIxMzNjMTE0NzA3NDBkN2VkMzNjODZjMzUwMWUzYWM4MjIxZmVjZTAzIn0.eyJTSEEyNTYiOiIxYzhlOWUyMDJhNDU3OWI3MDU2MTE1ZDExYjA4M2U4MjBlZTBiOWE5MDAwNmYyYmQ2MWYxY2NlZmI1NTZiNWViIn0.gB44MECSWqWTQH44zd3PC0S-JczV8UT8IFsIXg8AEpI8SpNuSDNRm-KAaL2MKA69yfqQT_OPXO8aUAgZxCpcHcRLVFuiulo6MPi5HkmEoCUubVBceDUqMAzI4L3H0AiqTicL7Qio5NWG-V-066PQaUNvaTr7iLU7Sn1TiBW_kg1a4FiWDabuFokp5SOiBdBHhDQ_d4cbfPj927b97_vOxqVW2wiJ1z1Bh6worO051AWvgqW7moDxCv_L5PFcemlV_75uRGOUVX9nGdHulSL6wB83Mi5gbnLGagGif9MKy1KPa8mAaV-wLUkmZHx_GJpayp7Tcfs53GnzA3Xabw7OA
```

```
{
 "debtorAccount": {
  "iban": "RO49AAAA1B31007593840000"
 },
 "instructedAmount": {
```


© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

```
  "currency": "RON",
  "amount": "101"
 },
 "creditorAccount": {
  "iban": "RO61TREZ27A660404200109X"
 },
 "creditorName": "PaySafe",
 "transactionStatus": "ACSC"
}
```


Request Response

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

4. Payment resource initiation

Register a payment resource

4.1. Resource Information

Method	Purpose
Response formats	JSON
Requires authentication	No
Rate limited	Yes
Requests	15

4.2. Request

Method	URL
POST	https://sandbox.banorientfrance.com/v1/{paymentservice}/{payment-product}

4.1. Parameters

Path Parameter	Required
payment-service	Mandatory
payment-product	Mandatory

Header Parameter	Required
x-jws-signature	Mandatory
Branch-Location	Mandatory
X-Request-ID	Mandatory
PSU-ID	Optional
PSU-ID-Type	Optional
PSU-Corporate-ID	Optional

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

PSU-Corporate-ID-Type	Optional
Consent-ID	Optional
PSU-IP-Address	Mandatory
TPP-Redirect-Preferred	Optional
TPP-Redirect-URI	Optional
TPP-Nok-Redirect-URI	Optional
TPP-Explicit-Authorisation-Preferred	Optional

Important

The Request Body might not contain the debtorAccount JSON field as it is not mandatory in our case. The user must select the account used to make the payment before he confirms the transaction.


```
{
 "endToEndIdentification": "string",
 "debtorAccount": {
  "iban": "string",
  "bban": "string",
  "pan": "string",
  "maskedPan": "string",
  "msisdn": "string",
  "currency": "string"
 },
 "debtorId": "string",
 "ultimateDebtor": "string",
 "instructedAmount": {
  "currency": "string",
  "amount": "string"
 },
 "transactionCurrency": "string",
```


© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

```
"creditorAccount": {
  "iban": "string",
  "bban": "string",
  "pan": "string",
  "maskedPan": "string",
  "msisdn": "string",
  "currency": "string"
},
"creditorAgent": "string",
"creditorAgentName": "string",
"creditorName": "string",
"creditorId": "string",
"creditorAddress": {
  "street": "string",
  "buildingNumber": "string",
  "city": "string",
  "postalCode": "string",
  "country": "string"
},
"ultimateCreditor": "string",
"purposeCode": "string",
"chargeBearer": "string",
"remittanceInformationUnstructured": "string",
"remittanceInformationUnstructuredArray": [
  "string"
],
"remittanceInformationStructured": {    "reference": "string",
  "referenceType": "string",
```


© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

```
  "referenceIssuer": "string"
 },
 "requestedExecutionDate": {},
 "requestedExecutionTime": {}
}
```


Request Body


```
{
"endToEndIdentification": "test",
"instructedAmount":{ "currency":"RON", "amount":"101" },
"creditorAccount":{"iban":"RO61TREZ27A660404200109X"}, "creditorName":"PaySafe"
}
```


Request Example


```
{
 "transactionStatus": "RCVD",
 "paymentId": "f5d208af-d9f2-4eab-af9e-570e515278c2",
 "_links": {
  "scaOAuth": "https://sandbox.banorientfrance.com/services/startAuthorize"
 }
}
```


Request Response

The response includes the paymentId resource created and the URL link for the user authentication/authorization step.

5. Retrieves access token

This service exchange the authorization code for access token and is the final step of OAuth2 authorization code flow.

For complete description of the OAuth2 flow please follow section 4 Testing a payment flow example.

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

5.1. Resource Information

Method	Purpose
Response formats	JSON
Requires authentication	Yes
Rate limited	Yes
Requests	15

5.2. Request

Method	URL
POST	https://sandbox.banorientfrance.com/token

5.3. Parameters

Header Parameter	Required
x-jws-signature	Mandatory

5.4. Request Body

Parameter	Required
grant_type	Mandatory
client_id	Mandatory
client_secret	Mandatory
code	Mandatory
payment_id	Mandatory


```
{
 "refresh_token": "PcHoHIcfb5ytbhV2OGZK5TPZGNrLpJAP",
 "token_type": "bearer",
 "access_token": "8714WYCTnTLDxyZtyRWNy7FaAhART4zH",
 "expires_in": 7776000
}
```


Request Response

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

The response includes the access token and the refresh token and the expiration period.

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

6. Content of a payment object

Returns the content of a payment object

6.1. Resource Information

Method	Purpose
Response formats	JSON
Requires authentication	Yes
Rate limited	Yes
Requests	15

6.2. Request

Method	URL
GET	https://sandbox.banorientfrance.com/v1/{paymentservice}/{paymentId}

6.1. Parameters

Path Parameter	Required
payment-service	Mandatory
paymentId	Mandatory

Header Parameter	Required
x-jws-signature	Mandatory
Branch-Location	Mandatory
X-Request-ID	Mandatory

```
{
 "debtorAccount": {
   "iban": "RO49AAAA1B31007593840000"
 },
 "instructedAmount": {
```


© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

```
  "currency": "RON",
  "amount": "101"
 },
 "creditorAccount": {
  "iban": "RO61TREZ27A660404200109X"
 },
 "creditorName": "PaySafe",
 "transactionStatus": "ACSC"
}
```


Request Response

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA

7. Checks the status of a payment initiation

Identify the resource and describe its purpose.

7.1. Resource Information

The resource information is as follows:

Method	Purpose
Response formats	JSON
Requires authentication	Yes
Rate limited	Yes
Requests	15

7.2. Request

Method	URL
GET	https://sandbox.banorientfrance.com/v1/{paymentservice}/{paymentId}/status

7.3. Parameters

Path Parameter	Required
payment-service	Mandatory
paymentId	Mandatory

Header Parameter	Required
x-jws-signature	Mandatory
Branch-Location	Mandatory
X-Request-ID	Mandatory

```
{
  "transactionStatus": "ACSC"
}
```

Request Response

© BANQUE BANORIENT FRANCE S.A. SUCURSALA ROMANIA