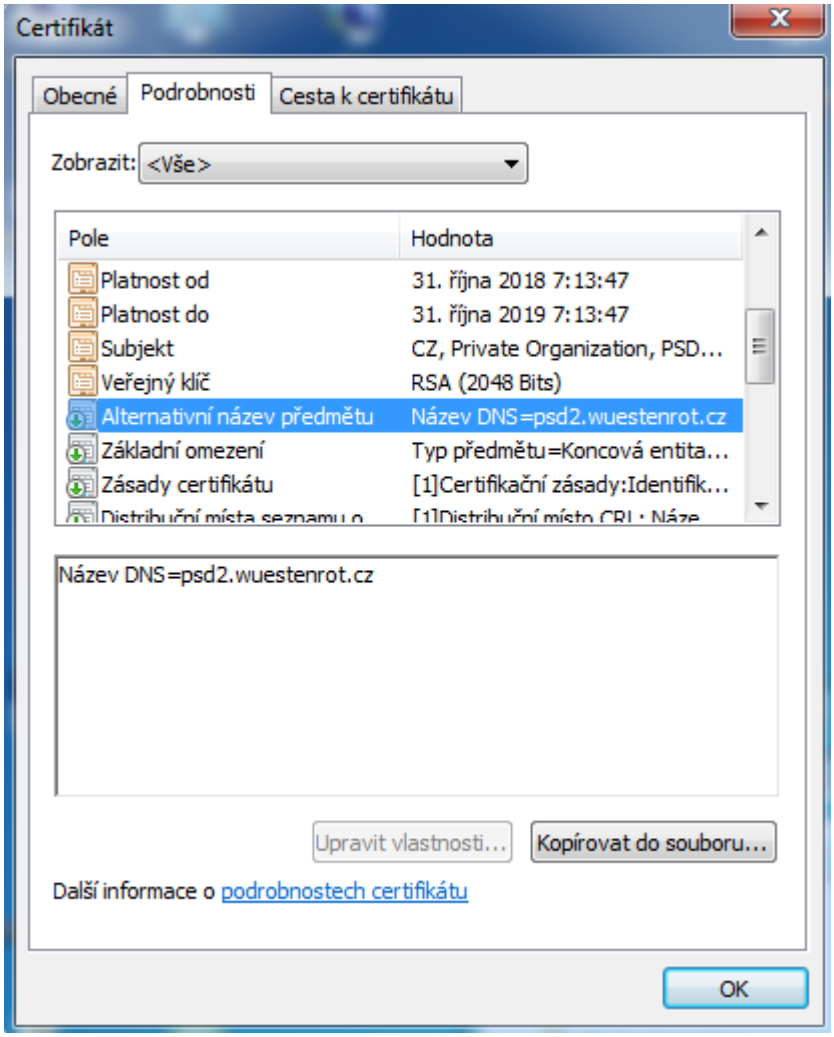
How to call PSD2 API

Contents

How to call PSD2 API 1
Prerequisites 1
Registering into developer portal 2
How to convert cert.pfx to cert.pem 3
Errors 3
Creating the application 3
API Subscription 5
Wait for a while 6
How to call AISP APIs 6
Authorization request 6
Get accounts example call 7
Account balance example call 8
Transaction history example call 9
How to call PISP APIs 10
Payment initiate 10
Strong customer authentication with dynamic linking 12
Payment confirm call 14
Refresh token 14
Sandbox 14
State parameter in authorization request 15
Technical contact 16
Payment complaints contact 16
Information about status and planned outages 16

Prerequisites

- Qualified certificate for PSD2 authentication with specified metadata (registrationNumber, registrator and roles)
- Ability to make OAuth2 client credentials grant and authorization code



Registering into developer portal

Use this TPPPublicRegistration API to register your company as a PSD2 Third Party Provider. Registration will result to Wüstenrot developer portal access under user that is allowed to call API's.

Usage:

```
curl -k -X POST --cert WuestenrotTest.pem:aaaa -H "accept: application/json" -
H "Content-Type: application/json"
"https://apipsd2.wuestenrot.cz/tppregistration/1.1.14/tpps"
-d @tppRegister.json
```

where tppRegister.json contains your data (be sure your character encoding is UTF-8)

```
{
  "tppId": "TRO011",
  "tppName": "Velké finance",
  "password": "xxxxx",
  "email": "trosa@trask.cz",
  "address": "Po přístřeším 87",
  "phoneNumber": "+420 123456787"
```

```
}
```

tppId is userName to the developer portal. It can't contain spaces, underscore or diacritics.

password is password to the identity server with policy pattern: ^((?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$%&*])).{0,100}$

email is required for confirmation email and notification purposes. You will receive email with confirmation link after successful registration.

address and **phone** number is required. The administrators of Wuestenrot will contact you for security reasons. Wrong information will be reported to local authority.

How to convert cert.pfx to cert.pem

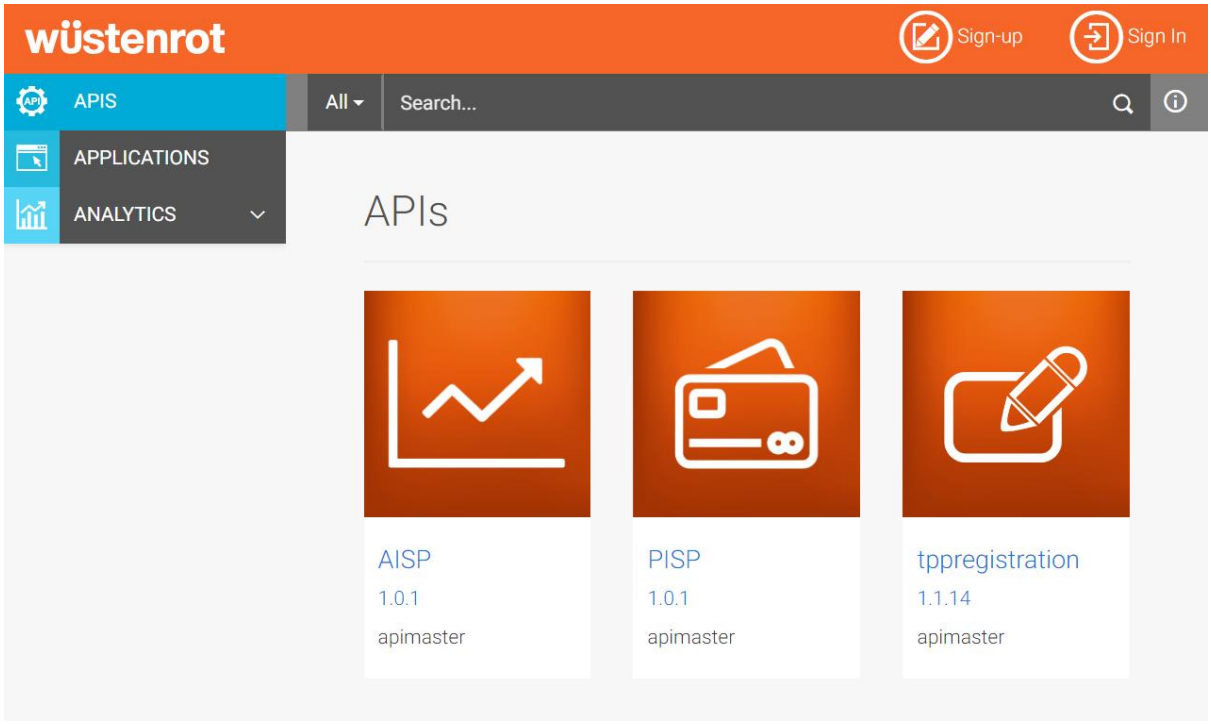
```
openssl.exe pkcs12 -in WuestenrotTest.pfx -out WuestenrotTest.pem -clcerts
```

Errors

http error	Error message	Description
403	User xxxx already exists in the system. Please use a different username.	Chosen tppId has been already registered. Try a different one.
400	The certificate is missing valid metadata	The certificate does not meet the expectations (see the prerequisites).
400	Some of following parameters is missing in JSON request: tppId or password or tppName or email or phoneNumber	The JSON in the body is not valid or complete.
400	The name should contain character '_'	
403	Unable to process registration. The TPP has been already registered under a different tppId: xxx	A different user with provided certificate is already registered.
200	User has been already registered. No changes has been done.	
403	The TPP has been already registered	
500	Unable to register user - internal error - contact support please	Internal error.
500	Unable to set roles	Internal error.

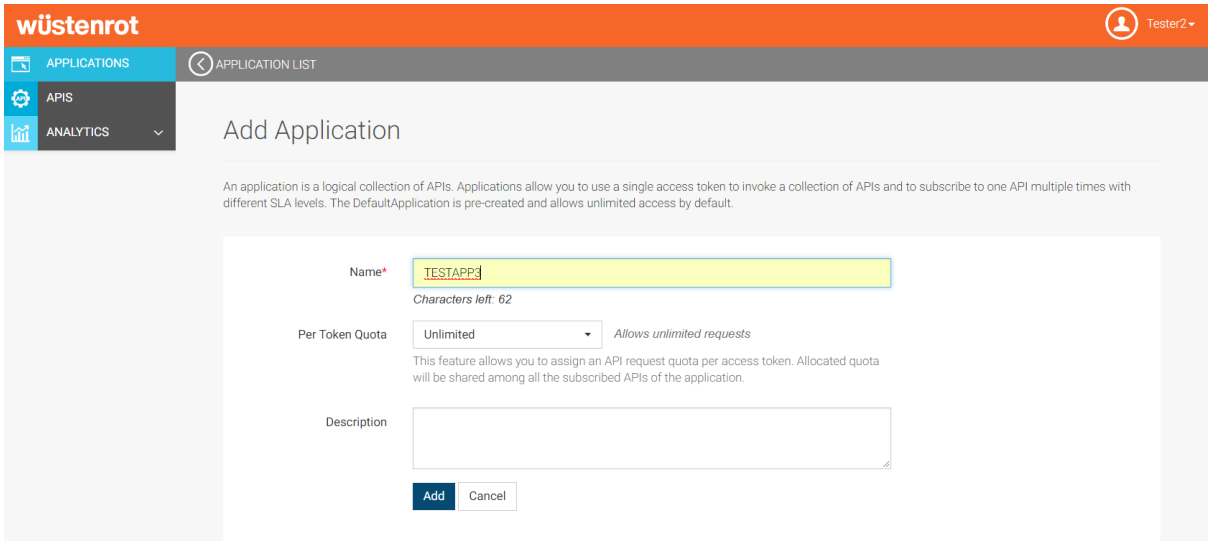
Creating the application

Open page https://developers.wuestenrot.cz and **sign in** to the developer portal. Please note that **sign-up** creates user that is **unable to call API**!



Go to the Applications and create your application. (You can use the default application, however the application name will be displayed on consent page to the end users).

Pay attention that application name can't contain underscore!

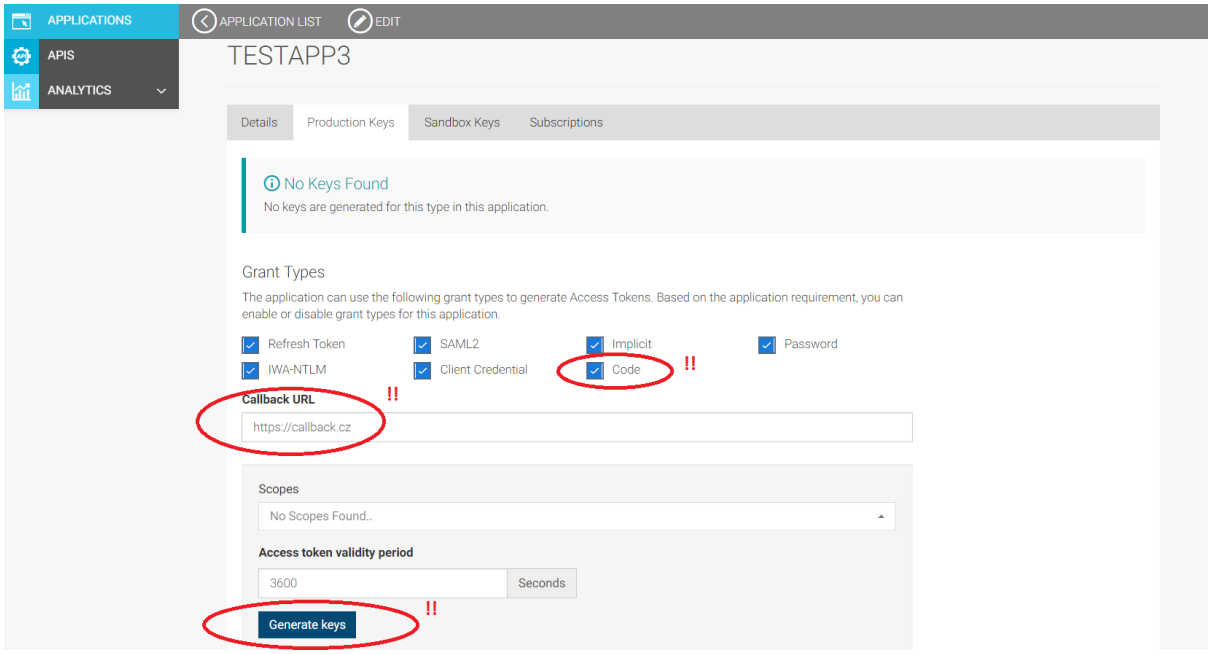


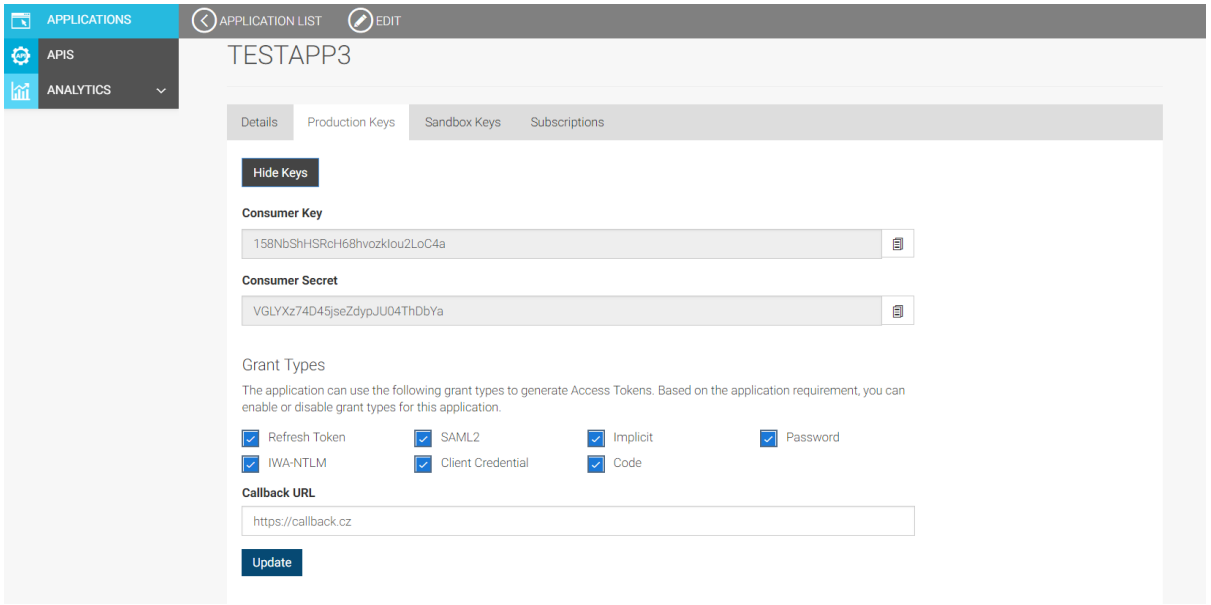
Per Token Quota field allows you to assign an API request quota per access token. Allocated quota will be shared among all the subscribed APIs of the application. You can leave the unlimited value.

Description is up to you, but it will not be displayed to the end users.

Entry CallbackURL for OAuth2 implicit and authorization grant code purposes. Note that Password grant is useless, client credentials can be used in certain situations only (e.g. payment initiate).

Then push Generate keys button to obtain clientId (consumerKey) and clientSecret (consumerSecret).

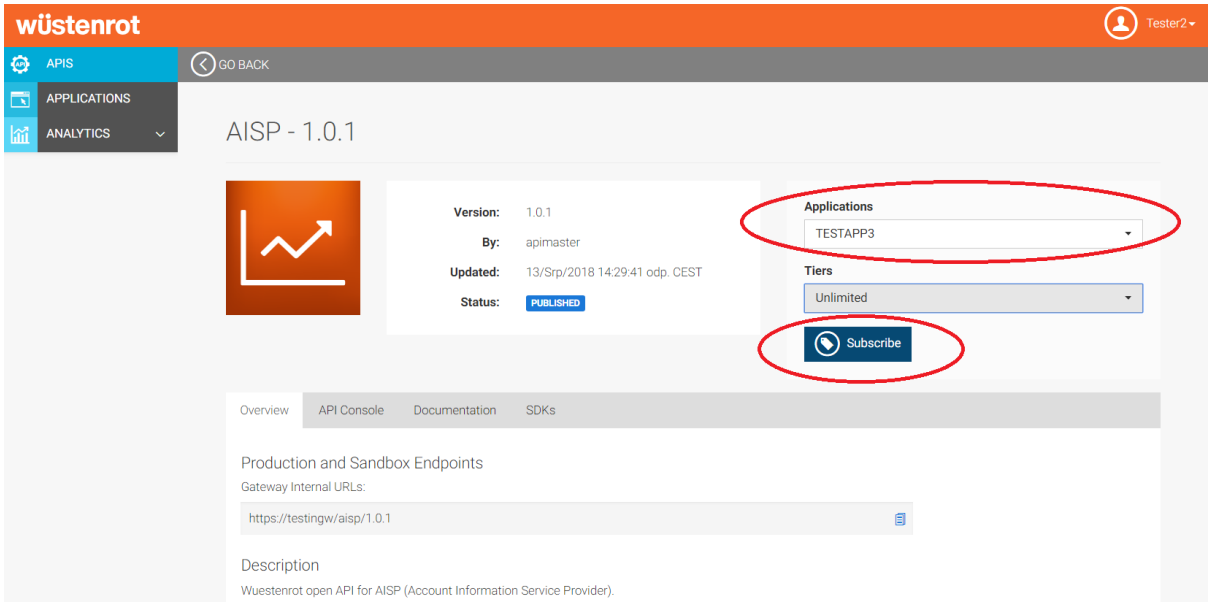




The usage of clientId and clientSecret generated on sandbox tab means that mock of back ends and authentication server is used.

API Subscription

The application has to be subscribed to API in order to make the API callable with tokens that has been created by clientId and clientSecret.



Wait for a while

The current situation required administrator assistance to set up authentication to your profile. Until that you will not be able to make OAuth2 authorization grant code (or implicit). We work on progress.

How to call AISP APIs

The API call requires:

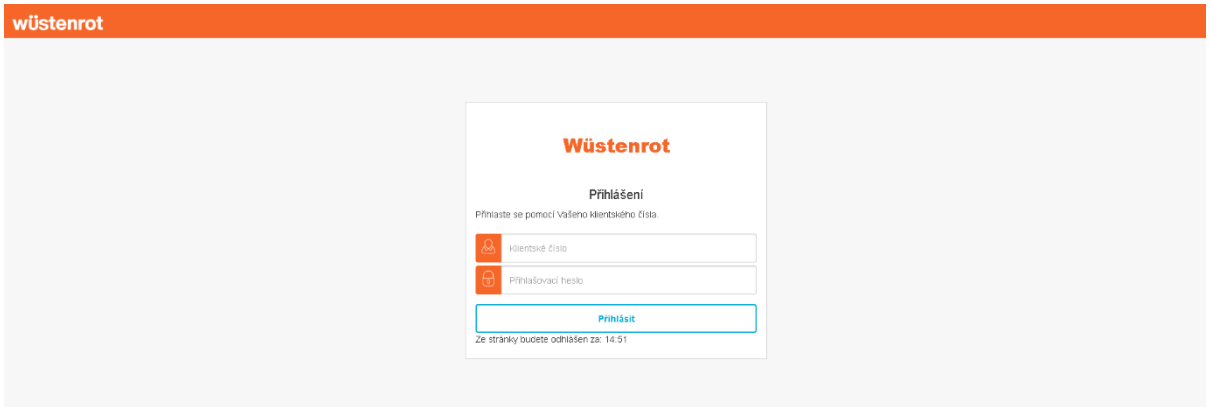
- Valid qualified certificate for TLS client authentication
- Oauth2 Access token with grant code (or implicit) and scope `AISP`

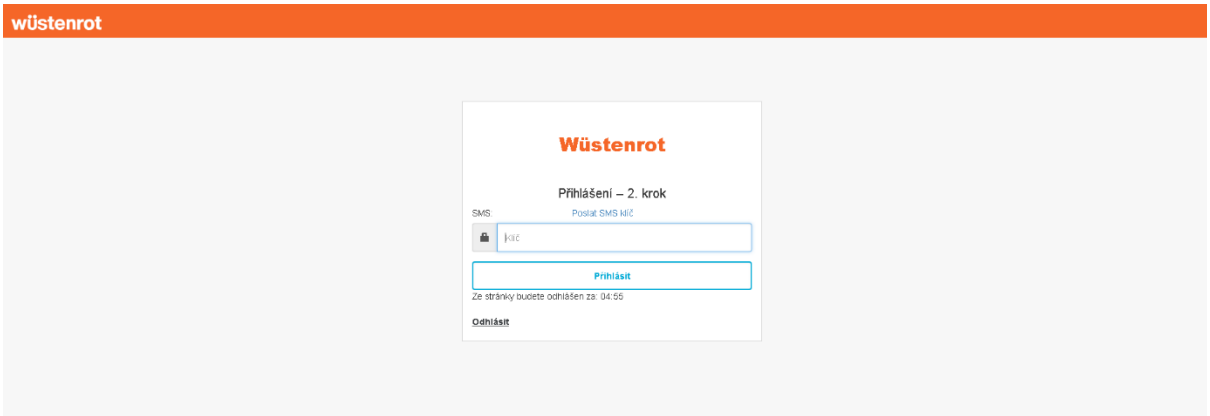
Authorization request

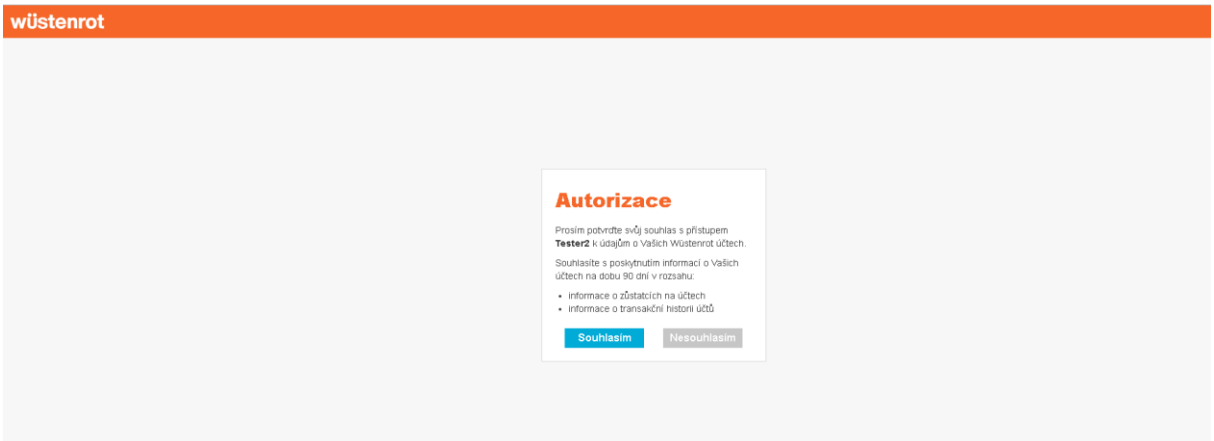
TPP application redirects the end user (browser) to https://apipsd2.wuestenrot.cz/authorize

```
https://apipsd2.wuestenrot.cz/authorize?scope=AISP&response_type=code&redirect_uri=http://callback.cz&client_id=<clientId>
```

End users goes through strong customer authentication







The TPP aplication receives code on redirect_uri and makes direct call to https://apipsd2.wuestenrot.cz/token to retrieve access token e.g.: http://callback.cz?code=6e215c0d-850a-375d-b440-4b32ab5a0221.

```
curl -k -u clientId:clientSecret -d
"grant_type=authorization_code&code=cc9c7ffc-6900-3e0e-a525-
cb98438a328a&redirect_uri=http://callback.cz"
https://apipsd2.wuestenrot.cz/token
```

If everything is correct the server returns access and refresh token.

```
{"access_token":"52ae61ce-121c-3ae5-a7ab-
3d6801d86021","refresh_token":"da256707-4676-3785-a516-
3f1b05812798","scope":"AISP","token_type":"Bearer","expires_in": 3600}
```

The clientId corresponds to consumerKey, clientSecret corresponds to consumerSecret in developer portal.

Redirect uri must be the same as registered in the application under the same clientId.

Code expires within 5 minutes.

If you would make get_accounts request or transaction_history request with fromDate older than 90 days, you need a SCA that is not older than 5 minutes!

Get accounts example call

Request:

```
curl -k -X GET --cert WuestenrotTest.pem:aaaa --header "Accept:
application/json" --header "Authorization: Bearer 52ae61ce-121c-3ae5-a7ab-
3d6801d86021" "https://apipsd2.wuestenrot.cz/aisp/1.0.1/accounts"
```

Response:


```
{
    "accounts": [
        {
            "identification": {
                "other": "0411095884",
                "iban": "CZ8779800000000411011111"
            },
            "nameI18N": "Janko Hraško",
            "servicer": {
                "bankCode": 7980,
                "countryCode": "CZ",
                "bic": null
            },
            "currency": "CZK",
            "id": "CZ8779800000000411011111",
            "productI18N": "Wüstenrot Spořicí účet"
        }
    ]
}
```


Account balance example call

Request:

```
curl -k -X GET --cert WuestenrotTest.pem:aaaa --header "Accept:
application/json" --header "Authorization: Bearer 52ae61ce-121c-3ae5-a7ab-
3d6801d86021"
"https://apipsd2.wuestenrot.cz/aisp/1.0.1/accounts/CZ8779800000000411011111/ba
lance"
```

Response:


```
{
    "balances": [
        {
            "date": {
                "dateTime": "2018-08-31T13:45:15.000Z"
            },
            "amount": {
                "currency": "CZK",
                "value": 218626.54
            },
            "type": {
                "codeOrProprietary": {
                    "code": "ITBD"
                }
```

```
            },
            "creditDebitIndicator": "CRDT"
        }
    ]
}
```


Transaction history example call

Request:


```
curl -k -X GET --cert WuestenrotTest.pem:aaaa --header "Accept:
application/json" --header "Authorization: Bearer 52ae61ce-121c-3ae5-a7ab-
3d6801d86021"
"https://apipsd2.wuestenrot.cz/aisp/1.0.1/accounts/CZ8779800000000411011111/tr
ansactions?fromDate=2018-07-30&toDate=2018-08-05&size=100&page=0"
```


Notice: All query attributes fromDate, toDate, size and page are mandatory.

Response:


```
{
    "pageCount": 1,
    "pageNumber": 0,
    "nextPage": null,
    "pageSize": 100,
    "transactions": [
        {
            "reversalIndicator": false,
            "amount": {
                "currency": "CZK",
                "value": 5.57
            },
            "bankTransactionCode": null,
            "bookingDate": null,
            "valueDate": {
                "date": "2018-07-31T21:59:59.000Z"
            },
            "entryDetails": {
                "transactionDetails": {
                    "references": null,
                    "relatedAgents": null,
                    "purpose": null,
                    "remittanceInformation": null,
                    "additionalTransactionInformation": null,
                    "relatedParties": {
                        "debtorAccount": {
                            "identification": {
                                "other": null,
                                "iban": "CZ8779800000000411011111"
                            }
                        },
```

```
                        "debtor": {
                            "name": "Bánovská Renata",
                            "postallAddress": null
                        },
                        "creditorAccount": {
                            "identification": {
                                "other": {
                                    "identification": "0000000953214905/null"
                                },
                                "iban": null
                            }
                        },
                        "creditor": {
                            "name": "Daňový účet pro BÚ",
                            "postallAddress": null
                        }
                    },
                    "amountDetails": null
                }
            },
            "creditDebitIndicator": "DBIT",
            "entryReference": null,
            "status": "BOOK"
        },
        {
            "reversalIndicator": false,
…
…
            "creditDebitIndicator": "CRDT",
            "entryReference": null,
            "status": "BOOK"
        }
    ]
}
```


How to call PISP APIs

PISP sequence consist of three basic steps:

1. Payment initiate
2. Strong customer authentication with dynamic linking
3. Payment confirm

Payment initiate

It uses access token that has been retrieved from client credentials grant authentication.

```
curl -k -d "grant_type=client_credentials" -u clientId:clientSecret
https://apipsd2.wuestenrot.cz/token
```

Then it is possible to call API

```
curl -k -X POST --cert WuestenrotTest.pem:aaaa --header "Content-Type:
application/json" --header "Accept: application/json" --header "Authorization:
Bearer 23c6fe64-604b-3da5-9151-8dd0c2896b07" -d @PISP.json
"https://apipsd2.wuestenrot.cz/pisp/1.0.1/payments"
```

PISP.json:

```
{
    "paymentIdentification": {
        "instructionIdentification": "124WC555",
        "endToEndIdentification": "string"
    },
    "debtorAccount": {
        "identification": {
            "iban": "CZ8779800000000411011111"
        },
        "currency": "CZK"
    },
    "amount": {
        "instructedAmount": {
            "currency": "CZK",
            "value": 1000
        }
    },
    "requestedExecutionDate": "2018-09-01",
    "remittanceInformation": {
        "unstructured": "Platba",
        "structured": {
            "creditorReferenceInformation": {
                "reference": "VS:54321"
            }
        }
    },
    "creditorAccount": {
        "identification": {
            "iban": "CZ6508000000192000145399"
        },
        "currency": "CZK"
    },
    "paymentTypeInformation": {
        "instructionPriority": "NORM"
    }
}
```

Response

```
{
```

```
    "paymentIdentification": {
        "instructionIdentification": "124WC555",
        "transactionIdentification": "PSD15355343543190023"
    },
    "debtorAccount": {
        "identification": {
            "iban": " CZ8779800000000411011111"
        },
        "currency": "CZK"
    },
    "amount": {
        "instructedAmount": {
            "currency": "CZK",
            "value": 12
        }
    },
    "requestedExecutionDate": "2018-08-30",
    "remittanceInformation": {
        "unstructured": "Platba",
        "structured": {
            "creditorReferenceInformation": {
                "reference": "VS:54321"
            }
        }
    },
    "creditorAccount": {
        "identification": {
            "iban": "CZ6508000000192000145399"
        },
        "currency": "CZK"
    },
    "paymentTypeInformation": {
        "instructionPriority": "NORM",
        "serviceLevel": {
            "code": "DOMESTIC"
        }
    },
    "signInfo": {
        "state": "Open",
        "signId": "PSD15355343543190023"
    },
    "instructionStatus": "ACTC"
}
```


Strong customer authentication with dynamic linking

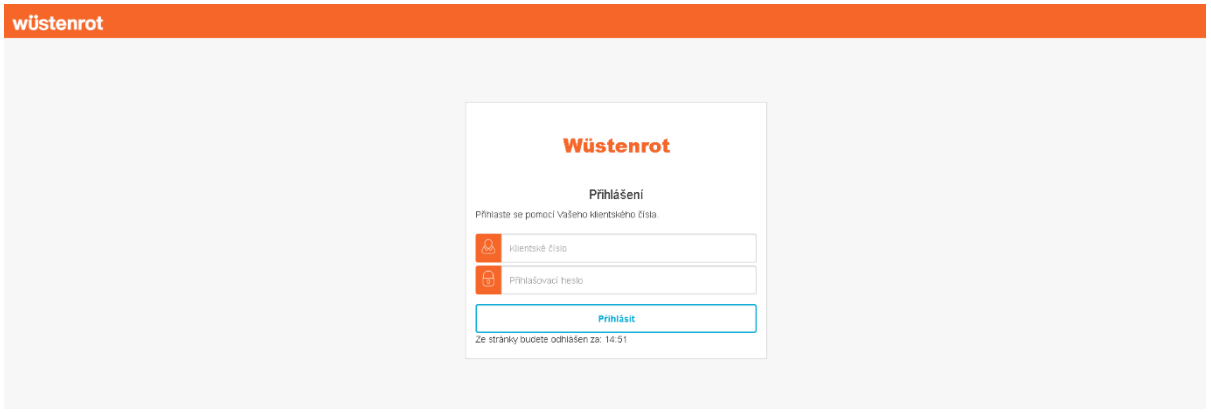
The SCA with dynamic linking is based on OAuth2 authorization code grant with passing new query paramenter operation_id that corresponds to transactionIdentification in previous reply.

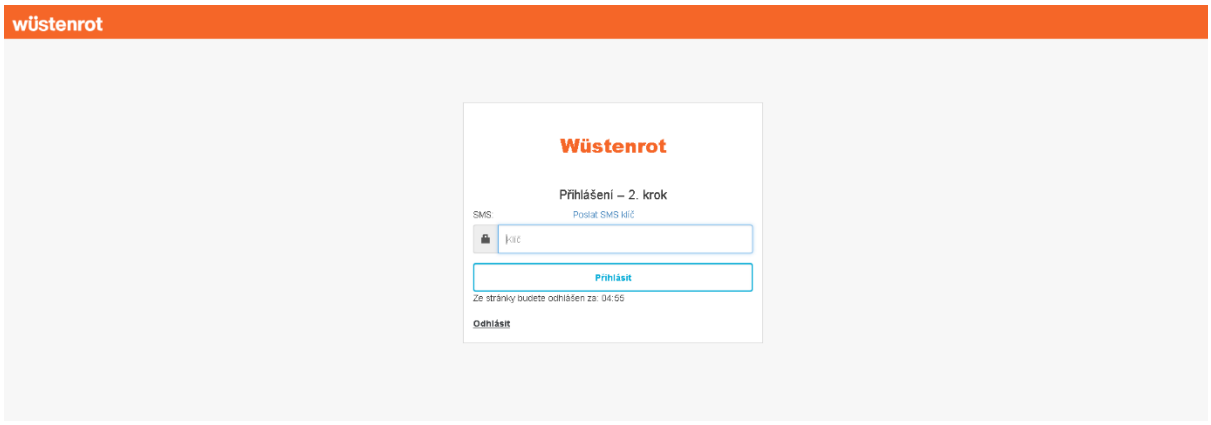
TPP application redirects the end user (browser) to https://apipsd2.wuestenrot.cz/authorize

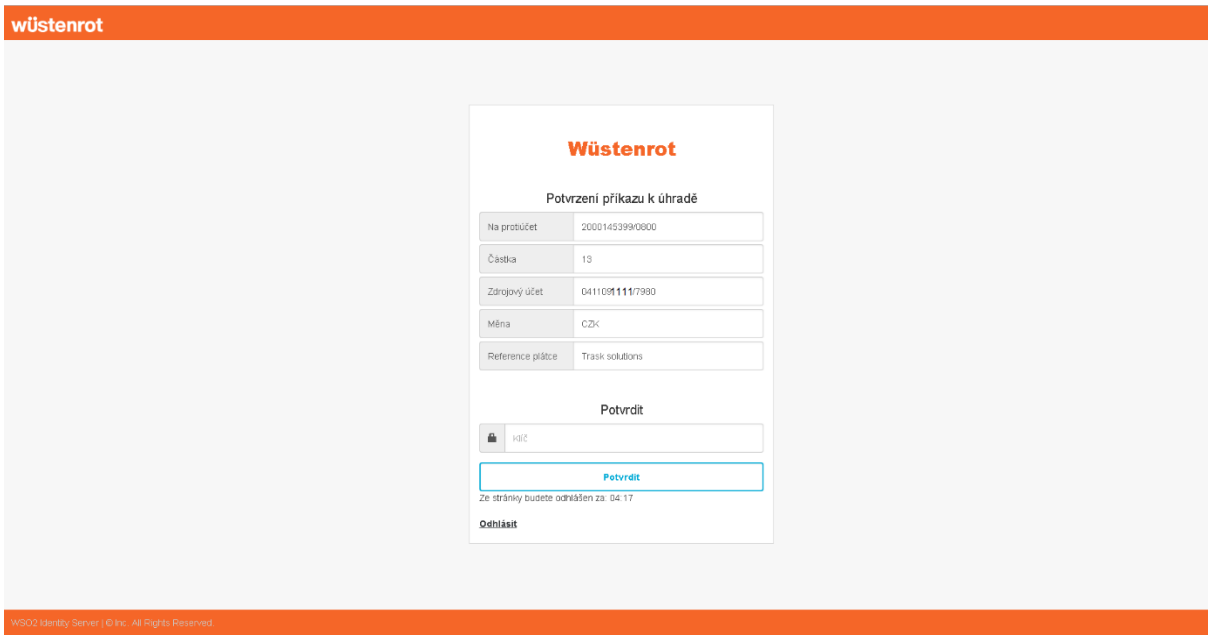
```
https://apipsd2.wuestenrot.cz/authorize?scope=PISP&response_type=code&redirect_uri=
https://callback.cz&client_id=<clientId>&operation_id=PSD153553435431900023
```

Where **operation_id** is **transactionIdentification** from previous response!

End users goes through strong customer authentication







The TPP aplication receives code on redirect_uri and makes direct call to https://apipsd2.wuestenrot.cz/token to retrieve access token e.g.: https://callback.cz?code=6e215c0d-850a-375d-b440-4b32ab5a0221.

```
curl -k -u clientId:clientSecret -d
"grant_type=authorization_code&code=cc9c7ffc-6900-3e0e-a525-
cb98438a328a&redirect_uri=http://callback.cz"
https://apipsd2.wuestenrot.cz/token
```

If everything is correct the server returns access and refresh token.

```
{"access_token":"52ae61ce-121c-3ae5-a7ab-
3d6801d86021","refresh_token":"da256707-4676-3785-a516-
3f1b05812798","scope":"PISP","token_type":"Bearer","expires_in": 3600}
```

The clientId corresponds to consumerKey, clientSecret corresponds to consumerSecret in developer portal.

Redirect uri must be the same as registered in the application under the same clientId.

Code expires within 5 minutes.

Payment confirm call

Request:

```
curl -k -X PUT --cert WuestenrotTest.pem:aaaa --header "Accept:
application/json" --header "Authorization: Bearer 52ae61ce-121c-3ae5-a7ab-
3d6801d86021"
"https://apipsd2.wuestenrot.cz/pisp/1.0.1/payments/PSD15355343543190023" -d
"{}"
```

Response:

```
{"instructionStatus":"ACSP"}
```

Refresh token

The access token is available only 3600 seconds. This is the reason why the credential code grant provides the refresh token besides access token. The refresh token is valid 90 days and can be used to retrieve new access token by OAuth refresh grant.

```
curl.exe -v -k -u <clientId>:<clientSecret> -d
"grant_type=refresh_token&refresh_token=<refresh_token>" -H "Content-Type:
application/x-www-form-urlencoded" https://apipsd2.wuestenrot.cz/token
```

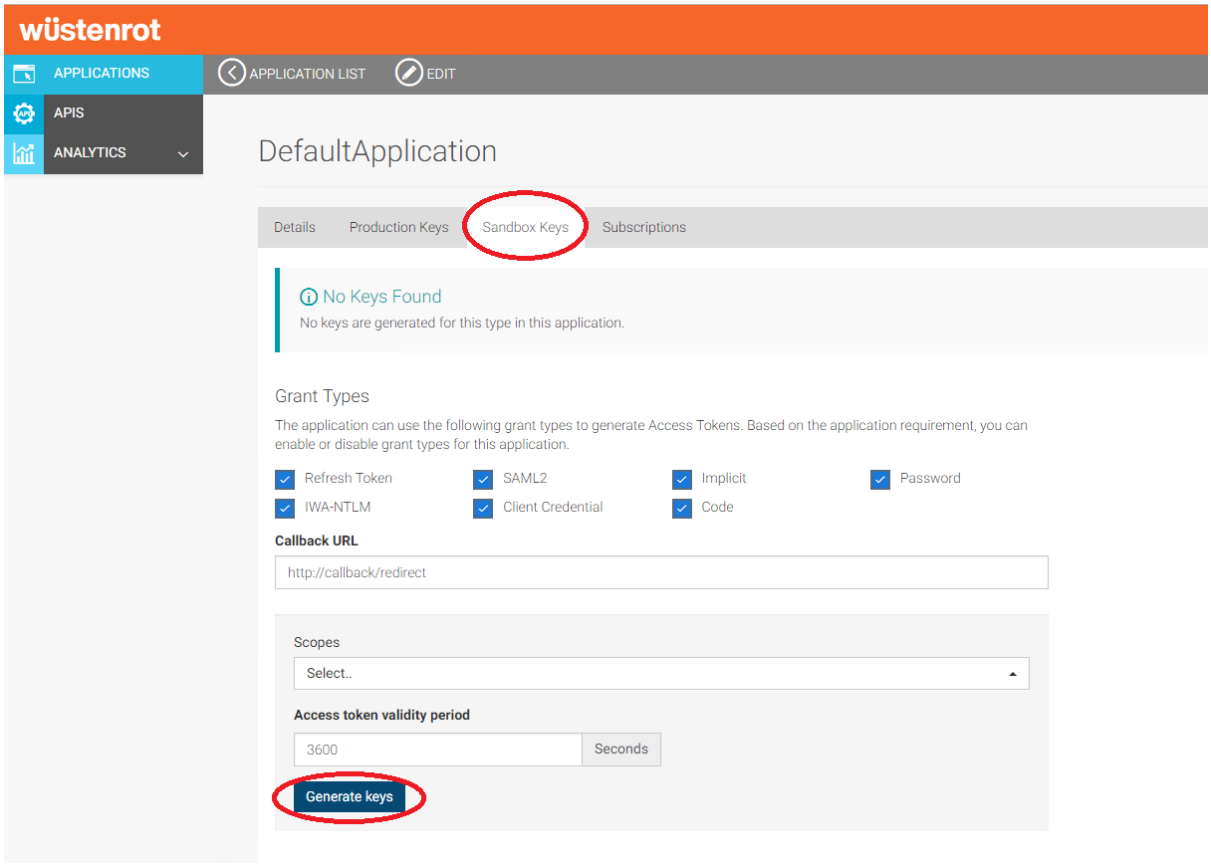
Note: The refresh token that is returned by refresh grant is the same until its expiration, it cannot be renewed.

Note2: If you make SCA with token request before the expiration you receive the same token and the expiration time is not reset.

Sandbox

Sandbox is environment determined for application development and testing by authorized users.

The sandbox usage requires the registration with production certificate. Ones you have successfully registered, you can choose between production and sandbox by using different set of OAuth keys. The sandbox clinetId and clientSecret is available on the “Sandbox Keys” tab of application management.



The following table provides the predefined properties of two sandbox users that you can use for SCA and AISP and PISP operations.

USERNAME	1000401222
PASSWORD	heslo
SECOND_FACTOR_KEY	123456
PAYMENT_CONFIRMATION_KEY	123456
ACCOUNTS	CZ6179800000000216007527

State parameter in authorization request

The client application can use *state* parameter to correlate authorization request with callback redirect.

State parameter must contain character ','. Otherwise it will end with error.

Example of authorization request call:

```
https://apipsd2.wuestenrot.cz/authorize?scope=AISP&response_type=code&redirect_uri=http://callback.cz&client_id=mpep2NBcJx_yldHNDtrwMtNTaOAa&state=neco,neco
```

Authorization reply

```
http://callback.cz?code=61d4e6a8-0fec-35b5-b86e-5ce1e4db2cb3&state=neco%2Cneco
```

Technical contact

In case of any technical question please contact our ESB at psd2api@wuestenrot.cz

Payment complaints contact

For a case of payments processing complaints please contact us at psd2api@wuestenrot.cz

Information about status and planned outages

Information about status and planned outages of PSD2 API are available at https://wuestenrot.cz/aplikace-tretich-stran