De Volksbank PSD2 Sandbox

Version 8 - 8 December 2020

Change log	
V5	In this version we have made some textual changes and added new endpoints: GET Sandbox AIS GetConsent for AIS, and the endpoints for CAF (Confirmation of the Availability of Funds): POST Sandbox CAF initiateConsent and POST Sandbox CAF FundsConfirmation.
V6	For V6, the OneTimeAgended payments have been added (named *future dated payments* by the Berlin Group) as well as the payment cancellation functionality.
V8	In this version, the initiatePeriodicPayment endpoint has been added.

Introduction

This document describes the test environment that de Volksbank offers for testing its PSD2 Open Banking APIs inside a 'sandbox' environment.

The difference between Sandbox and production

The production environment that enables the PSD2 Open Banking functionality is complex. It consists of many different systems in which each system takes care of a portion of the business logic.

The Sandbox looks identical to the production environment, but instead of returning live data (or live error responses) it returns static data. This does not mean that it always returns exactly the same data for each invocation: based on the given input, in particular the consentId, a different response (including error responses) may be returned.

The Sandbox environment enables you to develop and test your application:

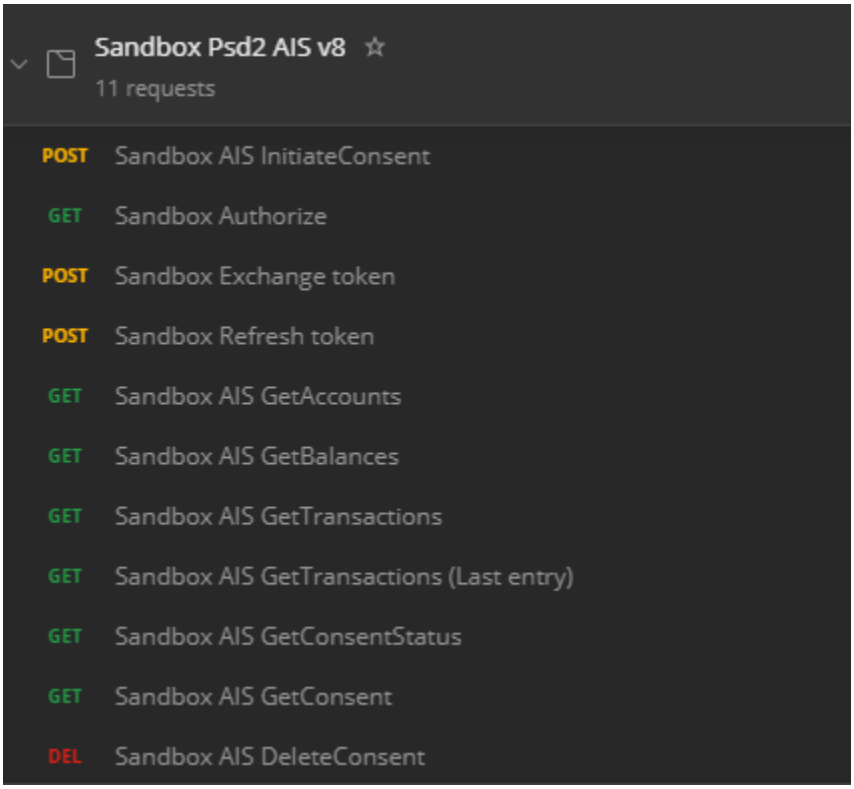
- It simulates all interactions with the Open Banking APIs of de Volksbank, similar to the production environment;
- It allows you to fully test the OAuth2 process without needing an actual de Volksbank account. This includes the interaction with our production Web Service Gateway (WSG), which requires a valid client certificate (see https://openbanking.devolksbank.nl/apis.html);
- It simulates specific error scenarios.

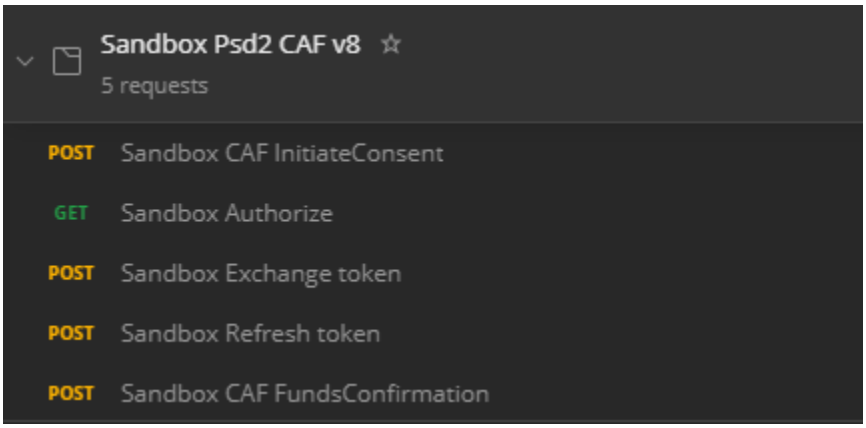
Get started

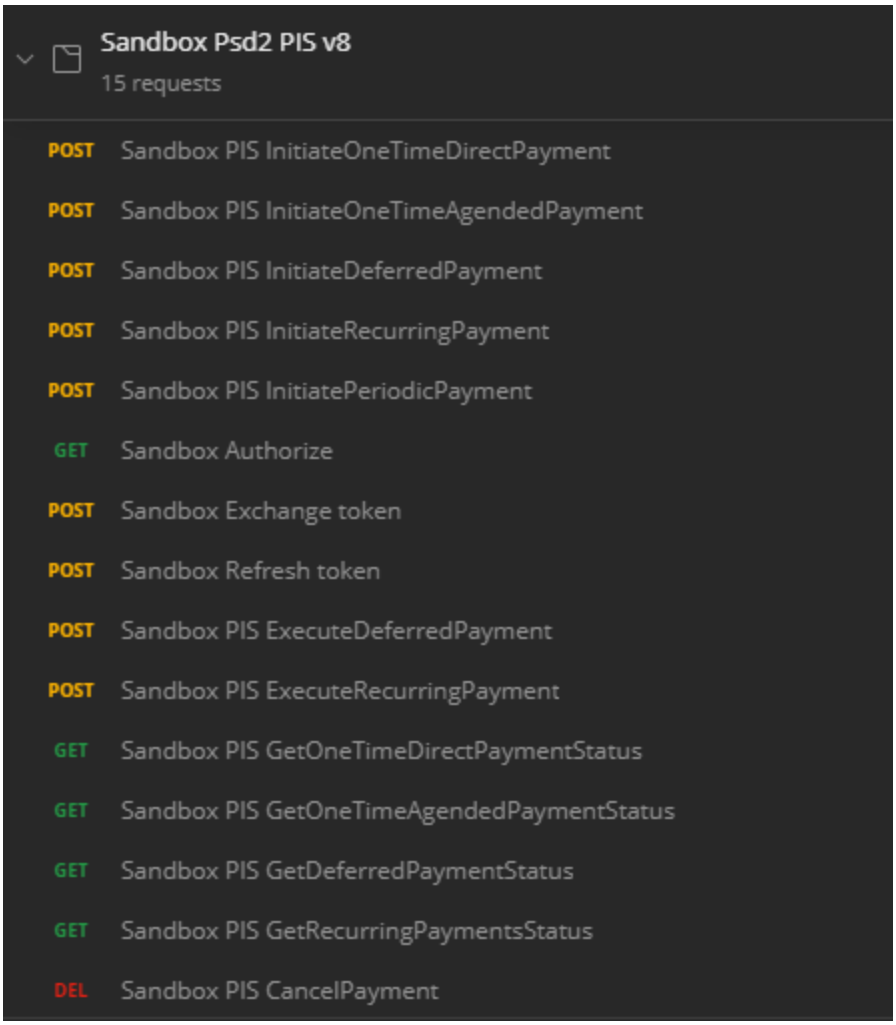
To give you a kick-start, we provided a zip file that contains three Postman collections and a Postman environment file. These can be used to test all available Authorize and PSD2 AIS, PIS and CAF APIs. The Postman files contain the correct URLs for the various endpoints and provide all the (required) header arguments and bodies (where applicable).

Inside *sandbox-devolksbank.postman_environment.json* there are a number of environment variable values that start with "*your-*". You should replace these with a valid value. In particular, the following environment variables are critical: *clientId, clientSecret, redirectUrl, accessToken, consentId, paymentId* and *resourceId.* The first three variables are fixed and are part of the initial sign up process. The latter variables are returned in the form of JSON response messages by various requests.
The zip file can be found at [https://openbanking.devolksbank.nl/documentation.html.](https://openbanking.devolksbank.nl/documentation.html)

Currently the following Postman requests are supported:







The main flows that are available in the Sandbox are explained in more detail below.

Happy flow scenario

The happy flow consists of three flows:

- A flow to initiate an AIS or CAF consent or to initiate a payment. The first API calls to retrieve an AIS, PIS or CAF resource (Sandbox Consent Services, Sandbox Payment Initiation Services and Sandbox Funds Confirmation Consent Services APIs on the Developer Portal) are:
  - Sandbox AIS InitiateConsent;
  - Sandbox PIS Initiate OneTimeDirect / OneTimeAgended / Deferred / Recurring / Periodic Payment;
  - Sandbox CAF InitiateConsent;
- A flow to retrieve an authorization code of the PSU and to exchange this code for access and refresh tokens (needed for AIS, PIS and CAF). See also the paragraph Authorization flow;
- A flow to execute AIS, PIS or CAF Services (for these calls you need a (new) access token, except for the getConsentStatus and CancelPayment calls, for which you only need a clientId):

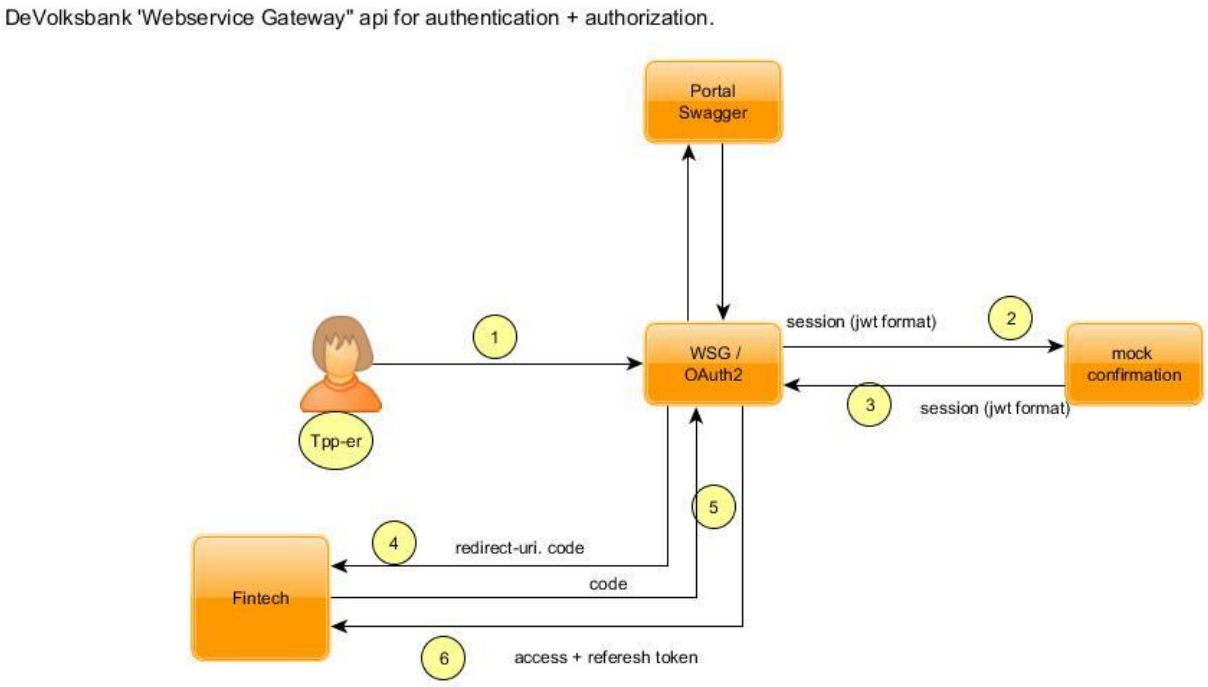
- AIS: a flow to retrieve account information (accounts, balances and transactions) and to manage an AIS consent (get status, get details and delete consent). On the Developer Portal: Sandbox Account Information Services, Sandbox Consent Status Services and Sandbox Manage Consent Services;
- PIS: a flow to execute a Deferred or Recurring Payment, to cancel a payment (only cancelling a OneTimeAgended payment is allowed) and to retrieve a payment status (for OneTimeDirect, OneTimeAgended, Deferred and Recurring Payments). On the Developer Portal: Sandbox Payment Execution Services, Sandbox Payment Cancellation Services and Sandbox Payment Status Services. Note: to get the correct response for a GetOneTimeAgendedPaymentStatus call, use the paymentId SNS1313131313021;
- CAF: a flow to request a confirmation of the availability of funds. On the Developer Portal: Sandbox Funds Confirmation Services.

The flows above are described in detail in the PDF files 'API AIS', 'API PIS' and 'API CAF' on our Open Banking website: https://openbanking.devolksbank.nl/documentation.html.

Authorization flow

Within the Sandbox, the flow as described below is used to simulate the process for authorizing a TPP to access data of a customer of de Volksbank.

Note that the authorization flow described below is not part of the Berlin Group API, and cannot be found in a Swagger file on the Developer Portal. This is a de Volksbank-specific API on its Web Service Gateway (WSG) that is used for authorization. The WSG is connected with a de Volksbank-specific OAuth2 server, which is used for authentication.



Explanation:

1. You, as TPP, initiate the authorization flow.
   See Postman request '**Sandbox Authorize'**.

2. All requests are routed through the WSG, including the request above. This particular request is forwarded to 'mock confirmation', the Sandbox implementation that mimics the behavior of the production login and consent flows. Here a screen (see note *1) is presented, with a button to grant access.

3. The 'mock confirmation' returns the sessionData in the form of a JWT token to the WSG.

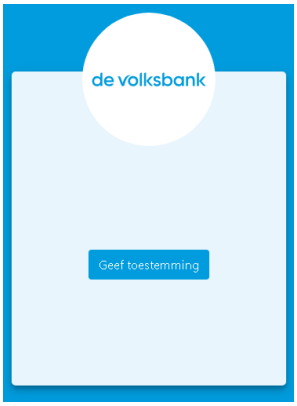
4. The WSG verifies this JWT token and redirects an authorization code to the redirect URL that you provided.

5. With this authorization code, in combination with the clientId and clientSecret that you received from de Volksbank, your app can make a request for an access token and a refresh token.
   See Postman request **'Sandbox Exchange Token'**.

6. If all data from above is valid, the WSG returns an access token and a refresh token.

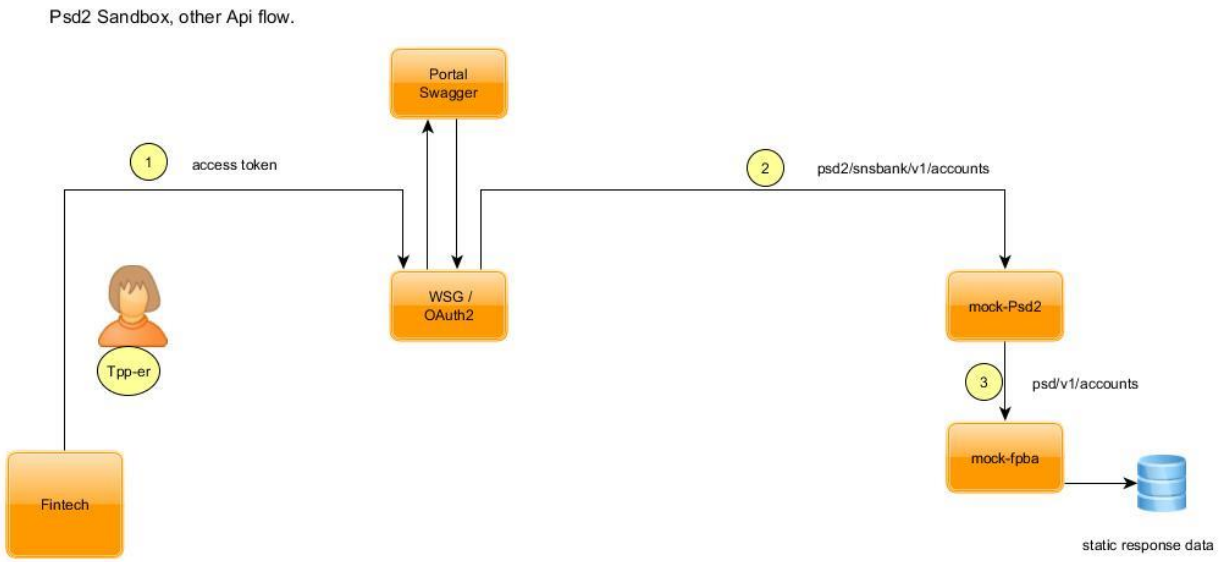
7. The access token is valid for 10 minutes. After 10 minutes this access token cannot be used anymore. With the refresh token that was returned, a new access token can be obtained.
   See Postman request **'Sandbox Refresh token'**.

Note *1: In production, de Volksbank customer is redirected to the login page, where they need to log in, select an account and finally grant access by hitting the 'Granting access' button. In the Sandbox the following screen is presented, where you only have to hit the 'Geef toestemming' button:



API flow

Once the TPP app obtains a valid access token, it can execute the API REST calls as described in the Sandbox Swagger files on the Developer Portal. For all these calls the following flow is used:



Explanation:

1. With the access token obtained in the previous flow, the TPP app can execute other operations from the de Volksbank PSD2 API.
   See for example Postman requests **'Sandbox AIS GetAccounts'** or **'Sandbox PIS Execute(Deferred or Recurring) Payment'**.

2. The request above is forwarded via the WSG to 'mock-Psd2'. Here validations on all input fields, including header and request parameters, take place. Note that these validations are production-like.

3. If all validations are passed the request is processed by returning static data. What static data is returned exactly depends on the input consentId, paymentId, or (in some scenarios) the resourceId.
   The paragraph 'Unhappy flow scenario' provides more details about the response that is returned.

Unhappy flow scenario

In the production environment, errors may occur even if all input is valid. These error scenarios can be simulated by using the following consentIds or paymentIds in a Postman request. This consentId or paymentId should be set in the Postman environment file.

Note that for requests that do not need a consentId or paymentId as input ('Initiate consent' and 'Initiate payment'), the resourceId can be used.

consentId	Unhappy scenario
SNS1313131313000	Mandate (resource) not found This response is returned when the consentId or paymentId is invalid.
SNS1313131313001	Mandate (consent/payment) is expired
SNS1313131313002	Mandate (consent/payment) is revoked by the PSU
SNS1313131313004	AIS GetTransactions input validation error For example endDate is before startDate.
SNS1313131313007	IBAN is not in contract
SNS1313131313019	Generic error from backend system.
SNS1313131313021	AIS GetConsentStatus, mandate is revoked by PSU
SNS1313131313022	AIS GetConsentStatus, mandate is expired
SNS1313131313023	AIS GetConsentStatus, mandate is terminated by TPP
SNS1313131313030	Resource not found This response is returned when the resourceId is invalid.

Testing the APIs

The easiest way to test the APIs is via the Postman collections and environment file that we prepared for the Sandbox.

Note that the returned HTTP response codes and response messages for the happy flows as well as all unhappy flows come from static data. By default all available Postman requests will return a happy flow scenario. By providing a special consentId or paymentId, different error scenarios can be simulated. The consentIds/paymentIds that correspond with the error scenarios can be found in the paragraph 'Unhappy flow scenario'.

In addition, the Sandbox tries to simulate production-like behavior (to some extent). For example, the validation on input request and/or header parameters is production-like. Because of this you may get an error response in some use-cases, if the input validation fails. For example, if you provide an invalid consentId/paymentId you will get a HTTP 401 response with the response message that this mandate cannot be found.

Also the AIS GetTransactions flow is simulated in such a way that the returned responses depend on the input: a (fixed) number of transactions or a response indicating that no more transactions are available can be returned.
See Postman request **'Sandbox AIS GetTransactions (Last entry)'**.
Postman request **'Sandbox GetTransactions'** looks like this:

```
{{host}}/psd2/sandbox/v1/accounts/SNS7642002867101/transactions
?bookingStatus=booked&dateTo=2018-10-31
```

If you replace for example: bookingStatus=booked with bookingStatus=XXX you will get a production-like response: HTTP response code = 400: Bad Request.