Raize PSD2 API

Developer Documentation

Overview

The purpose of this document is to guide developers in integrating their applications and services with the PSD2 API provided by Raize. This API consists of a set of RESTful HTTP services which allow a TPP (Third Party Payment Service Provider) to access account information and perform payment operations on behalf of Raize customers.

An OAuth2 authorization flow is used to obtain consent from a customer and allow access to their data. Multiple entities are involved including:

- **Client** – application or service provided by the TPP which will invoke the API. All clients must be registered with Raize before use.
- **User (Resource Owner)** – Raize customer using the client provided by the TPP.
- **Authorization Server** – handles the authentication of Raize customers and authorization requests from clients. Issues temporary tokens for use in invoking the API.
- **API (Resource Server)** – the PSD2 API responsible for providing account information and payment operations using the tokens issued by the authorization server.

A TPP wishing to use the API must first register a client and obtain a set of API credentials. Once a client has been registered, the TPP can obtain user consent and a token to access the API through an OAuth2 authorization flow. Finally this token can be used to call a set of API endpoints to retrieve account information and perform operations on behalf of the user. Each of these steps will be described in greater detail in the following sections.

This document will be updated as the structure of the API and corresponding functionality evolves. A summary of the changes for each version is provided below:

Version	Description	Date
1.0	Initial revision	14/03/2019

RAIZE

Registration

Any TPP authorized under the PSD2 regulation may request access to the API for a client by providing proof of registration with the Financial Supervision Authority of the corresponding European Union member state along with the following information:

	Use	Mandatory
Name of the application/service	Displayed to the user during the authorization flow.	Yes
Redirect URL	User will be redirected to this URL upon granting or denying access during the authorization flow.	Yes
Logo	Displayed to the user during the authorization flow.	No
Website URL	Displayed to the user during the authorization flow.	No

As a result of the registration, the TPP will be provided with:

	Description
API key	Uniquely identifies the calling application/service.
API secret	Secret shared between Raize and the TPP. **This value should be kept safe as it can be used to maliciously impersonate the TPP. It should be used exclusively for backend communication and should not be embedded in the code of any application running on the user's device.**

This information will be required during the authorization flow to identify the client.

Environments

A sandbox environment is provided to help test the integration with the API and corresponding OAuth2 authorization flow. This environment contains dummy data and can be used by clients during development or to test changes before deploying them to production.

When switching between environments, the following configuration changes should be performed:

	Sandbox	Production
Website	https://test.raize.pt	https://www.raize.pt
Authorization API	https://api-test.raize.pt	https://api.raize.pt
PSD2 API	https://api-sandbox.psd2.raize-ip.pt	https://api.psd2.raize-ip.pt

Scopes

Operations provided by the API can be grouped into scopes. When requesting authorization, the client must specify which scopes it requires access to. The scopes which can be requested by a given client are restricted by the role of the corresponding TPP (i.e., whether it is a AISP (Account Information Service Provider), PISP (Payment Initiation Service Provider) or Payment Instrument Issuing Service Provider (PIISP)).

The scopes which are currently available include:

Scope	Description	TPP roles
account	Allows access to account information including balances and executed transactions.	AISP

Authentication and authorization

To obtain consent from the user as well as a valid token to access the API, the client should implement the various steps of the authorization grant type OAuth2 flow as described below. If the using the sandbox environment, the URLs should be changed accordingly.

1. Redirect the user to the authorization page

To obtain the user's consent, redirect the user to the Raize website by constructing a URL such as:

`https://www.raize.pt/authorize?response_type=code&client_id=<CLIENT_ID>&redirect_uri=<REDIRECT_URI>&scope=<SCOPE>&state=<STATE>`

	Description	Madatory
CLIENT_ID	The API key received as a result of client registration.	Yes
REDIRECT_URI	The redirect URL provided during client registration. The user will be redirected back to this URL when allowing or denying the client request.	Yes
SCOPE	One or more scopes to which the client is requesting access to. In case of multiple scopes, they should be separated by spaces.	Yes
STATE	Can be used by the client to store information regarding the request. This field is not used by the authorization server and will be included in the response when redirecting the user back to the redirect URL.	No

Upon redirecting to the above URL, the user will be prompted to login to their Raize account (if not already logged in) and subsequently to allow or deny access to their account.

2. Handle the authorization result

If the user chooses to allow access, they will be redirected to:

`<REDIRECT_URI>?code=<AUTHORIZATION_CODE>&state=<STATE>`

The received code will be used in the next step to obtain an access token. This code is valid for 5 minutes.

If the user denies access instead, they will be redirected to:

`<REDIRECT_URI>?error=denied&state=<STATE>`

Note that the ***STATE*** parameter will only be included if it was provided in the initial request.

3. Exchange the authorization code for an access token

To exchange the authorization code for an access token, perform an HTTP POST request to the authorization server's token exchange endpoint:

`https://api.raize.pt/oauth2/token`

The body of the request should have the following format:

`grant_type=authorization_code&code=<AUTHORIZATION_CODE>&redirect_uri=<REDIRECT_URI>&client_id=<CLIENT_ID>&client_secret=<CLIENT_SECRET>`

	Description	Madatory
AUTHORIZATION_CODE	The authorization code to be exchanged.	Yes
REDIRECT_URI	The redirect URL provided during client registration.	Yes
CLIENT_ID	The API key received as a result of client registration.	Yes
CLIENT_SECRET	The API secret received as a result of client registration.	Yes

Each parameter should be a URL encoded string and the content type of the request should be `application/x-www-form-urlencoded`.

If the authorization code was successfully exchanged, the JSON response body will contain the access token:

```
{
  "access_token": <ACCESS_TOKEN>,
  "token_type": "Bearer"
}
```

Errors which can occur when calling the token exchange endpoint include:

HTTP status code	Response body	Description
400	Empty	The request body is missing parameters or is incorrectly formatted.

403	JSON response:
	
	{
	"error": *<ERROR_CODE>*
	}
		The authorization code could not be exchanged. The **ERROR_CODE** included in the response body can have one of the following values:
		
		• **INVALID_AUTHORIZATION_CODE** – the authorization code provided does not exist, has expired or has already been used;
		• **INVALID_CLIENT** – the API key and/or secret are invalid;
		• **INVALID_REQUEST_URI** – the request URL does not match the one registered to the client.

The access token is valid for 90 days and can be used to call the PSD2 API. When the token expires, the above process should be repeated to obtain a new access token.

Calling the API

The PSD2 API consists of a set of RESTful HTTP services. When calling the API, HTTP request bodies must be in JSON format (with the content type `application/json`), if present, and clients must include the following request headers:

HTTP header name	Header value
Authorization	`Bearer` ***<ACCESS_TOKEN>*** Includes the access token retrieved from the OAuth2 flow. When using the sandbox environment, an access token with the value ***dummy*** may be provided instead.
X-Request-ID	A string generated by the client which should uniquely identify the request. No validation is performed on the value but the use of a UUID is recommended.

Responses use standard HTTP status codes to indicate the result of the request:

HTTP status code	Description
200	The request was successful and includes a response body in JSON format.
204	The request was successful, but the response body is empty.
400	The request headers, body or query parameters are incorrectly formatted, or a mandatory parameter is missing.
401	Authentication is required but the access token was not provided.
403	The provided access token does not exist, has expired or does not grant access to the requested resource's scope.
404	The requested resource was not found.

Account API

A set of endpoints is provided to obtain information regarding a given customer's account:

Account summary	
Endpoint	`https://api.psd2.raize-ip.pt/v1/account`
Scope	account
Description	Used to retrieve a summary of the customer's account including current balance.

Request parameters	None		
Response	JSON dictionary with the following fields:		
	Field	**Type**	**Description**
	balance	string / decimal	The customer's current account balance in euros.

Example response	<pre>{
	  "balance": "132.16"
	}</pre>		

Account transactions			
Endpoint	`https://api.psd2.raize-ip.pt/v1/account/transactions`		
Scope	account		
Description	Used to retrieve a list of the customer's account transactions.		
Request parameters	None		
Response	JSON dictionary with the following fields:		
	Field	**Type**	**Description**
	transactions	array of *Transaction*	A list of the customer's transactions.
	Where custom types are defined as:		
	Transaction		
	Field	**Type**	**Description**
	date	string / date	The date the transaction was created.
	category	string	The classification of the transaction. Possible values: • *deposit* • *withdrawal* • *loanPayment* • *loanBought* • *loanSold* • *loanRecovered* • *loanInvestment* • *bonus*
	operation	string	The sign of the transaction. Possible values: • *debit* • *credit*
	amount	string / decimal	The amount the transaction refers to in euros.

Example response	<pre>{
	  "transactions": [
	  {
	  "date": "2018-12-10T09:10:11Z",
	  "category": "deposit",
	  "operation": "credit",
	  "amount": "1000.00"
	  },
	  {
	  "date": "2019-02-05T16:39:45Z",
	  "category": "withdrawal",
	  "operation": "debit",
	  "amount": "1000.00"
	  }
	  ]
	}</pre>		