Sommario

1. Introduction

1.1. History

DATE	DESCRIPTION
10/12/2021 – V 1.0	First Release
01/07/2022 – V 1.1	Add some required data in Payment data: debtorAccount and requestExecutionDate

2. Overview

2.1. Scope of documents

This document provides the TPPs the information to access to API Interface Agreement offered by Mooney to use the AISP and PISP services for Merchants payments accounts (https://www.mymooney.it/).

To offer the most beneficial information to the TPPs this document may receive further updates at future dates.

The API are based on last Berlin Group framework (https://www.berlin-group.org/nextgenpsd2-downloads) with some limitation described in this document.

2.2. API Swagger

API	Description	Url Open API Specifications
AISP / PISP	API for the following services: Payment Initiation Service (PIS) Account Information Service (AIS)	Attachment: mooney-psd2-compliance-v1.yaml Specification Download: https://www.mooney.it/documents/10549/18545781/mooney-psd2-compliance-v1.yaml.pdf/398d3389-e685-44ad-8893-322e5406ea77 Note: please rename the file after download removing .pdf filename extension
Fallback	API to obtain the token for fallback scenario to access to the home banking using screen scraping approach	Attachment: mooney-psd2-fallback-v1.yaml Specification Download: https://www.mooney.it/documents/10549/18545781/mooney-psd2-fallback-v1.yaml.pdf/94e43118-92ae-4722-a15a-d71f7d63cee5 Note: please rename the file after download removing .pdf filename extension

2.3. TPP Onboarding

The TPP (Third party provider) can access to the API directly using an eIDAS (QWAC) certificate compliant to PSD2 and issued by a valid CA (Certification Authority)

2.4. Enviroments

Sandox (AIS PIS)

- Base Url: https://psd2.mooney.it/api/psd2-sandbox/v1

Production (AIS PIS)

- Base Url: https://psd2.mooney.it/api/psd2/v1/

Production (Fallback)

- Base Url: https://psd2.mooney.it/api/psd2-fallback/v1/

2.5. Access to API

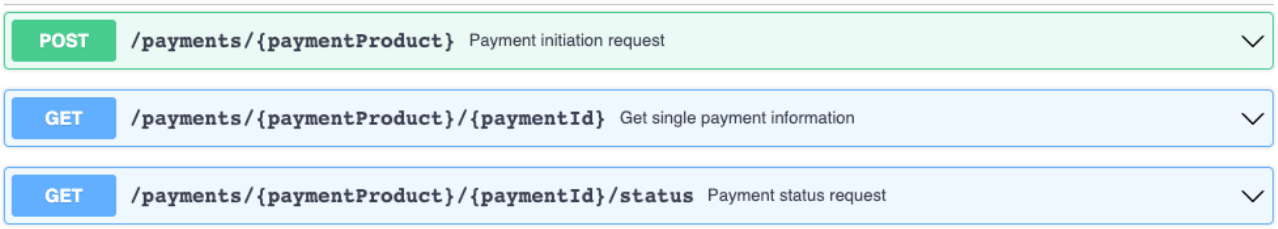
The API are available for all TPP that are using an eIDAS certificate (QWAC).
Any API must be signed using QSEAL certificate.

2.6. SCA Approach

The SCA Approach enabled is REDIRECT mode.
The user will receive an OTP text via sms to confirm the access after login.

3. API - Payment Initiation Service (PIS)

The following API are available. Download and use the open api specification mooney-psd2-compliance-v1.yaml to get all details about headers, body, request and response.



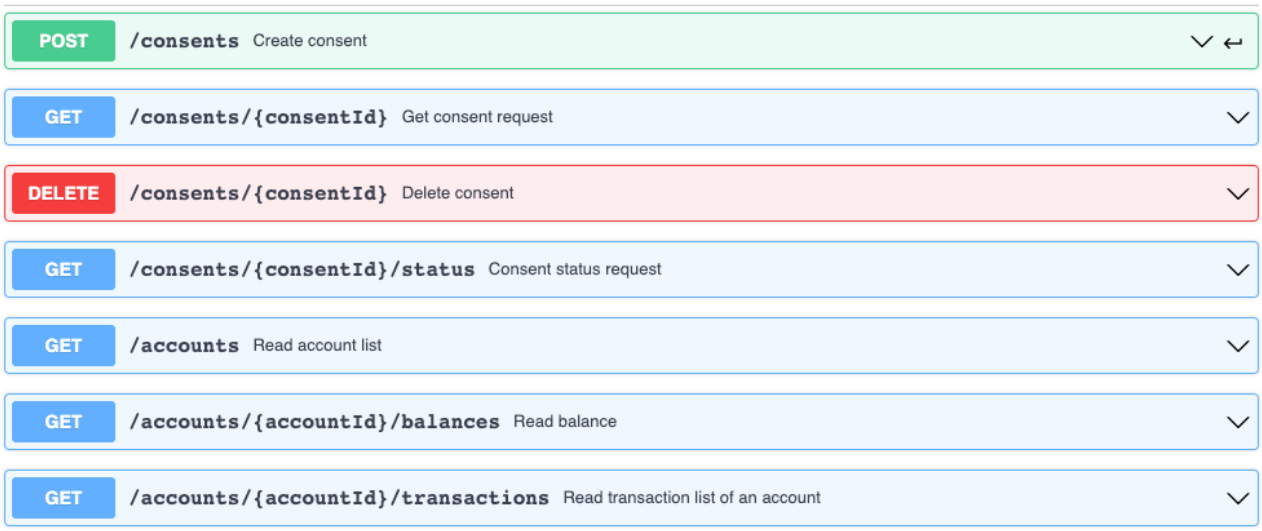
The Payment initialization (POST /payments/{paymentProduct}) has to use only with creditor data:
Example

```
{
  "instructedAmount": {
    "currency": "EUR",
    "amount": "123"
  },
  "creditorAccount": {
    "iban": "FR7612345987650123456789014"
  },
  "creditorName": "Creditor Name",
  "debtorAccount": {
    "iban": "IT7612345987650123456789014"
```

```
  },
  "requestedExecutionDate": "2022-06-22"
  "remittanceInformationUnstructured": "Ref Number Merchant"
}
```

4. API - Account Information Service (AIS)

The following API are available. Download and use the open api specification mooney-psd2-compliance-v1.yaml to get all details about headers, body, request and response.



The consent initialization (POST /consents) support only access with **allPsd2** tag with value **allAccountsWithOwnerName**
Example

```
{
"access": {
    "allPsd2": "allAccountsWithOwnerName"
},
"recurringIndicator": "true",
"validUntil": "2022-02-10",
"frequencyPerDay": 4
}
```

5. API – Sandbox dataset

For sandbox environment are available the following scenario and data set to use

VERB	PATH	SANDBOX DATA SET

POST	/payments /{paymentProduct}	The formal and certificates checks are the same of the production service. In order to test different behaviour you can specify different **Creditor Iban** The use cases for the different **Iban** are: * **IT17X0331713401000001234567**: it returns a response with * transactionStatus **RCVD** * Http Status **201** * Link to a payment with status **ACCC** * Link to a payment with cancellation status **CANC** * **IT66C0100503382000000218020**: it returns a response with * transactionStatus **RCVD** * Http Status **201** * **transationFees** populated * Link to a payment with status **ACTC** * Link to a payment with cancellation status **CANC** * **IT60X0542811101000000123456**: it returns a response with * transactionStatus **RCVD** * Http Status **201** * Link to a payment with status **RJCT** * Link to a payment with cancellation status **RJCT** * **IT12T1234512345123456789123**: it returns a response with * transactionStatus **RCVD** * Http Status **201** * Link to a payment with status **RCVD** * Link to a payment with cancellation status **RCVD** and second sca flow * **Any other values** returns an error response with **404** Http Status
GET	/payments /{paymentProduct} /{paymentId}	The formal and certificates checks are the same of the production service. In order to test different behaviour you can specify different **paymentId**. The use cases for the different **paymentId** are: * **f1d0d0af-d102-4dfe-849c-0a5c6c651bb1**: it returns a payment with status **ACCC** and an **200** Http Status * **d5306ee7-a02d-4d78-95c7-40c9da97e5fe**: it returns a payment with status **ACTC** and an **200** Http Status * **86cced5e-dc3c-4a74-a81e-a7cab15657f6**: it returns a payment with status **RJCT** and an **200** Http Status * **e94fb4bc-6f5c-41f0-9502-fe0316452d51** it returns a payment with status **RCVD** and an **200** Http Status * **Any other values** returns an error response with **404** Http Status

GET	/payments /{paymentProduct} /{paymentId} /status	The formal and certificates checks are the same of the production service. In order to test different behaviour you can specify different **paymentId**. The use cases for the different **paymentId** are: * **f1d0d0af-d102-4dfe-849c-0a5c6c651bb1**: it returns a payment with status **ACCC** and an **200** Http Status * **d5306ee7-a02d-4d78-95c7-40c9da97e5fe**: it returns a payment with status **ACTC** and an **200** Http Status * **86cced5e-dc3c-4a74-a81e-a7cab15657f6**: it returns a payment with status **RJCT** and an **200** Http Status * **e94fb4bc-6f5c-41f0-9502-fe0316452d51** it returns a payment with status **RCVD** and an **200** Http Status * **Any other values** returns an error response with **404** Http Status
POST	/consents	The formal and certificates checks are the same of the production service. In order to test different behaviour you can specify different **PSU-ID** header The use cases for the different **PSU-ID** are: * **6a401a51-e8b3-400b-b2f1-9d717706b084**: it returns a response with * consentStatus **received** * Http Status **201** * Link to a Consent with status **received** * **df565ef5-cd0b-487e-83f6-b5bd3dc23d9f**: it returns a response with * consentStatus **received** * Http Status **201** * Link to a Consent with status **rejected** * **fc2532d2-8363-433a-a28e-2fd68c1cb8f4**: it returns a response with * consentStatus **received** * Http Status **201** * Link to a Consent with status **valid** * **ac0dcab2-8b60-4753-b087-0ea954445382**: it returns a response with * consentStatus **received** * Http Status **201** * Link to a Consent with status **revokedByPsu** * **7e531dfa-a3fe-46fa-9120-abbeda89624d**: it returns a response with * consentStatus **received** * Http Status **201** * Link to a Consent with status **expired** * **c7c238bf-bd55-4eed-b230-10a8655586d4**: it returns a response with * consentStatus **received** * Http Status **201** * Link to a Consent with status **terminatedByTpp** * **If missing** returns one of the previous data randomly * **Any other values** returns an error response with **404** Http Status

GET	/consents /{consentId}	The formal and certificates checks are the same of the production service. In order to test different behaviour you can specify different **consentId** header The use cases for the different **consentId** are: * **6a401a51-e8b3-400b-b2f1-9d717706b084**: it returns a response with consentStatus **received** * **df565ef5-cd0b-487e-83f6-b5bd3dc23d9f**: it returns a response with consentStatus **rejected** * **fc2532d2-8363-433a-a28e-2fd68c1cb8f4**: it returns a response with consentStatus **valid** * **ac0dcab2-8b60-4753-b087-0ea954445382**: it returns a response with consentStatus **revokedByPsu** * **7e531dfa-a3fe-46fa-9120-abbeda89624d**: it returns a response with consentStatus **expired** * **c7c238bf-bd55-4eed-b230-10a8655586d4**: it returns a response with consentStatus **terminatedByTpp** * **Any other values** returns an error response with **404** Http Status
DELETE	/consents /{consentId}	The formal and certificates checks are the same of the production service. In order to test different behaviour you can specify different **consentId** header If you use one of the following **consentId** receive a positive response: * **6a401a51-e8b3-400b-b2f1-9d717706b084** * **df565ef5-cd0b-487e-83f6-b5bd3dc23d9f** * **fc2532d2-8363-433a-a28e-2fd68c1cb8f4** * **ac0dcab2-8b60-4753-b087-0ea954445382** * **7e531dfa-a3fe-46fa-9120-abbeda89624d** * **c7c238bf-bd55-4eed-b230-10a8655586d4** **Any other values** returns an error response with **404** Http Status

GET	/consents /{consentId} /status	The formal and certificates checks are the same of the production service. In order to test different behaviour you can specify different **consentId** header The use cases for the different **consentId** are: * **6a401a51-e8b3-400b-b2f1-9d717706b084**: it returns a response with consentStatus **received** * **df565ef5-cd0b-487e-83f6-b5bd3dc23d9f**: it returns a response with consentStatus **rejected** * **fc2532d2-8363-433a-a28e-2fd68c1cb8f4**: it returns a response with consentStatus **valid** * **ac0dcab2-8b60-4753-b087-0ea954445382**: it returns a response with consentStatus **revokedByPsu** * **7e531dfa-a3fe-46fa-9120-abbeda89624d**: it returns a response with consentStatus **expired** * **c7c238bf-bd55-4eed-b230-10a8655586d4**: it returns a response with consentStatus **terminatedByTpp** * **Any other values** returns an error response with **404** Http Status
GET	/accounts	In order to test different behaviour you can specify different **Consent-ID** header The use cases for the different ** Consent-ID** are: * **fc2532d2-8363-433a-a28e-2fd68c1cb8f4**: it represents a *valid* consent and it returns a response with accounts list and **200** Http Status * **7e531dfa-a3fe-46fa-9120-abbeda89624d**: it represents a *expired* consent and it returns a response with **401** Http Status an **CONSENT_EXPIRED** errore code * **df565ef5-cd0b-487e-83f6-b5bd3dc23d9f**: they represent *invalid* consents and they return a response with status **401** Http Status and **CONSENT_INVALID** error code * **e64377af-457d-49c6-ae2e-10718d224de3**: it represents a *valid* consent ant it returns a response with **429** Http Status and **ACCESS_EXEEDED** error code * **Any other values** returns an error response with **404** Http Status

GET	/accounts /{accountId} /balances	In order to test different behaviour you can specify different **Consent-ID** header The use cases for the different ** Consent-ID** are: * **fc2532d2-8363-433a-a28e-2fd68c1cb8f4**: it represents a *valid* consent and it returns a response with accounts list and **200** Http Status * **7e531dfa-a3fe-46fa-9120-abbeda89624d**: it represents a *expired* consent and it returns a response with **401** Http Status an **CONSENT_EXPIRED** errore code * **df565ef5-cd0b-487e-83f6-b5bd3dc23d9f**: they represent *invalid* consents and they return a response with status **401** Http Status and **CONSENT_INVALID** error code * **e64377af-457d-49c6-ae2e-10718d224de3**: it represents a *valid* consent ant it returns a response with **429** Http Status and **ACCESS_EXEEDED** error code * **Any other values** returns an error response with **404** Http Status If a valid Consent-ID is used ("fc2532d2-8363-433a-a28e-2fd68c1cb8f4"), than you have to use a right accountId, one on resourceIds returned into "accounts" call: * **631bed12-f9a9-423c-a870-ddab36fa5265** * **60ac4d00-f07d-448e-a943-890a5471098f** * **d6cfde5d-28f3-449c-88c6-e10f5388dbe1** * **0af911d4-cb64-4d44-a0b5-a528e945e5bc** * **Any other values** returns an error response with **404** Http Status
GET	/accounts /{accountId} /transactions	In order to test different behaviour you can specify different **Consent-ID** header The use cases for the different ** Consent-ID** are: * **fc2532d2-8363-433a-a28e-2fd68c1cb8f4**: it represents a *valid* consent and it returns a response with accounts list and **200** Http Status * **7e531dfa-a3fe-46fa-9120-abbeda89624d**: it represents a *expired* consent and it returns a response with **401** Http Status an **CONSENT_EXPIRED** errore code * **df565ef5-cd0b-487e-83f6-b5bd3dc23d9f**: they represent *invalid* consents and they return a response with status **401** Http Status and **CONSENT_INVALID** error code * **e64377af-457d-49c6-ae2e-10718d224de3**: it represents a *valid* consent ant it returns a response with **429** Http Status and **ACCESS_EXEEDED** error code * **Any other values** returns an error response with **404** Http Status If a valid Consent-ID is used ("fc2532d2-8363-433a-a28e-2fd68c1cb8f4"), than you have to use a right accountId, one on resourceIds returned into "accounts" call: * **631bed12-f9a9-423c-a870-ddab36fa5265** * **60ac4d00-f07d-448e-a943-890a5471098f** * **d6cfde5d-28f3-449c-88c6-e10f5388dbe1** * **0af911d4-cb64-4d44-a0b5-a528e945e5bc** * **Any other values** returns an error response with **404** Http Status

6. API – Fallback

By the Revised Payment Services Directive [PSD2], Account Servicing Payment Service Providers (ASPSPs), are required to grant Third Party Payment Services Providers (TPPs) - conditionally on the requirements of the PSD2 and the RTS - access to their customers' (Payment Service User's – PSU) bank accounts.

For this purpose, Mooney implements dedicated interfaces through which TPPs access the ASPSPs administration system and, thus, the PSU bank accounts. The dedicated interface allows Mooney not only to identify the TPP by certificates, but provides a secure access environment to protect PSU data. With respect to the dedicated interface's performance and availability, the EBA asks ASPSPs to monitor both and provide contingency (fallback) mechanisms in case the dedicated

interface is unavailable. Therefore, in agreement with the regulator, a fallback mechanism is temporarily made available by Mooney bank until the dedicated APIs are in place.

The proposed fallback solution consists of a secured proxy service, carrying out regulatory controls required by the EBA, before redirecting the TPP to Mooney Home Banking site. These controls correspond to:

- TLS MA 1.2 resolution (mutual authentication) with the TPP during each exchange
- Verification that the certificate's CA is an official QTSP of the EU Trusted List
- Verification that the certificate is not revoked by QTSP via Certificate Revocation List (CRL)

The Fallback session is represented by a unique identifier (called TOKEN) which is provided by the secured proxy service and which identifies the session. The TOKEN is valid for 15 minutes and the TPP must persist it on their side and provide it in the scrapping request. A Session is defined as follow:

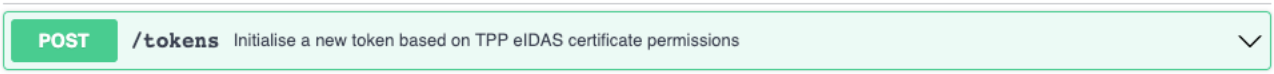
- AIS : Account consultation initialization transaction : a session is a scrapping period of maximum 15 minutes
- PIS : Payment initiation workflow – A session is defined for each payment transaction

For auditing purposes, the unique TOKEN linked to the TPP will be tracked by Mooney.
All these verification requests will be stored and tracked by the secured proxy service. Once all checks have been carried out and validated by the bank's verification service, the TPP will be redirected to the Home Banking site (using http 30X redirect) and will be able to perform "web-scrapping" on the html content as the TPP currently does.
It is the responsibility of the TPP to:

- Respect the fallback process and the RTS (including the limit of 4 AIS workflow without the PSU involvement in a period of 24h).
- Scrapp only the payment accounts for which the PSU gave his consent. The TPP is not allowed to scrapp other existing accounts of the PSU.

Mooney will be able to identify fraudulent TPP as TPP offering connection to the Bank service without being visible in the audit trail of the Fallback solution.

The following API are available. Download and use the open api specification mooney-psd2-fallback-v1.yaml to get all details about headers, body, request and response.



This API is helpfull in order to obtain a unique link to accesso to home banking in order to do the screen scraping when API are not available.

The TPP must be access to the home banking only with the link returned by the API in the field _links.location.href


```
{
  "token": "string",
  "validUntil": "2021-12-10T16:51:39.444Z",
  "_links": {
      "location": {
      "href":"https://www.mymooney.it/hb/nav/user/login?login=true&token=1234567
      890abcdefghilm"
    }
  }
}
```