
# PolishAPI

## Specification of an interface for the needs of services provided by third parties on the basis of access to payment accounts

*Document developed by the PolishAPI Project Group*

09 July 2018
**Version 2.0**

**Licence**

# Table of Contents

# Table of Figures

# 1 Introduction

## 1.1 Context

The implementation by the European Union of the new directive on payment services in the internal market (PSD2) introduces a possibility to offer new products and services related not only to the payment service market but also the financial service market in the wider sense. Both the entities present on the market, such as banks, cooperative savings and credit unions (SKOK) or branches of foreign credit institutions, as well as new types of entities (third party providers - TPP) will be able to take advantage of the possibility to offer new services built on the basis of the PSD2 Directive, the implementing acts (including the regulatory technical standards - RTS) and national acts of law. The new categories of services are:

a) **Account Information Service (AIS)** – as defined in Art. 67 of PSD2
b) **Payment Initiation Service (PIS)** – as defined in Art. 66 of PSD2
c) **Confirmation of the Availability of Funds (CAF)** – as defined in Art. 65 of PSD2

Allowing the performance of the above-mentioned services by entities authorised to do so required the preparation by account servicing payment service providers (ASPSP) of dedicated interfaces allowing access to payment accounts (the XS2A interface) by authorised third party providers (TPP), based on an open API.

Banks and other entities cooperating with the Polish Bank Association took a decision on the creation of a common and universal API standard drawing on the existing achievements of the Polish banking and payment sectors, the best practices and experiences, including those resulting from foreign API standards, as well as the existing interfaces of the interbanking infrastructure. Banks and other ASPSP will be able to implement the standard, depending on the business decisions they take independently. During the work of the business, IT, security and legal task forces, assumptions were formulated and then the standard description was created as presented herein below.

The basis assumed for this standard version was the Delegated Regulation with regard to regulatory technical standards for strong customer authentication and common and secure open standards of communication (RTS), as published in the Official Journal of the European Union on 13 March 2018 (https://eur-lex.europa.eu/legal-content/PL/TXT/?uri=uriserv:OJ.L_.2018.069.01.0023.01.POL&toc=OJ:L:2018:069:TOC).

The following entities took part in the preparation of this standard (in an alphabetical order):

1) Allegro Group
2) Biuro Informacji Kredytowej S.A.
3) Billbird S.A.
4) Blue Media S.A.
5) Diners Club Polska
6) Krajowa Izba Rozliczeniowa S.A.
7) Kontomierz.pl Sp. z o.o.
8) National Savings and Credit Union
9) Polish Association of Cooperative Banks
10) PayU S.A.
11) Polish Chamber of Information Technology and Telecommunications (PIIT)
12) Polish Insurance Association (PIU)
13) Polski Standard Płatności Sp. z o.o.
14) Polish Organisation of Non-banking Payment Institutions (PONIP)
15) Skycash Poland S.A.
16) F. Stefczyk Cooperative Savings and Credit Union

17) Polish Bank Association together with its associated bank members[1]

The specification draft was subject to public consultation (between 17-31 January 2018), in result of which 21 Polish and foreign entities submitted approx. 300 comments and observations, partially allowed for in this document.

## 1.2 Document Structure

The document consists of two fundamental parts and of annexes:

a) Part concerning the business characteristics of the PolishAPI Standard (Chapters 1 – 4)
b) Part concerning the technological solutions adopted in the PolishAPI Standard (Chapters 5 – 13)
c) Annexes, the list of which is given in Chapter 14

## 1.3 Mission of the PolishAPI Standard

The main objective of this document is to define interfaces for services described in PSD2 and related acts of law as regards the interactions between ASPSPs and TPPs during the performance of the AIS, PIS and CAF services. The requirement of open APIs also provides a chance that ASPSPs and TPPs will obtain an opportunity under a single standard to offer not only law-required services but also additional services the scope of which exceeds the framework defined by the legislator. Therefore, the following scopes of services can be identified within the PolishAPI standard:

a) **Compliance Scope** of the AIS, PIS and CAF services - services required by PSD2,
b) **Premium Scope** of the AIS, PIS and CAF services - services exceeding the requirements laid down in PSD2, outside the scope of this document.

Each ASPSP and TPP may use the PolishAPI standard as an open standard. The application of the standard is not mandatory. Each of the entities operating on the market on the basis of the PSD2 Directive may use any solution compliant with PSD2 and the related acts of law.

The interactions between the TPPs and PSUs and between ASPSPs and PSUs, as well as the matters related to the processes of making an entry in the national register of TPPs, and of granting of authorisations for the operation of TPPs in the scope related to PSD2-related services by public administration authorities are outside the scope of this document.

A part of the problems remaining within the standard specification scope will be systematically added over time as the project and agreement work (including public consultation) will advance. The above reservation concerns, without limitation, the problems related to specific functionalities of corporate accounts.

[1] Alior Bank S.A., Bank BGŻ BNP Paribas S.A., Bank Handlowy w Warszawie S.A., Bank Millennium S.A., Bank Pekao S.A., Bank Pocztowy S.A., Bank Polskiej Spółdzielczości S.A., Bank Zachodni WBK S.A., Credit Agricole Bank Polska S.A., Deutsche Bank Polska S.A., DNB Bank Polska S.A., Eurobank S.A., Getin Noble Bank S.A., Idea Bank S.A., ING Bank Śląski S.A., mBank S.A., Nest Bank S.A., PKO Bank Polski S.A., Raiffeisen Bank Polska S.A., SGB-Bank S.A.

## 1.4 Main Assumptions

### 1.4.1 Actors in the PolishAPI Standard-defined Processes

The standard defined only three categories of actors that can take part in processes defined by the PolishAPI standard:

a) **Payment Service User (PSU)** – User of the payment account the given payment transaction refers to
b) **Account Servicing Payment Service Provider (ASPSP)** – Provides maintaining the payment account and making the XS2A interface available for TPPs
c) **Third Party Provider (TPP)** – Entity using the XS2A interface on the basis of and in accordance with the consents granted by the PSUs. The ASPSP may also act as a TPP and use the interfaces made available by other ASPSPs

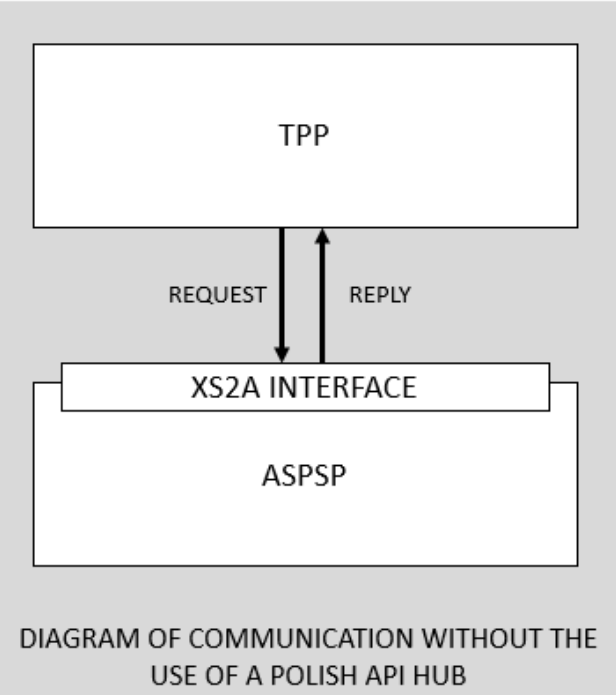



***Figure 1: General diagram of PolishAPI Standard communication***

The standard defined three roles which the actors taking part in the PolishAPI standard-defined processes can play. The categorisation below does not restrict the entities acting as TPPs to apply for an entry in the national register in more than a single role but aims at defining the roles of particular actors in the description of communication under the PolishAPI standard.

a) **Account Information Service Provider (AISP)** – TPPs using the XS2A interface to access information about the PSU's payment account.
b) **Payment Initiation Service Provider (PISP)** – TPPs using the XS2A interface to initiate the a payment transaction debited to the PSU's account.
c) **Payment Instrument Issuer Service Provider (PIISP**) – TPPs using the XS2A interface to confirm the availability at the PSU's payment account of an amount necessary to effect the payment transaction performed on the basis of an instrument issued by the PIISP.

The actors may play the following roles:

| Role \ Actor | PSU | ASPSP | TPP |
|---|---|---|---|
| **AISP** | NO | YES | YES |
| **PISP** | NO | YES | YES |
| **PIISP** | NO | YES | YES |

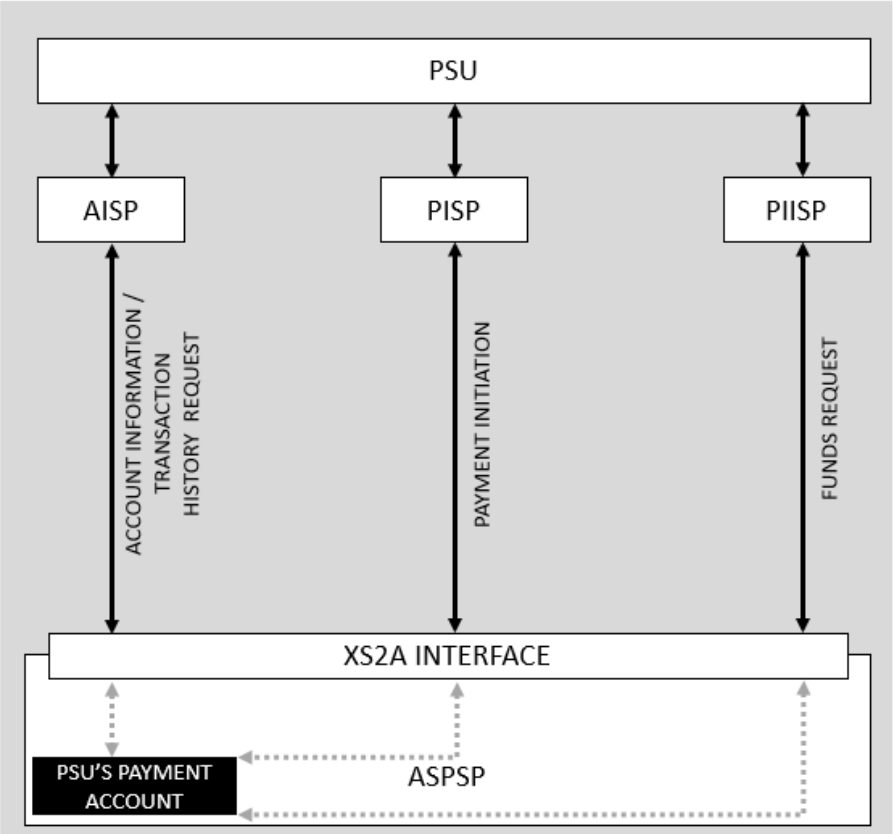



***Figure 2: General diagram of dependencies between actors in the PolishAPI Standard***

### 1.4.2 Requirements concerning Actors in the PolishAPI Standard-defined Processes

a) The ASPSP must implement an XS2A interface compliant with the PolishAPI standard. The ASPSP may also implement other XS2A interface standards, which however will not be covered by the scope of this document
b) The interfaces implemented by ASPSPs must be compliant with PSD2, the Payment Services Act and related acts, in particular the RTSs
c) The TPP must be registered in at least one register of the European Union Member State in the role it intends to play during the PolishAPI standard-based communication
d) The TPP and ASPSP must have a valid certificate used for mutual identification in the XS2A interface obtained from a qualified provider of trust services and meeting the regulatory requirements concerning the electronic identification and trust services. The certificate should

additionally meet the requirements defined in the RTSs and in the ETSI technical specification (TS 119 495).

e) PSU may be present in the context of an account for individual clients and in the context of an account for corporate (business) clients. The context of an account for an individual client is the default one. For calls in the context of an account for a corporate client, there must be the 'isCorporateContext' tag sent with the value of '*true*' in the body of the request sent as per the technical specification. Additional parameters allowing the narrowing down of the business scope of information returned by the XS2A interface are the following:
    - psuIdentifierType – type of PSU's identifier (the available range of identifiers may be different for each ASPSP and will be defined by it in a detailed XS2A interface specification; the PolishAPI specification defines a complete list of identifier types available: N - NIP, P - PESEL, R - REGON, 1 – ID card number, 2 – Passport number, 3 - Other),
    - psuIdentifierValue – value of PSU's identifier.

The parameters indicated are used to indicate a more detailed context of the XS2A interface method calling, i.e. the account holder's identifier - both an individual and a corporate one. The parameters may be used on case of PSU, who is at the same time a proxy to accounts of many clients with the given ASPSP.

### 1.4.3 PSU Authentication Mechanisms

The PolishAPI Standard allows the PSU authentication mechanisms as listed below. The ASPSP may freely select the authentication method. The selection made should be compliant with the regulations in force.

#### 1.4.3.1 Authentication Mechanism on the ASPSP's Side

The PolishAPI Standard allows the use of a mechanism of ASPSP-side authentication, which assumes a redirection to the ASPSP's website during the execution of the AIS, PIS and CAF services. This means that the PSU's authentication and authorisation data are given exclusively at the ASPSP's website. The PSU is authenticated in the ASPSP's interface.

#### 1.4.3.2 Authentication Mechanism in an External Authorisation Tool (Decoupled)

The PolishAPI Standard allows the use of an authentication mechanism using an external authorization tool during the performance of the AIS and PIS services. The mechanism of authentication in an external authorisation tool has been presented at a high level in the diagram below. Details concerning its use have been described in chapter 7.3.

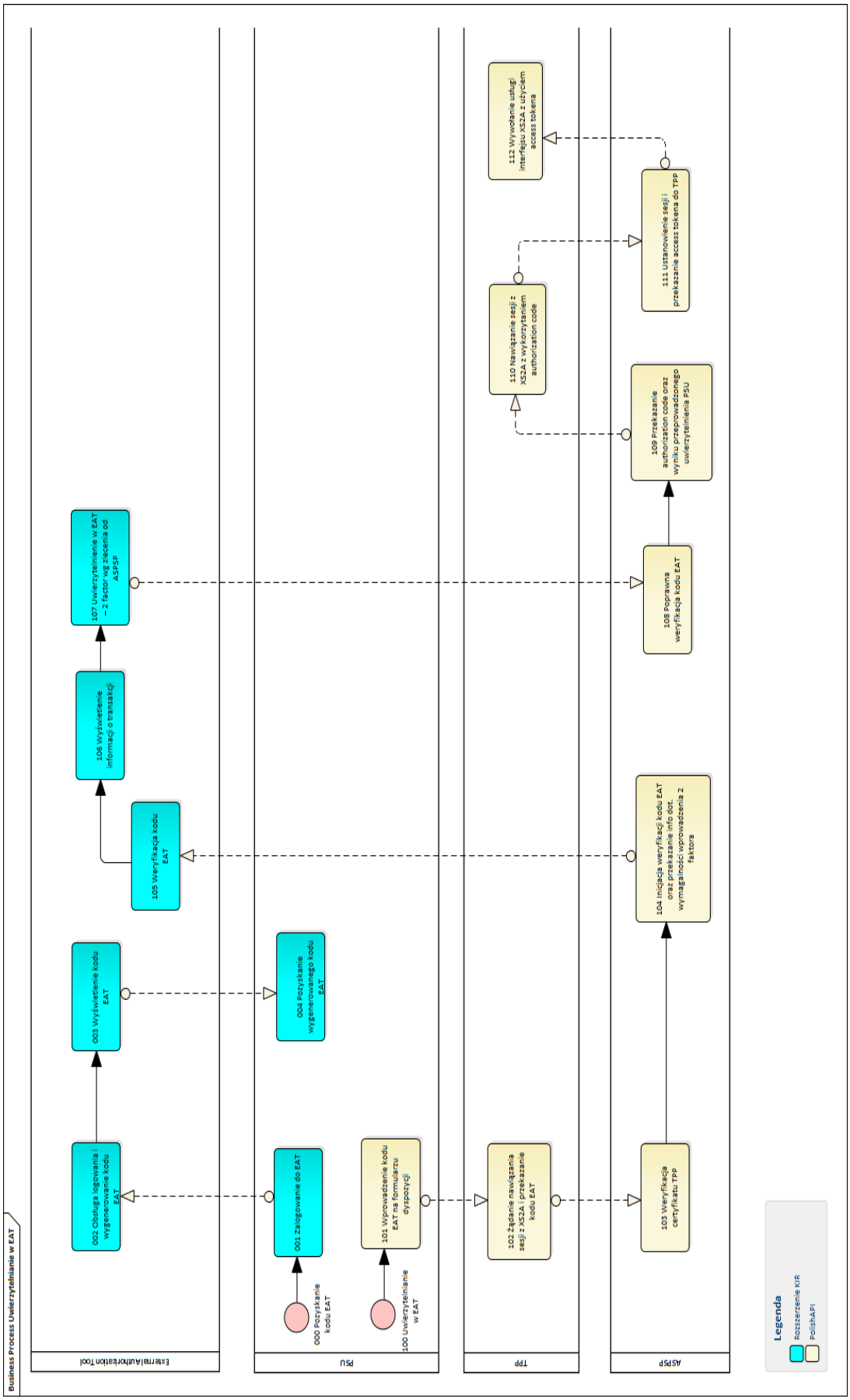



***Figure 3: Authentication in an external authorization tool***

The following assumptions have been made:

- ASPSP cooperates with the provider of an external authorization tool (hereinafter referred to as EAT).
- PSU holds an account with EAT and has taken steps necessary to use the code generator function.
- ASPSP prepares a message containing basic information about the transaction displayed in EAT before confirmation.

#### 1.4.3.2.1 Code acquisition from EAT:

- 001 / In order to obtain an EAT code, PSU logs into a dedicated tool compliant with PSD2 and with ASPSP's safety requirements.
- 002 / EAT supports PSU's logging in and generates an EAT code.
- 003 / EAT displays the code to PSU.
- 004 / PSU acquires the EAT code generated.

#### 1.4.3.2.2 PSU's authentication using EAT:

- 101 / PSU inserts the EAT code in the instruction form at TPP's.
- 102 / TPP forwards the request to start a session with XS2A and to send the EAT code to ASPSP.
- 103 / ASPSP verifies TPP, there is, inter alia, a verification of TPP's certificate.
- 104 / ASPSP initiates the EAT code verification and provides information concerning the use of a 2nd factor by EAT and basic information about the transaction.
- 105 / EAT verifies the code.
- 106 / EAT displays to the user the information about the transaction in process (agreed details).
- 107 / EAT performs authentication – the 2nd factor is used as per ASPSP's request.
- 108 / EAT sends information to ASPSP about a correctly identified EAT code.
- 109 / ASPSP provides the *authorization code* and the result of PSU's authentication to TPP.
- 110 / TPP starts a session with XS2A using the *authorization code*.
- 111 / ASPSP establishes a session and provides an *access token* to TPP.
- 112 / TPP recalls the XS2A interface service using the *access token*.

### 1.4.3.3 Other Authentication Mechanisms

The standard may contain a description of other mechanisms of authentication which meet the regulatory requirements and requirements agreed during the work of the project group. They will be published in subsequent versions of this document.

## 1.4.4 Management of PSU's Consents for the Performance of Services by a TPP

Pursuant to PSD2, the TPP may perform services for a PSU only upon his/her consent and within the scope covered by such consent. The PolishAPI standard defines the framework of consent grant and revocation by PSUs.

### 1.4.4.1 Process of Granting Consent by PSU to Effect the PIS Service

It is assumed that the payment initialization process performance will be in each case related to the grant of consent by the PSU within the framework of the TPP interface. The consent granting process

described in chapter 1.4.4.1.2 is an optional process and its implementation depends on ASPSP's decision.

#### 1.4.4.1.1 Option in case the authentication mechanism on the ASPSP's side is used

- 001 / PSU initiates the process in the TPP's interface
- 002 / TPP displays the list of ASPSPs
- 003 / PSU selects an ASPSP from the list and grants a consent to TPP to provide the PIS service
- 004 / PSU completes the transfer form, which should contain at least the information indicated in Chapter 3.1.3 of this specification: 'List of Fields Required by the ASPSP in the Compliance Scope'
- 005 / TPP transfers the payment initiation request to the ASPSP and a redirection is made to the ASPSP's domain in order to authenticate the PSU
- 006 / ASPSP verifies the TPP's identify on the basis of a certificate (or also on the basis of the register of TPPs)
- 007 / ASPSP sends a PSU authentication request
- 008 / Authentication
- 009 / ASPSP displays transaction details to PSU
- 010 / ASPSP generates and sends to the PSU an additional authorization element (e.g. OTP) – provided that it is required in accordance with the regulations in force
- 011 / PSU authorises the transaction using the method applied in relations with the ASPSP (the PSU has an option to refuse authorisation, which results in the fact that the payment transaction is not effected)
- 012 / ASPSP performs the request and then a redirection is made to the TPP's domain

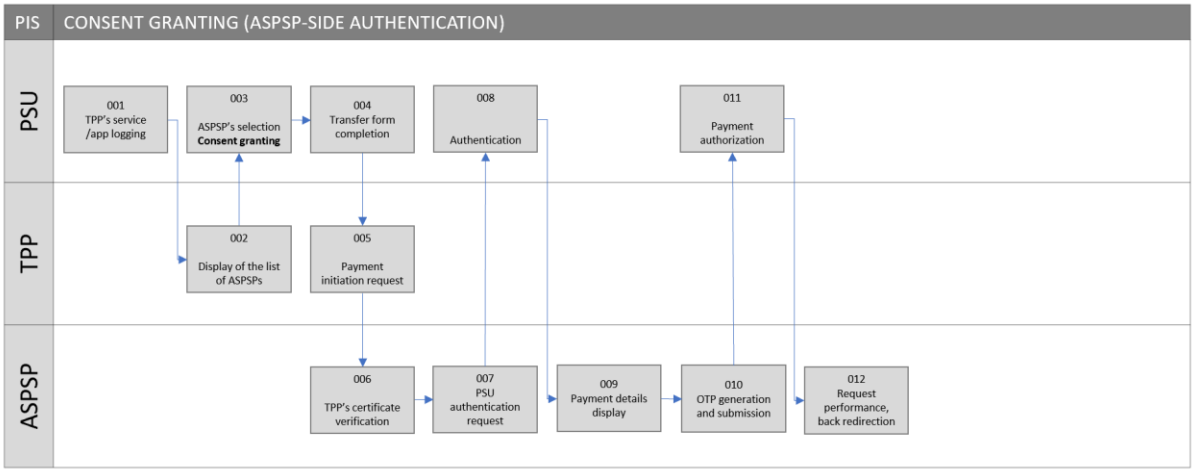



***Figure 4: PIS. Consent grant (authentication on the ASPSP's side)***

#### 1.4.4.1.2 Option in case the ASPSP-side authentication mechanism is used with the ASPSP-side account selection

- 001 / PSU initiates the process in the TPP's interface
- 002 / TPP displays the list of ASPSPs
- 003 / PSU selects an ASPSP from the list and grants a consent to TPP to provide the PIS service
- 004 / PSU completes the transfer form, which should contain at least the information indicated in Chapter 3.1.3 of this specification: 'List of Fields Required by the ASPSP in the Compliance Scope'

- 005 / TPP transfers the payment initiation request to the ASPSP and a redirection is made to the ASPSP's domain in order to authenticate the PSU
- 006 / ASPSP verifies the TPP's identify on the basis of a certificate (or also on the basis of the register of TPPs)
- 007 / ASPSP sends a PSU authentication request
- 008 / SCA authentication
- 009 / ASPSP presents the PSU in its own interface a list of payment account from which a payment transaction may be initiated to select from
- 010 / PSU selects one account from the list. and accepts the transaction
- 011 / ASPSP generates and sends to the PSU an additional authorization element (e.g. OTP) – provided that it is required in accordance with the regulations in force
- 012 / PSU authorises the transaction using the method applied in relations with the ASPSP (the PSU has an option to refuse authorisation, which results in the fact that the payment transaction is not effected)
- 013 / ASPSP performs the request and then a redirection is made to the TPP's domain

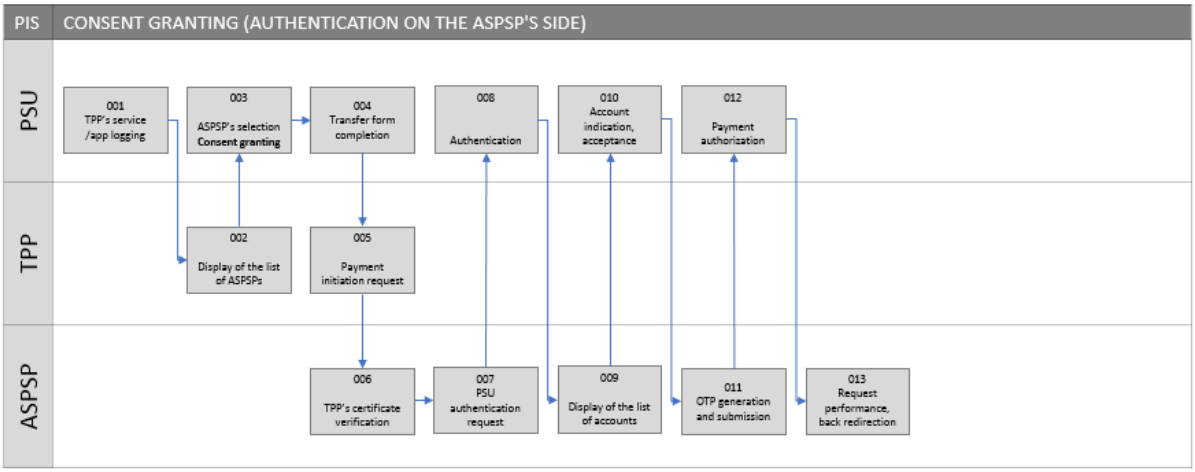



***Figure 5: PIS. Consent grant (ASPSP-side authentication), ASPSP-side selection of the account***

#### 1.4.4.1.3 Option in case an authentication mechanism in an external authorization mechanism is used

- 001 / PSU initiates the process in the TPP's interface
- 002 / TPP displays the list of ASPSPs
- 003 / PSU selects an ASPSP from the list and grants a consent to TPP to provide the PIS service
- 004 / PSU completes the transfer form, which should contain at least the information indicated in Chapter 3.1.3 of this specification: 'List of Fields Required by the ASPSP in the Compliance Scope' (specifying, without limitation, the number of account from which the payment is to be initiated)
- 005 / TPP transfers the payment initiation request to the ASPSP
- 006 / ASPSP verifies the TPP's identify on the basis of a certificate (or also on the basis of the register of TPPs)
- 007 / ASPSP initiates the process of PSU's authentication, including provides an instruction concerning the use of the 2nd authorisation element for EAT – if this is required as per the regulations in force
- 008 / TPP transfers the authentication request to the PSU
- 009 / Authentication in an external authorisation tool (cf. item 1.4.3.2)

- 010 / The external authorization tool displays the payment details and the PSU accepts the transaction
- 011 / PSU authorises the transaction (PSU has an option to refuse authorisation, which results in the fact that the payment transaction is not effected)
- 012 / ASPSP performs the request

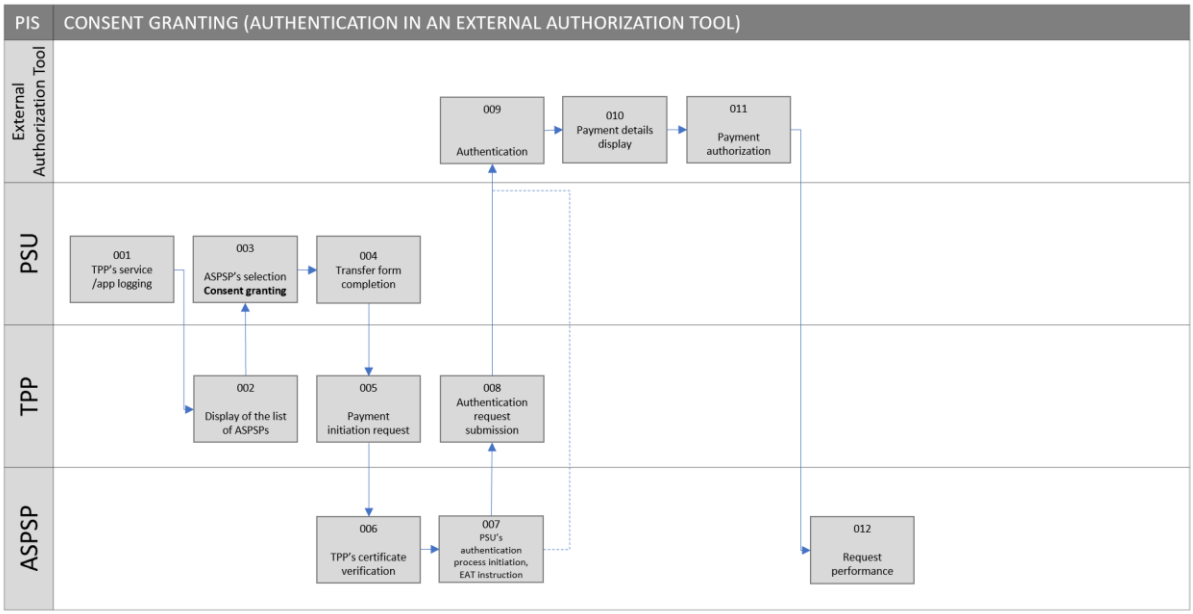



***Figure 6: PIS. Grant of consent (authentication in an external authorization tool)***

### 1.4.4.2 Process of Granting Consent by PSU to Effect the AIS Service

In this Chapter, the term 'consent' refers exclusively to the provision of the AIS service and means the grant of consent for the service without indicating specific accounts (in case of an option with a retrieval of a list of accounts) or with an indication of the same (in case of an option with a manual account number insertion). This process is always linked with the strong customer authentication (SCA).
Determination of access parameters (in case of an option with a retrieval of a list of accounts) means each operation on specific accounts within the limits of the consent to effect the AIS services, including:

- indication of a specific account
- change of parameters for a specific account (e.g. date of access)
- withdrawal of indication of a specific account or
- withdrawal of consent

These operations do not require the strong customer authentication (SCA).

The standard allows two processes of granting a consent for the AIS service (in options allowing the authentication on the ASPSP's side and in an external authorisation tool) as described in items from 1.4.4.2.1 to 1.4.4.2.4. The ASPSP may implement one process or both. The consent granting processes comprising a retrieval of a list of accounts are optional processes and their implementation depends on ASPSP's decision.

#### 1.4.4.2.1 Grant of consent with a manual insertion of the account number in case of authentication on the ASPSP's side

- 001 / PSU initiates the process in the TPP's interface

- 002 / TPP displays the list of ASPSPs
- 003 / PSU selects an ASPSP from the list and grants a consent to the given TPP to provide the account information service concerning an account maintained by the given ASPSP
- 004 / PSU inserts the account number and defines the scope of access
- 005 / TPP sends a token request to the ASPSP
- 006 / ASPSP verifies the TPP's identify on the basis of a certificate (or also on the basis of the register of TPPs)
- 007 / ASPSP sends a PSU authentication request
- 008 / SCA authentication
- 009 / ASPSP gives the TPP an access token (containing information about the scope of access), in case of AIS a multiple one with a refresh token

In case the account number or one of the many account numbers inserted is incorrect or when an access to the given account cannot be given, error message 400 will be returned.

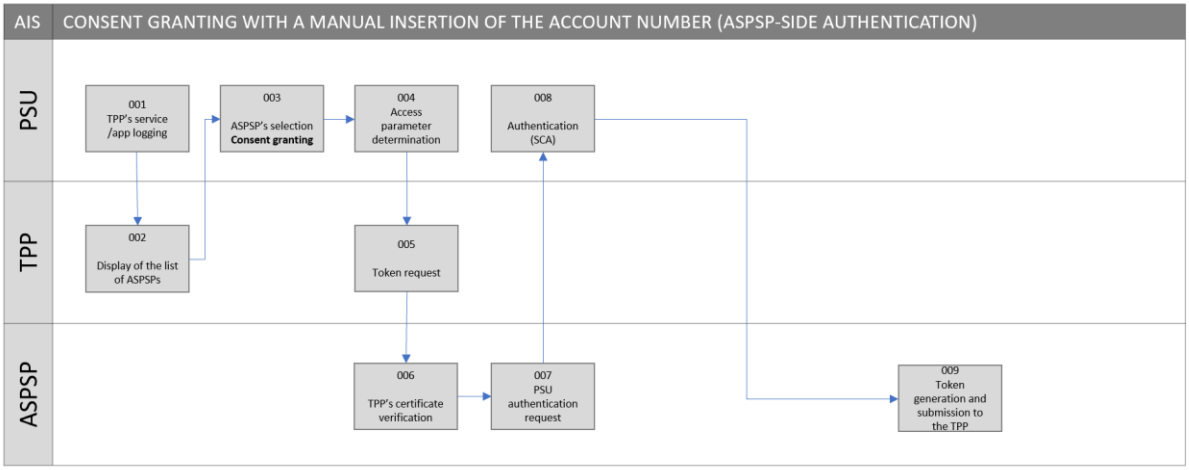



***Figure 7: AIS. Grant of consent with a manual insertion of the account number (authentication on the ASPSP's side)***

#### 1.4.4.2.2 Grant of consent with a manual insertion of the account number in case of authentication in an external authorization tool

- 001 / PSU initiates the process in the TPP's interface
- 002 / TPP displays the list of ASPSPs
- 003 / PSU selects an ASPSP from the list and grants a consent to the given TPP to provide the account information service concerning an account maintained by the given ASPSP
- 004 / PSU inserts the account number and defines the scope of access
- 005 / TPP sends a token request to the ASPSP
- 006 / ASPSP verifies the TPP's identify on the basis of a certificate (or also on the basis of the register of TPPs)
- 007 / ASPSP initiates the process of PSU's authentication
- 008 / TPP transfers the authentication request to the PSU
- 009 / SCA authentication in an external authorisation tool (cf. item 1.4.3.2)
- 010 / ASPSP gives the TPP an access token (containing information about the scope of access), in case of AIS a multiple one with a refresh token

In case the account number or one of the many account numbers inserted is incorrect or when an access to the given account cannot be given, error message 400 will be returned.

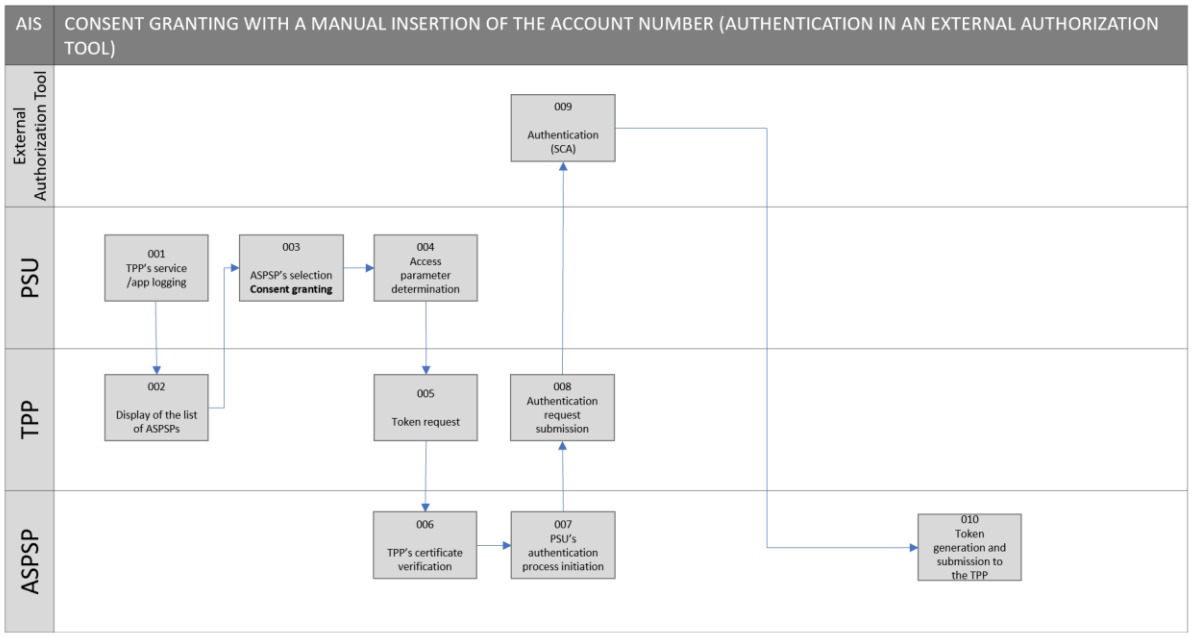



***Figure 8: AIS. Grant of consent with a manual insertion of the account number (authentication in an external authorization tool)***

#### 1.4.4.2.3 Grant of consent with an account list retrieval in case of authentication on the ASPSP's side

- 001 / PSU initiates the process in the TPP's interface
- 002 / TPP displays the list of ASPSPs
- 003 / PSU selects an ASPSP from the list and grants a consent to the given TPP to provide the account information service concerning an account maintained by the given ASPSP
- 004 / TPP sends a token request to the ASPSP
- 005 / ASPSP verifies the TPP's identify on the basis of a certificate (or also on the basis of the register of TPPs)
- 006 / ASPSP sends a PSU authentication request
- 007 / SCA authentication
- 008 / ASPSP transfers the access token to the TPP
- 009 / TPP sends a request to the ASPSP for a transfer of the PSU's account list (together with an access token)
- 010 / ASPSP gives the TPP a list of PSU's accounts (a full or a partially masked account number + product name + account type)
- 011 / TPP displays a list of accounts
- 012 / PSU indicates an account (or accounts) in order to determine the scope of access
- 013 / PSU determines the access parameters
- 014 / TPP sends a token exchange request to the ASPSP (for a token containing access details)
- 015 / ASPSP verifies the identity of the TPP and the PSU (on the basis of the first access token)
- 016 / ASPSP gives the TPP a new access token (containing information about the scope of access), in case of AIS a multiple one with a refresh token

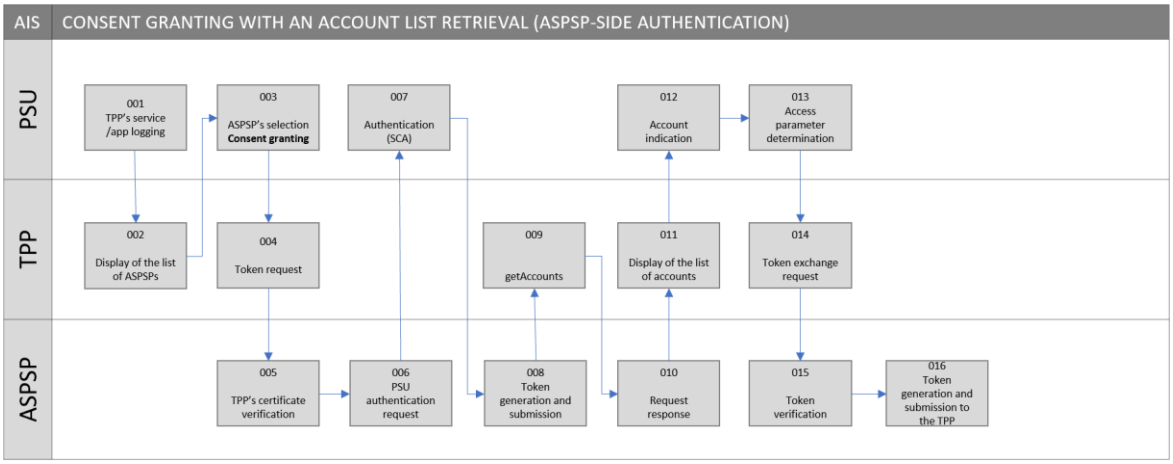



***Figure 9: AIS. Grant of consent with an account list retrieval (authentication on the ASPSP's side)***

#### 1.4.4.2.4 Grant of consent with an account list retrieval in case of authentication in an external authorization tool

- 001 / PSU initiates the process in the TPP's interface
- 002 / TPP displays the list of ASPSPs
- 003 / PSU selects an ASPSP from the list and grants a consent to the given TPP to provide the account information service concerning an account maintained by the given ASPSP
- 004 / TPP sends a token request to the ASPSP
- 005 / ASPSP verifies the TPP's identify on the basis of a certificate (or also on the basis of the register of TPPs)
- 006 / ASPSP initiates the process of PSU's authentication
- 007 / TPP transfers the authentication request to the PSU
- 008 / SCA authentication in an external authorisation tool (cf. item 1.4.3.2)
- 009 / ASPSP transfers the access token to the TPP
- 010 / TPP sends a request to the ASPSP for a transfer of the PSU's account list (together with an access token)
- 011 / ASPSP gives the TPP a list of PSU's accounts (a full or a partially masked account number + product name + account type)
- 012 / TPP displays a list of accounts
- 013 / PSU indicates an account (or accounts) in order to determine the scope of access
- 014 / PSU determines the access parameters
- 015 / TPP sends a token exchange request to the ASPSP (for a token containing access details)
- 016 / ASPSP verifies the identity of the TPP and the PSU (on the basis of the first access token)
- 017 / ASPSP gives the TPP a new access token (containing information about the scope of access), in case of AIS a multiple one with a refresh token

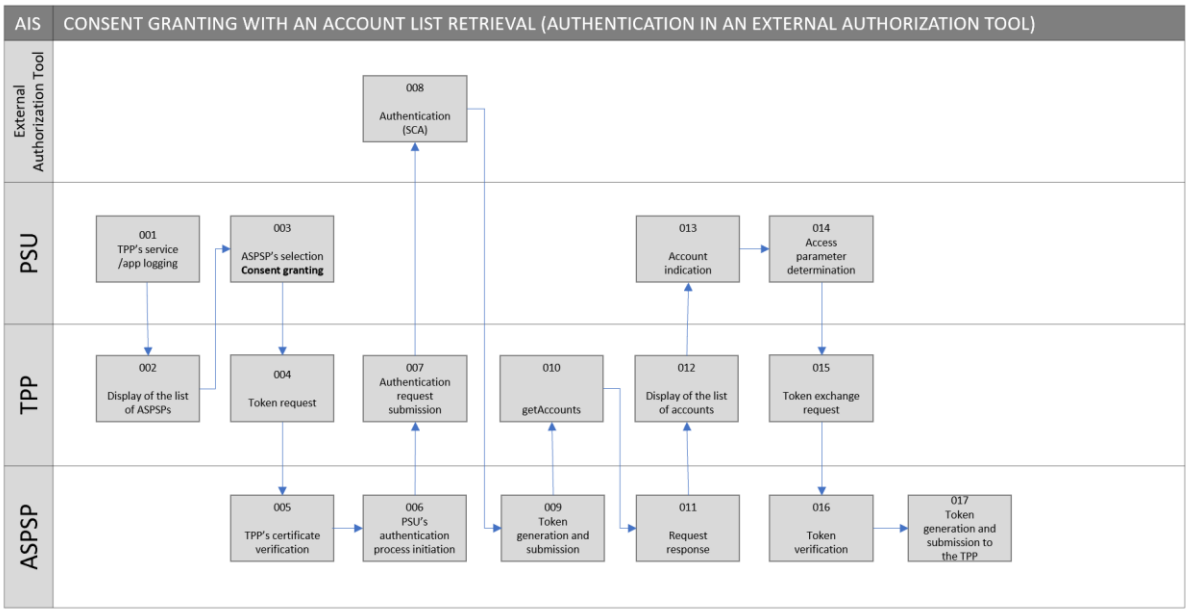



***Figure 10: AIS. Grant of consent with an account list retrieval (authentication in an external authorization tool)***

#### 1.4.4.2.5 Withdrawal of consent

- 001 / PSU initiates the process in the TPP's interface
- 002 / TPP displays a list of consents
- 003 / PSU selects a specific consent from the list of consents within the framework of which the changes will be made
- 004 / PSU withdraws the consent for the AIS service
- 005 / TPP sends a token revocation request to the ASPSP
- 006 / ASPSP verifies the TPP's identify on the basis of a certificate (or also on the basis of the register of TPPs)
- 007 / ASPSP invalidates the token related to the consent

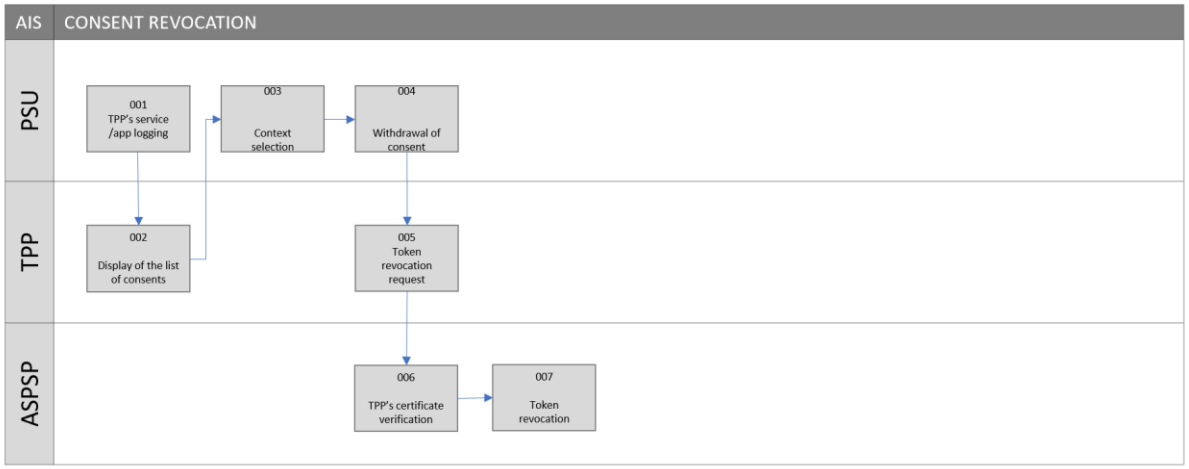



***Figure 11: AIS. Withdrawal of consent***

### 1.4.4.3 Process of Granting Consent by PSU to Effect the CAF Service

The process of the PSU's granting a consent for the ASPSP to effect the CAF service will be developed in the next version of this document. For the purposes of the current version, it is assumed that the

request within the CAF service is made exclusively in the situation when the consent has been made previously.

### 1.4.5 Application of the Strong Customer Authentication (SCA) Mechanism

ASPSPs use any given strong PSU authentication system (SCA) they selected and the PolishAPI standard does not define and does not recommend any way in which this procedure may be conducted. Furthermore, the decision to release a given transaction from the SCA procedure remains in the exclusive competence of the ASPSP.

### 1.4.6 Provision of Services within the Compliance Scope

Each ASPSP is obliged to make available the services from the Compliance Scope pursuant to PSD2 and the related acts of law. ASPSP makes the accounts available in accordance with the definition given in Chapter 3.2.1 and takes independent decisions as to the scope of payment account data available online within the framework of this service. The performance of services within the Compliance Scope will not require a contractual relation between the ASPSP and the TPP.

### 1.4.7 Provision of Services within the Premium Scope

Each ASPSP takes the decisions on making available the services within the Premium Scope and, in case of a decision to start offering them, determines the extent of such services independently. The performance of services within the Premium Scope will not require a contractual relation between the ASPSP and the TPP.

## 1.5 Development of the PolishAPI Standard

Currently, the PolishAPI standard defines the Compliance Scope of the AIS, PIS and CAF services. A permanent development of the standard in response to regulatory, technological and business changes on the Polish and European market is assumed. The changes will be published as subsequent versions of the PolishAPI standard specification.

# 2 Glossary of Terms used in the Document

**Account Information Service (AIS)** – as defined in Art. 66 of PSD2.

**Account Information Service Provider (AISP) –** TPPs using the XS2A interface to access information about the PSU's payment account.

**Confirmation of the Availability of Funds (CAF)** – a service defined in Art. 65 of PSD2.

**External Authorization Tool (EAT)** – a system providing the SCA procedure, i.e. the strong authentication of PSU.

**European Banking Authority (EBA) –** the European Banking Authority.

**ETSI –** European Telecommunication Standardisation Institute.

**OAuth2** – Oauth2 is an open authorisation standard. It allows users to share their private resources (e.g. pictures, films, contacts) stored at a given site with another party without a necessity to fathom the complexities of authorisation, usually providing the user name and a token (one-time passwords).

**Payment Initiation Service Provider (PISP)** – TPPs using the XS2A interface to initiate the a payment transaction debited to the PSU's account.

**Payment Initiation Services (PIS) –** as defined in Art. 67 of PSD2.

**Payment Instrument Issuer Service Provider (PIISP**) – TPPs using the XS2A interface to confirm the availability at the PSU's payment account of an amount necessary to effect the payment transaction performed on the basis of an instrument issued by the PIISP.

**Payment Services Directive (PSD) –** Directive 2007/64/EC of the European Parliament and of the Council on payment services in the internal market.

**Payment Services Directive 2 (PSD2) –** Directive 2015/2366 of the European Parliament and of the Council on payment services in the internal market and repealing Directive 2007/64/EC.

**Payment Services User (PSU) –** natural or legal person making use of a payment service in the capacity of either payer or payee, or both.

**Payment account –** an account held in the name of one or more payment service users which is used for the execution of payment transactions.

**Regulatory Technical Standard (RTS) –** Commission Delegated Regulation (EU) No. 2018/389 of 27 November 2017 supplementing Directive (EU) 2015/2366 of the European Parliament and of the Council with regard to regulatory technical standards for strong customer authentication and common and secure open standards of communication.

**Revised Payment Services Directive (PSD2) –** Directive (EU) 2015/2366 of the European Parliament and of the Council of 25 November 2015 on payment services in the internal market (revised payment services directive).

**Strong Customer Authentication (SCA)** - means an authentication based on the use of two or more elements (components) categorised as knowledge (something only the user knows), possession (something only the user possesses) and inherence (something the user is) that are independent, in that the breach of one does not compromise the reliability of the others, and is designed in such a way as to protect the confidentiality of the authentication data.

**Swagger –** is an open source software which helps design, build, document and consume the RESTful Web services.

**TS 119 495** – (v. 1.1.1) a technical specification of the standard concerning the qualified certificate profile for the needs of the Payment Services Directive (Electronic Signatures and Infrastructures (ESI); Sector Specific Requirements; Qualified Certificate Profiles and TSP Policy Requirements under the Payment Services Directive 2015/2366/EU), published in January 2018.

**Granting Consent** – a process in result of which the PSU grants TPP consent to access his/her account held by the ASPSP in order to effect a service, including the AIS, PIS and CAF services.

**Authentication** – a process in result of which the ASPSP verifies the PSU's identity.

**Payment Services Act –** Polish Payment Services Act of 19 August 2011.

**XS2A (Access to Account)** – access to payment accounts used to perform AIS, PIS, CAF and other services effected as part of the PolishAPI.

**Premium Scope** of the AIS, PIS and CAF services **–** services exceeding the requirements laid down in PSD2.

**Compliance Scope** of the AIS, PIS and CAF services **–** services required by PSD2.

# 3 Business Definition of the Compliance Scope Services

## 3.1 Business Definition of the Compliance Scope for the PIS Service

The Payment transaction initiation service within the Compliance Scope consists in making available by the ASPSP of a possibility to initiate a payment from the payment account by the PSU via a TPP who acts as a PISP after obtaining prior consent from the PSU as appropriate.

### 3.1.1 Types of Transactions within the Compliance Scope

As part of the PIS service within the Compliance Scope, the ASPSP will make available to the PSU, via the TPP (PISP), an initiation of payments that meet the following cumulative conditions:

a) The payment is a bank transfer
b) The payment is a single transfer
c) The payment is a transfer with the current date
d) The payment is a transfer made to an IBAN number (NRB number in case of ASPSPs operating in Poland), including a transfer to a Polish tax office
e) If it is a domestic transfer, it is settled using one of the following systems (depending on which of the systems is supported by the ASPSP):
    a. Elixir,
    b. Express Elixir,
    c. SORBNET2,
    d. Blue Cash.
f) If the payment is a foreign transfer, it is settled in one of the systems listed below:
    a. SWIFT
    b. SEPA
    c. TARGET
g) It is available in the online interface of the given ASPSP
h) The PSU will complete all the data required to order a transfer (the ASPSP will not provide support in the form of dictionaries, dropdown lists or other creators) or in case of the process described in item 1.4.4.1.2 all the data save the number of the account from which the payment will be initiated.

The data given by the TPP in the transfer order should not be modified by the PSU in the ASPSP's domain. Each ASPSP is obliged to make available the services from the Compliance Scope pursuant to PSD2 and the related acts of law. The ASPSP takes independent decisions as to the scope of payment account data available online within the framework of this service. The performance of services within the Compliance Scope will not require a contractual relation between the ASPSP and the TPP.

### 3.1.2 Information about the Transaction Status

As part of the message exchange in the PIS service within the Compliance Scope, the ASPSP will immediately advise the TPP about the order acceptance or rejection. Additionally, the TPP will be able to retrieve information about the payment status using the getPayment method with an option to enquire about the status of many payments (getMultiplePayments), provided the ASPSP offers such a functionality. The ASPSP will have an optional possibility to transfer to the TPP (asynchronous) information about the payment status using the /v1.0/accounts/v1.0/paymentCallBack method.

The following statuses are defined:

a) Submitted
b) Pending
c) Rejected

d) Done

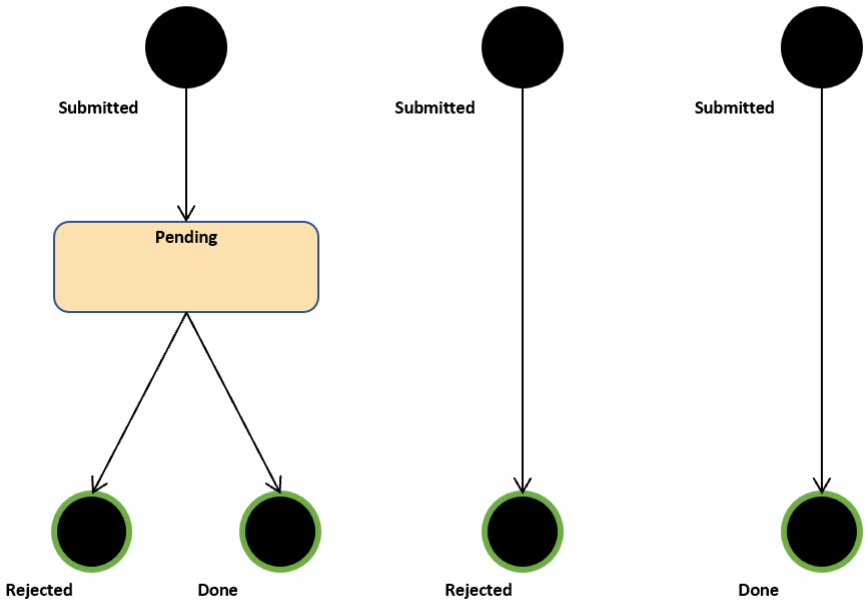



*Figure 12: Diagram of payment statuses*

### 3.1.3 Definition of a Payment Account

This service is provided only for payment accounts to which the given PSU has an on-line access. The account must meet all of the following cumulative conditions:

a) It is an account held for one or more users which is used to effect payment transactions (as per the definition laid down in the PSA).
b) The PSU has an on-line access to the account.

### 3.1.4 List of Fields Required by the ASPSP in the Compliance Scope

In order to initiate the PIS service payment transaction within the Compliance Scope correctly, the ASPSP may request from the PSU, via the TPP (PISP), that the following fields be completed with transaction order data. Each ASPSP may expect the PSU to transfer another set of data via the TPP.

With reference to foreign transfers, the use of certain fields will be optional, depending on the functionalities supported by the given ASPSPs. The ASPSP will effect the payments on condition that the payment will be initiated as appropriate by the TPP, i.e. appropriate fields with appropriate content are submitted.

Mandatory fields defining the TPP:

a) TPP's name

#### 3.1.4.1 National transfer

| FIELD NAME | REQUIRED | COMMENTS |
|---|---|---|
| Address of the transfer payee | No | |
| Effective date of the transfer | No | For the future effective date of the transfer, the urgency mode refers to that date |
| Transfer amount | Yes | |
| Name of the transfer sender | No | Sender's name completed by the ASPSP in order to avoid a situation when the transfer order coming from the ASPSP contains data other than data of the holder of the account debited. |
| Name of the transfer payee | Yes | |
| Transfer sender’s account number | Yes | It may be specified by the TPP, however the PSU should have an option to select the account to be debited after redirection to the ASPSP. |
| Transfer payee’s account number | Yes | |
| Transfer description field | Yes | |
| Urgency mode | Yes | ExpressD0, StandardD1 |
| Transfer type (system) | No | In case of a domestic transfer: Elixir, ExpressElixir, Sorbnet, BlueCash, Internal |
| Currency | No | In case the field is empty, the ASPSP will make the transfer in the account currency. |
| Block | No | Bool type field, owing to which the client will be able to make an explicit wish to make a block (in case of, e.g. a payment order on a free day). A default behaviour in case the parameter is not provided is defined by ASPSP. |
| Transaction ID as assigned by the TPP | Yes | |

#### 3.1.4.2 Domestic transfers to tax authorities / customs authorities in Poland

| FIELD NAME | REQUIRED | COMMENTS |
|---|---|---|
| Address of the transfer payee | Yes | |
| Data of the authorities | | |
| Effective date of the transfer | No | For the future effective date of the transfer, the urgency mode refers to that date |
| Payer's ID | Yes | |
| Payer’s ID type | Yes | Dictionary:<br><br>N - NIP, P - PESEL, R - REGON, 1 – ID card number, 2 – Passport number, 3 - Other |
| Liability ID | No | |
| Transfer amount | Yes | |
| Name of payer | No | Sender's name completed by the ASPSP in order to avoid a situation when the transfer order coming from the ASPSP contains data other than data of the holder of the account debited. |
| Period number | Yes | |

| | | |
|---|---|---|
| Transfer sender's account number | Yes | It may be specified by the TPP, however the PSU should have an option to select the account to be debited after redirection to the ASPSP. |
| Transfer payee's account number | Yes | |
| Form symbol | Yes | |
| Period type | Yes | |
| Transfer type | Yes | Constant value – transfer to the tax office |
| Urgency mode | Yes | ExpressD0, StandardD1 |
| Transfer execution mode (system) | No | Standard (Elixir), express (ExpressElixir) |
| Currency | Yes | |
| Block | No | Bool type field, owing to which the client will be able to make an explicit wish to make a block (in case of, e.g. a payment order on a free day). A default behaviour in case the parameter is not provided is defined by ASPSP. |
| Transaction ID as assigned by the TPP | Yes | |

#### 3.1.4.3 EEA foreign transfer

| FIELD NAME | REQUIRED | COMMENTS |
|---|---|---|
| Address of the transfer payee | No | |
| Effective date of the transfer | No | For the future effective date of the transfer, the urgency mode refers to that date |
| Transfer amount | Yes | |
| Name of the transfer sender | No | Sender's name completed by the ASPSP in order to avoid a situation when the transfer order coming from the ASPSP contains data other than data of the holder of the account debited. |
| Name of the transfer payee | Yes | |
| Transfer sender's account number | Yes | It may be specified by the TPP, however the PSU should have an option to select the account to be debited after redirection to the ASPSP. |
| Transfer payee's account number | Yes | |
| Transfer description field | Yes | |
| Transfer execution mode | Yes | Standard, express |
| Transfer type (system) | No | SEPA, Instant SEPA, Target |
| Currency | No | Constant value - EUR |
| Block | No | Bool type field, owing to which the client will be able to make an explicit wish to make a block (in case of, e.g. a payment order on a free day). A default behaviour in case the parameter is not provided is defined by ASPSP. |
| Transaction ID as assigned by the TPP | Yes | |

#### 3.1.4.4 Foreign transfer other than EEA

| FIELD NAME | REQUIRED | COMMENTS |
|---|---|---|
| Effective date of the transfer | No | For the future effective date of the transfer, the urgency mode (of the transfer) refers to that date |
| Transfer sender's account number | Yes | Sender's name completed by the ASPSP in order to avoid a situation when the transfer order coming from the ASPSP contains data other than data of the holder of the account debited. |
| Transfer payee's account number | Yes | |
| Name of the transfer sender | No | Sender's name completed by the ASPSP in order to avoid a situation when the transfer order coming from the ASPSP contains data other than data of the holder of the account debited. |
| Name of the transfer payee | Yes | |
| Address of the transfer payee | No | |
| Transfer description field | Yes | |
| Transfer amount | Yes | |
| Currency | Yes | |
| BIC/SWIFT of the payee's bank | No | Conditional fields – required depending on the final specification of the Bank implementing the PolishAPI standard |
| Country of the payee's bank | No | |
| Name of the payee's bank | No | |
| Address of the payee's bank | No | |
| Code of the payee's bank | No | |
| Cost clause | No | |
| Transfer execution mode | Yes | Standard, urgent, express |
| Transfer type (system) | No | SWIFT |
| Block | No | Bool type field, owing to which the client will be able to make an explicit wish to make a block (in case of, e.g. a payment order on a free day). A default behaviour in case the parameter is not provided is defined by ASPSP. |
| Transaction ID as assigned by the TPP | Yes | |

### 3.1.5 Diagrams of Requests under the PIS Service within the Compliance Scope

The diagram was presented in Use Case #1 in Chapter 4.

### 3.1.6 Authorisation of a payment transaction initiated by means of a PIS service

The ASPSP provides a possibility to authorise a payment transaction ordered by the PSU under a payment initiation service within the understanding of the Payment Services Act (PSA), irrespective of the authorisation method and its complexity. The authorisation method is selected by the ASPSP.

## 3.2 Business Definition of the Compliance Scope for the AIS Service

The account information service within the Compliance Scope consists in making available by the ASPSP of data concerning the transaction history and selected information about the payment account to which the PSU has an active on-line access. The access is granted to a TPP acting as an AISP after a prior acquisition of consent as appropriate from the PSU. Additionally, the ASPSP makes available its data filtering mechanisms in accordance with the criteria available on-line in the ASPSP system (i.e. via the electronic banking), e.g.:

a) The transaction booking date (as per the indicated specific booking date and within the specified range of dates)
b) The transaction amount
c) The payment account debits and credits

### 3.2.1 Definition of a Payment Account

This service is provided only for payment accounts to which the given PSU has an on-line access. The account must meet all of the following cumulative conditions:

c) It is an account held for one or more users which is used to effect payment transactions (as per the definition laid down in the PSA).
d) The PSU has an on-line access to the account.

### 3.2.2 Frequency of Requests within the Compliance Scope

As part of the AIS service within the Compliance Scope, the TPP (AISP) may request that the ASPSP sent a payment account history and selected information about the payment account:

a) Up to 4 times within a 24 hour time span from the first request in case when the data collection is not initiated at the PSU's request via the TPP (AISP), but by TPP (AISP) on the basis of consent provided earlier by the PSU;
b) In each instance when the request is initiated directly by PSU via the intermediation of the TPP (AISP).

A higher frequency of requests in case when the data collection is not initiated at the PSU's request via the TPP (AISP) but by the TPP (AISP) on basis of consent expressed earlier by the PSU may be allowed with regard to the AIS service only in the Premium Scope and is subject to separate bilateral arrangements by and between the ASPSP and the TPP (AISP).

### 3.2.3 Scope of Information concerning the Payment Account History within the Compliance Scope

Within the Compliance Scope, the AIS service comprises the provision to the PSU, together with data filtering mechanisms (including transaction history date scopes), of all available on-line account history of transactions booked, pending and rejected at the given payment account and blocked funds, which are visible to the PSU in the APSPS's on-line channel. Whereby, a pending transaction means a transaction that is not booked, not modifiable and which influences the available funds (available balance). Pursuant to the regulations, the provision of the account history is related to the SCA process always (irrespective of the exclusions from the SCA application obligation applied), when the client obtains an online account access for the first time and when the request concerns a history of 90+ days. The SCA may be abandoned if the request concerns a history of payment transactions effected within the last 90 days, on condition that no more than 90 days have lapsed since the access to a history of not more than 90 days was obtained and the SCA was applied.

### 3.2.4 List of Fields Required by the ASPSP in the Compliance Scope

In response to the request sent by the TPP (AISP), the ASPSP sends information from the following fields.

| FIELD NAME | REQUIRED | COMMENT |
|---|---|---|
| Account number | Yes | Account number |
| Account currency | No | For each payment account |
| Given names and surname / PSU's name | Yes | Given name for a natural person and a business name for a legal person |
| PSU's address | No | Address of the account holder |
| Available funds | Yes | Funds available in the account currency - after the transaction |
| Transaction identifier | Yes | Unique identifier of the given transaction as assigned by the ASPSP |
| Amount in original currency | No | For each account history transaction |
| Book balance of the account | Yes | Book balance of the account - after the transaction |
| Account type code | No | A key clearly identifying the account type in the account type dictionary as defined by the ASPSP |
| Account type | Conditionally | A business description of the account related to the dictionary key, provided in the 'Account type code' field.<br>E.g. account for the consumer / business account + product reference, e.g. account, credit card, savings account, etc.<br>Value required conditionally, in case the 'Account type code' field is provided. |
| Transaction date | Yes | For each account history transaction |
| Amount | Yes | For each account history transaction |
| Sender's account number | Conditionally | For each account history transaction. Depending on the transaction type - credit / debit. |
| Payee's account number | Conditionally | For each account history transaction. Depending on the transaction type - credit / debit. |
| Description | No | For each account history transaction |
| Title | Yes | For each account history transaction |
| Transaction status code | No | For each account history transaction. Value of the key of the transaction status dictionary as defined by the ASPSP. |
| Transaction status | Conditionally | Business description of the transaction status.<br>Required, if the dictionary value was provided in the 'Transaction status code' field |
| Transaction type | No | Credit/debit transaction.<br>For each account history transaction |
| Currency of original transaction | No | For each account history transaction |
| Account type name (defined by the ASPSP) | | Product's commercial name |
| Data of the tax office | No | Only for transfers to tax authorities / customs authorities in Poland |
| Book date | No | For each account history transaction |
| Exchange rate date | No | For each account history transaction |
| TPP's Transaction ID | No | Unique ID of the given transaction as assigned by the TPP |
| Payer's ID with the tax authorities | Conditionally | Only for transfers to tax authorities / customs authorities in Poland |

| | | |
|---|---|---|
| Liability ID with the tax authorities | No | Only for transfers to tax authorities / customs authorities in Poland |
| Transaction exchange rate | No | For each account history transaction |
| Currency code before the conversion transaction | No | As per the ISO standard |
| Currency code after the conversion transaction | No | As per the ISO standard |
| Period number | Conditionally | Only for transfers to tax authorities / customs authorities in Poland |
| Payee's account virtual number | No | For each account history transaction |
| Year | Conditionally | Only for transfers to tax authorities / customs authorities in Poland |
| Form symbol | Conditionally | Only for transfers to tax authorities / customs authorities in Poland |
| Period type | Conditionally | Only for transfers to tax authorities / customs authorities in Poland |
| Payment type | No | Only for transfers to the Social Insurance Institution (ZUS) |
| Unique ID of the payment instrument by which the transaction was effected | No | E.g. payment card number (masked, as per the data presentation in the ASPSP's online interface) |
| Data of the owner of the payment instrument by which the transaction was effected | No | Data of the payment card holder |
| Type of operation | Yes | For each account history transaction |
| Account balance after the transaction | No | For each account history transaction |
| Type of the payer's ID with the tax authorities | Conditionally | Only for transfers to tax authorities / customs authorities in Poland |
| Name of the incoming transfer payee | No | For each incoming transfer transaction at the account history |
| Address of the incoming transfer payee | No | For each incoming transfer transaction at the account history |
| Name of the outgoing transfer payee | Conditionally | For each outgoing transfer transaction at the account history. Depending on the transaction type - credit / debit. |
| Address of the outgoing transfer payee | No | For each outgoing transfer transaction at the account history |
| Address of the payee's bank | Conditionally | For foreign transfers only |
| Code of the payee's bank | Conditionally | For foreign transfers only |
| Name of the payee's bank | Conditionally | For foreign transfers only |
| BIC/SWIFT of the payee's bank | Conditionally | For foreign transfers only |
| Code of the country of the payee's bank | Conditionally | For foreign transfers only |
| Name of the incoming transfer sender | Yes | For each incoming transfer transaction at the account history |
| Address of the incoming transfer sender | No | For each incoming transfer transaction at the account history |
| Name of the outgoing transfer sender | Yes | For each outgoing transfer transaction at the account history |
| Address of the outgoing transfer sender | No | For each outgoing transfer transaction at the account history |
| Account name as defined by the client | No | Provided the ASPSP makes available such a service. |
| Name of the transaction initiator | Conditionally | In case of transactions originated by people other than the account holder (given name and surname) |

| | | |
|---|---|---|
| Address of the transaction initiator | No | In case of transactions originated by people other than the account holder |
| TPP's name | No | In case of transactions initiated as part of the PIS service |
| MCC | No | Code for each transaction / operation made using the card |
| Reason for rejection | No | In case of rejected transactions |
| Rejection date | No | In case of rejected transactions |
| Hold end date | No | In case of account holds |
| VAT No | Conditionally | Payer's basic identifier with the Social Insurance Institution (ZUS), i.e. NIP. |
| Payer's additional identification number with the Social Insurance Institution (ZUS) | Conditionally | Value of the payer's additional identifier with the Social Insurance Institution (ZUS) (the value appropriate to the type of payer's additional identifier selected in the 'Type of payer's additional identifier' field) |
| Type of payer's additional identifier | No | Dictionary value defining the type of the payer's additional identifier with the Social Insurance Institution (ZUS). |
| Declaration number | No | Value of declaration number for transfers to the Social Insurance Institution (ZUS) compliant with the form of this type of transfers |
| Declaration period | No | Value of declaration period for transfers to the Social Insurance Institution (ZUS) compliant with the form of this type of transfers |
| Liability ID with the Social Insurance Institution (ZUS) | No | Value of ID of the liability to be transferred to the Social Insurance Institution (ZUS) compliant with the form of this type of transfers |

The fields described in the table above become mandatory for ASPSPs in relation to the scope of information about payment accounts and transactions the given ASPSP makes available in its online interface, save exceptions stipulated in the law (e.g. with regard to particularly protected data concerning payments or personal data). To the scope of data concerning the account and transactions, each ASPSP may add additional fields, taking advantage for this purpose of the auxData type Map field within the AccountInfo, TransactionInfo, TransactionHoldInfo, TransactionPendingInfo and TransactionRejectedInfo structures.

The list of fields made available when the ASPSP allows the use of the account list retrieval within the process of granting consent for the AIS or PIS services.

| FIELD NAME | REQUIRED | COMMENT |
|---|---|---|
| Account number | Yes | Account number in a masked form, only the 2 first and last 4 digits of the account number visible without masking, according to the ASPSP's decision |
| Account type name (defined by the Bank) | Yes | Product's commercial name |
| Account type | Yes | E.g. account for the consumer / business account + product reference, e.g. account, credit card, savings account, etc. |

### 3.2.5 Diagrams of Requests under the AIS Service within the Compliance Scope

The diagram was presented in Use Case #2 in Chapter 4.

## 3.3 Business Definition of the Compliance Scope for the CAF Service

The service of confirmation of funds at the payer's payment account in an amount sufficient to effect the payment transaction within the Compliance Scope consists in sending a request by the TPP acting as a PIISP to the ASPSP for a confirmation whether or not the PSU's payment account has funds in the amount as determined in the request on the basis of consent granted earlier by the PSU. In response, the ASPSP sends a message in the form of 'YES' or 'NO'.

### 3.3.1 List of Fields Required by the ASPSP in the Compliance Scope

In order to support a request concerning a conformation of funds at the payer's payment account in an amount sufficient to effect a CAF payment transaction within the Compliance Scope correctly, the ASPSP may request from the PSU, via the TPP (PIISP), that the following fields be completed with transaction order data. Details concerning the fields are defined in Chapter 5.7 Canonical Data Model.

| FIELD NAME | REQUIRED | COMMENTS |
|---|---|---|
| Identifier of account the request concerns | Yes | Account previously connected with the payment instrument on the basis of consent granted by the PSU. |
| Amount | Yes | |
| Currency | Yes | Currency of transaction |

### 3.3.2 Diagrams of Requests under the CAF Service within the Compliance Scope

The diagram was presented in Use Case #3 in Chapter 4.

# 4 Sample Use Cases

The current PolishAPI standard version described the manner of performance of XS2A interface-based transactions within the Compliance Scope as defined in Chapter 3 hereof and the TPP may participate in such transactions in one of the roles defined.

Examples illustrating the use of particular services were presented in this chapter. Their aim is only to illustrate the steps for particular services and should not be treated as an exhaustive list of admissible use cases.

## 4.1 Use Case #1: initiation of a single payment by the PISP (PIS)

The use of a PIS service within the Compliance Scope as presented in this Use Case consists in the initiation by the TPP acting as a PISP of a payment transaction debited to the PSU's payment account held by the ASPSP on the basis of applicable provisions of the Payment Services Act. The ASPSP may reject the transaction if the TPP (PISP) has not been identified as an entity authorised to effect a PIS service.

### 4.1.1 Single payment initiation by the PISP using a mechanism of ASPSP-side authentication

The diagram below concerns the processes described in Chapter 1.4.4.1.1 (Process of Granting Consent by PSU to Effect the PIS Service – Authentication on the ASPSP's Side) and 3.1.2 (Information about the Transaction Status).

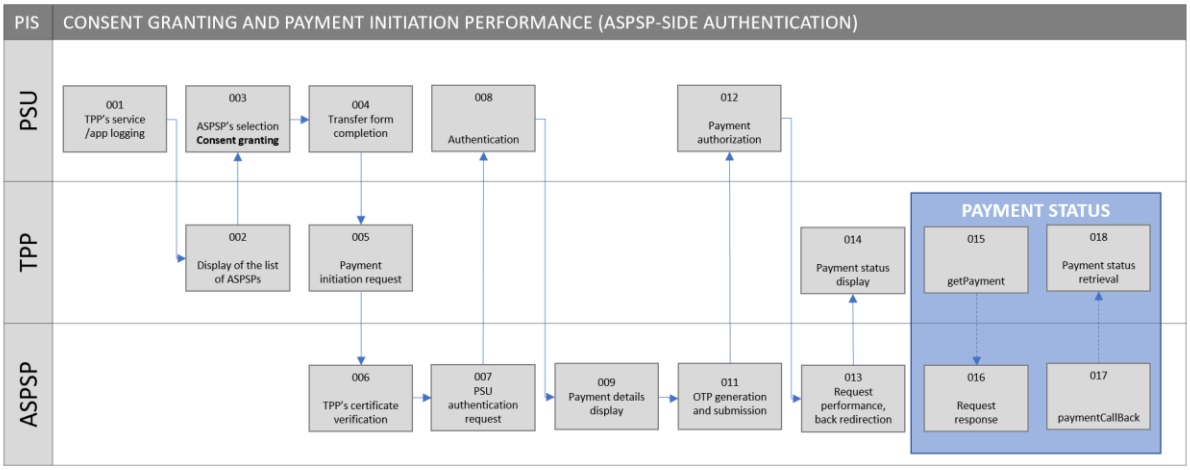



*Figure 13: PIS / Grant of consent and performance of payment initiation and payment status retrieval*

### 4.1.2 Single payment initiation by the PISP using a mechanism of ASPSP-side authentication with the ASPSP-side selection of the account

The diagram below concerns the processes described in Chapter 1.4.4.1.1 (Process of Granting Consent by PSU to Effect the PIS Service – Authentication on the ASPSP's Side) and 3.1.2 (Information about the Transaction Status).

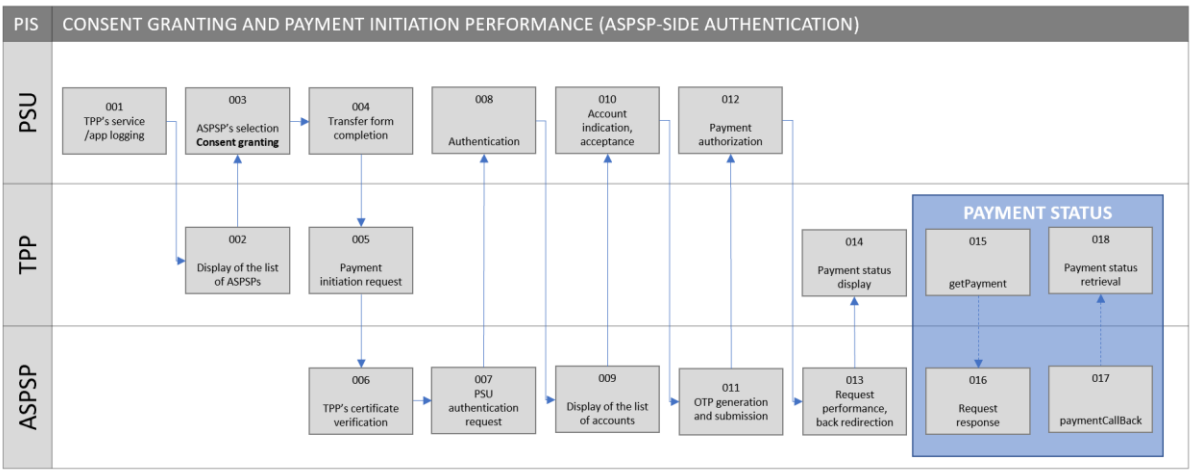



***Figure 14: PIS / Grant of consent and performance of payment initiation (ASPSP-side selection of the account) and payment status retrieval***

### 4.1.3 Single payment initiation by the PISP using an authentication mechanism in an external authorization tool

The diagram below concerns the processes described in Chapter 1.4.4.1.3 (Process of Granting Consent by PSU to Effect the PIS Service – Authentication in An External Authorization Tool) and 3.1.2 (Information about the Transaction Status).

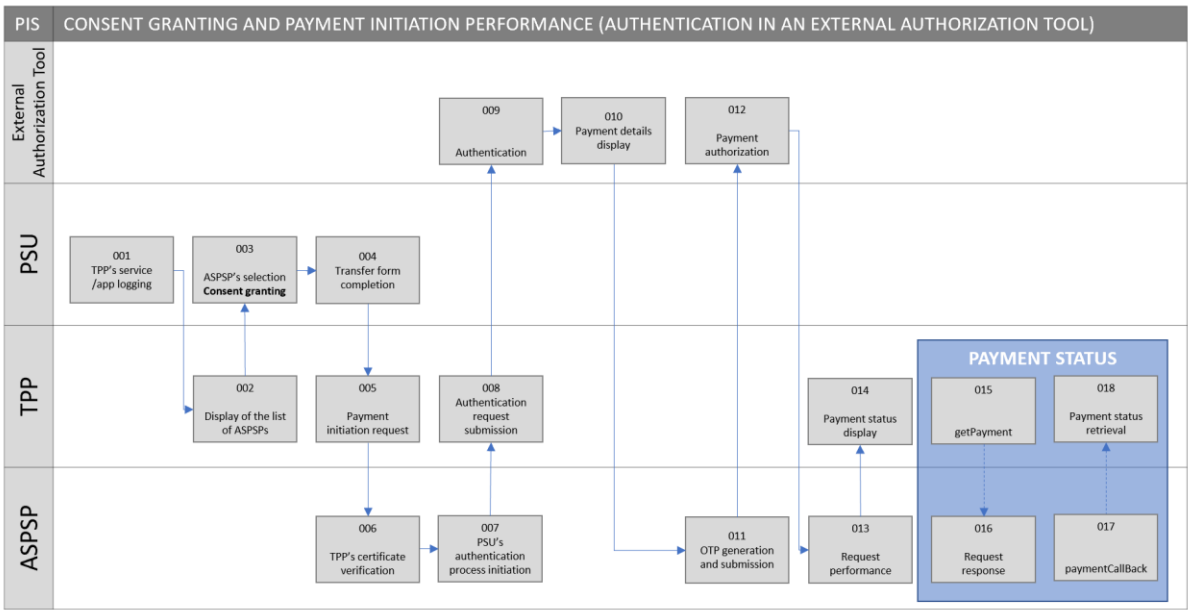



***Figure 15: PIS / Grant of consent and performance of payment initiation (authentication in an external authorization tool) and payment status retrieval***

## 4.2 Use Case #2: payment account information display by the AISP (AIS)

The use of an AIS service within the Compliance Scope as presented in this Use Case consists in the acquisition by the TPP acting as an AISP of information about the PSU's payment account held by the ASPSP on the basis of applicable provisions of the Payment Services Act.

The process of granting consent to use the AIS service has been described in Chapter 1.4.4.2. The Case described below assumes that the PSU granted AISP his/her consent to collect a defined scope of data. A request is initiated by the AISP on behalf of the PSU.

This process has been presented at a high level in the diagram below.

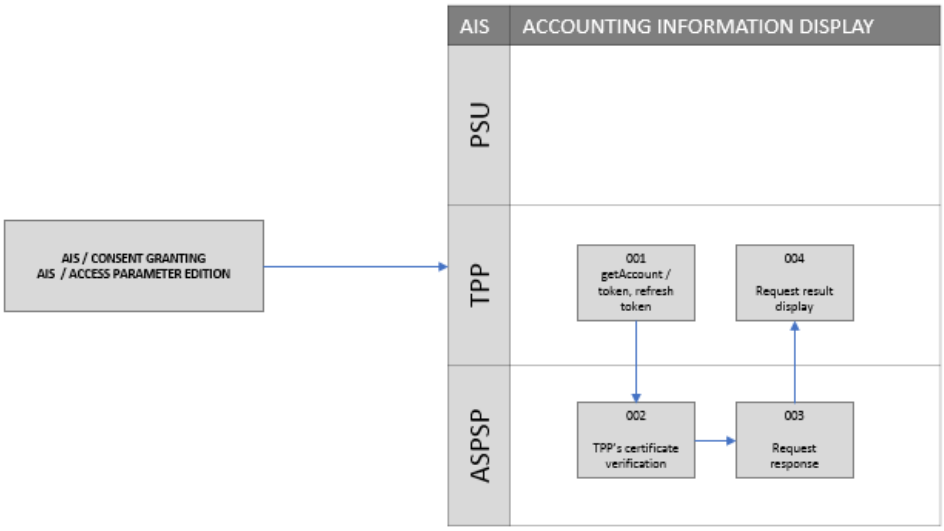



***Figure 16: AIS / Payment account information display by the AISP***

## 4.3 Use Case #3: request for confirmation of funds by a PIISP (CAF)

The use of an CAF service within the Compliance Scope as presented in this Use Case consists in the initiation by the TPP acting as an PIISP of a request for availability of funds in the transaction amount at the PSU's payment account on the basis of applicable provisions of the Payment Services Act.

The PSU must previously indicate to the PIISP a payment account that will be verified in each case in terms of funds availability and grants his/her consent for the ASPSP holding the given payment account to answer such requests. The PSU initiates a business process requiring a verification whether or not the payment account previously indicated by the PSU has funds available in the amount equal at least to the requested amount. In order to effect the service, the PIISP establishes an XS2A session with ASPSP, sends a request and receives a 'YES' or 'NO' answer.

This process has been presented at a high level in the diagram below.

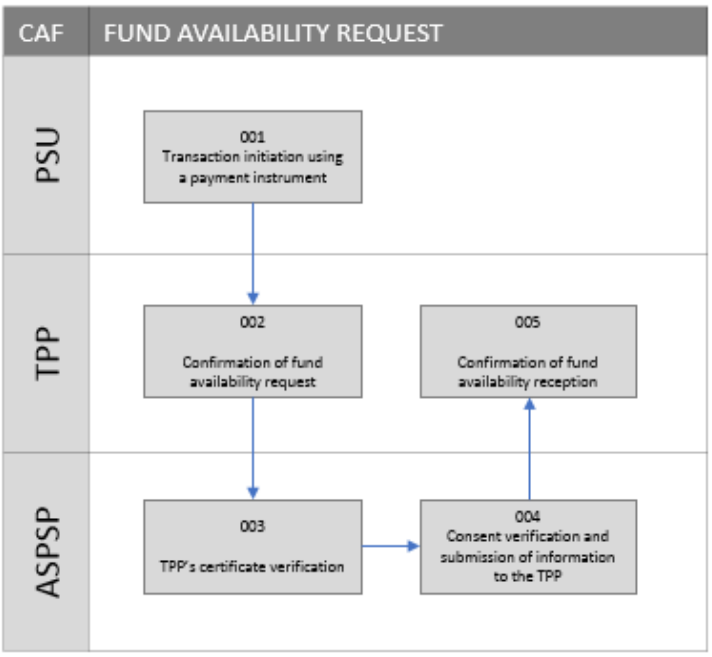



***Figure 17: Figure 16: CAF / Request for confirmation of funds***

This process may be used to, for example, authorise a transaction effected by payment instruments not linked to the payment account held by the instrument issuer.

# 5 Polish API Technical Specification

## 5.1 Technical Assumptions

The table below presents the technical assumptions made for the PolishAPI:

| No. | ASSUMPTION | DESCRIPTION | GROUNDS |
|---|---|---|---|
| 1 | Direct TPP-ASPSP communication | In the basic variant of the PolishAPI, the TPP and the ASPSP communicate with each other directly. | The peer-to-peer architecture used increases the safety and efficiency as well as allows the avoidance of a single point of failure. |
| 2 | Role of the PSD2 HUB | In case the ASPSP uses the services of a PSD2 HUB, it is neutral for the TPP. The PSD2 HUB presents itself using the ASPSP's certificate, from the TPP's perspective, there is no difference whether it gets connected with the PSD2 HUB or directly with the ASPSP. | Efficient and Safe API and PolishAPI Standard Implementation. |
| 3 | The TPP-ASPSP communication is a server-server one | No direct communication of the client's device (e.g. a mobile app) with the ASPSP's PolishAPI servers is allowed. The TPP should be legally obliged to secure the access keys (so-called access certificate). In particular, the access certificates may not be installed in mobile apps made available to the PSUs) | |
| 4 | Separation of the client's consent step from the operation performance step | The client's consent for the service step will be separated from the performance of the operation itself. One of the effects will be the fact that the consent in itself will not entail any financial consequences. | Flexibility in the implementation of new services, including the Premium Services. |
| 5 | Scope of the PolishAPI | The scope of PolishAPI specifies the following:<br>- way of granting consent for the performance by the TPP of an operation on behalf of the customer<br>- scope of operations and rights<br>- URL where the given service is available<br>- standard scope of parameters per service<br>- security mechanisms<br>- communication principles<br>- error handling<br>The PolishAPI does not specify the following:<br>- full scope of functionalities to be made available by the ASPSP as well as the information as to which ones of them will not be | The RTS make the scope of functionality and the scope of data dependent on the scope of functionality made available in the Internet banking, which is different for each ASPSP |

| | | | |
|---|---|---|---|
| | | within the Compliance Service<br>- full specification of fields per service for each ASPSP | |

## 5.2 XS2A Session Establishment

The use by a TPP of business services (AIS, PIS, CAF), made available at the ASPSP's side, requires the so-called communications session to be started at the side of the technical solutions of the entities listed.

The process of starting a communications session with the XS2A interface comprises the requests and responses exchanged between the TPP and ASPSP using the technical services of that interface (AS – Authorization Service), in effect of which the communications session is started and the ASPSP's side and its technical representation, including such metadata as its validity date, is sent to the TPP.

The establishment of a communications session may take into consideration the necessity to ensure a strong authentication of the PSU. Considering the SCA method selected (*ASPSP-side* or *decoupled*), the communications session starting process may vary. Irrespective of the said differences as regards the SCA methods, the communications session establishment is based on the OAuth 2.0 standard assumptions in the following matters:

a) The required method of authorisation of access to the ASPSP resources made available via the XS2A business interface is the return by the ASPSP-side server, in response to each request sent by the TPP, of a one-time authorisation code – within the understanding of Art. 4 of the RTSs, which will be used by the TPP in the next step to obtain an access code as per the provisions of the OAuth 2.0 standard.
b) The *state* parameter sent by the TPP in the authorization request (item a) must be unique for each authorization process performed by the given TPP.
c) It is suggested that the one-time authentication code on the ASPSP's server side and the access token were the resource identifier in the database, in which the indicated resource data will be used to identify the PSU for whom the access token is generated or the defined business operation is effected.
   The use of the so-called *stateless token* (e.g. JWT Token - RFC 7519) should be resorted to only in case when the disclosure of the ASPSP's customer data (including the ID) is compliant with the security policy
d) Together with the access token, the scope parameter (the same as in the request sent by the TPP) is transferred to the TPP.

The TPP's use of a valid communications session is a pre-condition for the reception of correct responses to the requests sent to the XSS2A interface business services.

The diagram of the XS2A communications session establishment process flow is presented below.

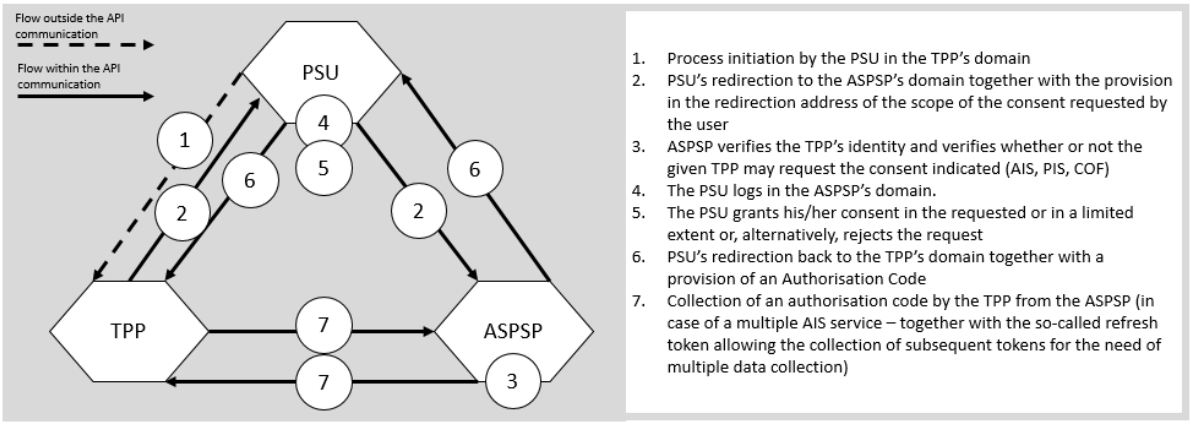



***Figure 18: Multilayer XS2A session establishment diagram***

Each transaction under the XS2A interface business services takes place as part of a dedicated and separate communications session, whereby for the selected methods under the AIS and PIS services and assuming the fulfilment of the conditions defined in the business and technical parts of the PSD2 Directive, a multiple use of the same communications session when sending requests to the XS2A interface business services, without the need to perform the SCA procedure for the PSU on each occasion and while maintaining the validity of the communications session, after which it will be automatically cancelled by the ASPSP.

The XS2A interface technical service may offer an alternative and automatic (i.e. not requiring any interaction with the PSU) mechanism of starting a communications session, i.e. the refresh token. The mechanism allows a refreshment of the communications session previously cancelled by the ASPSP without the need to repeat the SCA procedure, based on a separate session identifier (*refresh token*) sent to the TPP in response to the request to start the original communications session. This communications session starting mechanism may not be used only for those XS2A interface business services for which it was allowed under the regulatory provisions of the PSD2 Directive, e.g. multiple AIS. This mechanism is additionally described in Chapter 11.3.

## 5.3 Definition of Access Token

Access token is a technical representation of the communications session with a defined validity that was established by and between the TPP and the ASPSP in the context of a precisely defined PSU and for a precisely defined scope of the ASPSP's side services and resources to which the TPP obtained access. The access token is a series of characters the role of which is to confirm the access authorization to secured resources made available via the XS2A interface services. The access token may have various forms and ways of interpretation. The final properties of the access token depend on the ASPSP-side authorization systems which implement the Polish API standard.

Pursuant to the regulations described in the RTSs of the PSD2 Directive, depending on the XS2A interface business service (AIS, PIS), for which the communications session was established, the access token may be used one time or many times before it is cancelled by the ASPSP, which will entail a necessity to re-run the SCA procedure for the PSU, in case of an intention to use the service again.

## 5.4 Mutual Authentication of the TPP and the ASPSP

The mutual authentication of the TPP and the ASPSP takes place on the basis of the X.509v3 certificates issued by a trusted third party. A trusted third party may by, in particular, an institution performing the role of the Identity Hub. It may also be any other party offering a trust-based relationship based on public key infrastructure mechanisms.

All operations consisting in the flows specified and described in the standard are possible only in a situation of a correct authentication in a process that comprises a mutual authentication of the server and the client (mutual authentication). The TPP and the ASPSP may have the role of both a server and a client, but, in each case, it is required that the parties to the communication mutually authenticate each other.

A description of the public key infrastructure used for the needs of authentication of parties (TPP, ASPSP, PISP) is not a part of the Polish API standard. It should be described in separate documents (working standards), taking into account the structure of trust relation between the certification institutions and the interoperability of the Polish API with other solutions of this type in place in other countries.

## 5.5 Communication Protocol

HTTP /2 or HTTP 1.1, secured by TLS 1.2+ with a mutual authentication of the client and the server by means of the X.509v3 certificates will be used as the communication protocol. Due to the requirement to ensure non-repudiation (request and response signing), only the POST method will be used in the http communication.

## 5.6 Resource Name Diagram

The PolishAPI services will be made available under addresses compliant with the following model:

https://{DNS domain}/{v{Resource version number 1}/{Resource name 1}}/.../v{Resource version number n}/{Resource name n}

Field description:

a) DNS domain/address – where the ASPSP makes the PolishAPI services available (information made available in the PSD2 register)
b) Resource version number – number of the version according to PolishAPI specification (digit before the dot) and subsequent interface number within the given ASPSP (digit after the dot)
c) Resource name – number of the resource the request concerns; resources nesting routes, e.g. /v{version number of resource accounts}/accounts/v{number version of resource transactionsDone}/transactionsDone

## 5.7 Canonical Data Model

In order to unify the data types, a canonical data model was proposed as presented in the table below. A detailed model definition is given in Annex No. 1.

| CLASS NAME | PURPOSE | API |
|---|---|---|
| AccountIban | Account no | Primary |
| AccountBaseInfo | Class containing basic information about the account data | Primary |

| | | |
|---|---|---|
| AccountForeign | Account number in the format for foreign transfers | Primary |
| AccountInfo | Class of information about the account | Basic / CallBack |
| AccountInfoRequest | Class of request concerning a single account | Primary |
| AccountResponse | Class of response to a request concerning the PSU's account | Primary |
| AccountsRequest | Class of request concerning accounts | Primary |
| AccountsRequest | Class of callback with a list of accounts | CallBack |
| AccountsResponse | Class of response to a request concerning many PSU's accounts | Primary |
| AddPaymentResponse | Class of response to the payment initiation request | Primary |
| Address | Class containing data of a postal address | Basic / CallBack |
| AuthorizationDataRequest | Class with feedback with authorization code compliant with the OAuth 2.0 standard for the authorization code method | CallBack |
| AuthorizeRequest | Class containing data required to for the TPP's authorization | Primary |
| BankAccountInfo | Class containing bank's data | Basic / CallBack |
| Bank | Class containing bank's data used in AIS requests | Basic / CallBack |
| CallBackResponse | Basic class for responses to callback | CallBack |
| ConfirmationOfFundsRequest | Class of request concerning the available funds at the account | Primary |
| ConfirmationOfFundsResponse | Class of response to a request concerning the available funds at the account | Primary |
| CurrencyRate | Class containing conversion rates | Primary |
| DeleteConsentRequest | Class containing data allowing the identification of consents to be deleted at the ASPSP's side | Primary |
| DictionaryItem | Class containing dictionary item data | Basic / CallBack |
| EatCodeRequest | Class of request to obtain an authorization code compliant with OAuth 2.0 on the basis of a one-time code generated in EAT | Primary |
| Error | Class of information containing data about the error returned | Basic / CallBack |
| GetPaymentResponse | Class of response to the payment request | Primary |
| Map | Class of map <string, string> | Basic / CallBack |
| NameAddress | Class containing the data of the name and address in the form of four data lines | Basic / CallBack |
| PageInfo | Class containing data allowing the use of the paging mechanism | Primary |
| PaymentDomesticRequest | Class of request concerning the standard domestic transfer initiation | Primary |
| PaymentEEARequest | Class of request concerning the SEPA foreign transfer initiation | Primary |
| PaymentInfo | Class of information about the payment | Primary |
| PaymentNonEEARequest | Class of request concerning the initiation of a foreign transfer other than SEPA | Primary |
| PaymentRequest | Class of request concerning the payment status | Primary |
| PaymentStatus | Dictionary of payment statuses | Basic / CallBack |
| PaymentStatusInfoRequest | Class of a callback with information about the payment status | CallBack |
| PaymentTaxRequest | Class of request concerning the tax transfer | Primary |

| | | |
|---|---|---|
| | initiation | |
| PaymentsRequest | Class of request concerning multiple payment status | Primary |
| PaymentsResponse | Class of response to a request concerning multiple payments | Primary |
| Payor | Class of information about the payer to the Social Insurance Institution (ZUS) and Tax Office | Primary |
| RecipientPIS | Class containing recipient's data used in PIS requests | Primary |
| RecipientPISTax | Class containing recipient's data used in PIS requests for tax operations | Primary |
| RecipientPISForeign | Class containing recipient's data used in PIS requests for foreign operations | Primary |
| RequestHeader | Class containing information about the PSU | Primary |
| RequestHeader | Class containing metadata of the request to the callback interface | CallBack |
| RequestHeaderAIS | Class containing information about the PSU for the requests to the AIS service of the XS2A interface | Primary |
| RequestHeaderAISCallback | Class containing information about the PSU for the requests to the AIS service of the XS2A interface enabling the response to be sent in the form of a request to the callback interface | Primary |
| RequestHeaderCallback | Class containing information about the PSU for the requests enabling the response to be sent in the form of a request to the callback interface | Primary |
| RequestHeaderWithoutToken | Class containing information about the PSU for the requests not requiring an access token | Primary |
| RequestHeaderWithoutTokenCallback | Class containing information about the PSU for the requests that do not require an access token and enabling the response to be sent in the form of a request to the callback interface | Primary |
| ResponseHeader | Class containing response metadata | Primary |
| ResponseHeader | Class containing response metadata | CallBack |
| SenderPIS | Class containing sender's data used in PIS requests | Primary |
| SenderRecipient | Class containing sender's/recipient's data used in AIS requests | Basic / CallBack |
| SocialSecurityPayor | Class contain information about the payer of contributions to the Social Insurance Institution (ZUS) | Primary |
| TokenRequest | Class containing data required to obtain an access token | Primary |
| TokenResponse | Class of responses containing, without limitation, the access token | Primary |
| TransactionDetailRequest | Class of request concerning a single transaction | Primary |
| TransactionDetailResponse | Class of response to a single transaction request | Primary |
| TransactionsDoneInfoResponse | Class of response containing a list of transactions done | Primary |
| TransactionDoneInfoRequest | Class of callback containing a list of transactions done | CallBack |
| TransactionHoldInfo | Class containing data about holds at the account | Primary |
| TransactionHoldInfoResponse | Class of response containing a list of holds at the account | Primary |

| | | |
|---|---|---|
| TransactionHoldInfoRequest | Class of callback containing a list of holds at the account | CallBack |
| TransactionHoldRequest | Class of request concerning holds at the account | Primary |
| TransactionInfo | Class describing the payment transaction booked | Basic / CallBack |
| TransactionInfoBase | Base class describing a payment transaction | Basic / CallBack |
| TransactionInfoCard | Class representing information about the card within the framework of a transaction | Primary |
| TransactionInfoRequest | Class of a request concerning transactions | Primary |
| TransactionInfoRequestBase | Base class for requests concerning transactions | Primary |
| TransactionInfoTax | Class of information given for a transfer to a tax office / customs authorities | Primary |
| TransactionInfoZUS | Class of information given for a transfer to the Social Insurance Institution (ZUS) | Primary |
| TransactionPendingInfo | Class describing the pending payment transaction | Basic / CallBack |
| TransactionPendingInfoResponse | Class of response containing a list of pending transactions | Primary |
| TransactionPendingInfoRequest | Class of callback containing a list of transactions pending | CallBack |
| TransactionRejectedInfo | Class describing a rejected payment transaction | Basic / CallBack |
| TransactionRejectedInfoResponse | Class of response containing a list of rejected transactions | Primary |
| TransactionRejectedInfoRequest | Class of callback containing a list of transactions rejected | CallBack |
| TransferData | Class containing transfer data | Primary |
| TransferDataBase | Class containing general transfer data | Primary |
| TransferDataCurrencyRequired | Class containing transfer data with the required information about the currency | Primary |

## 5.8 Operations

Due to the requirement to ensure non-repudiation in the http communication, only the POST method will be used as it allows the JWS Signature format signature. Within the operation, the context of a specific user is determined on the basis of an access token. This principle applies both to the requests sent by the TPP to the ASPSP's XS2A interface , as well as requests sent from the ASPSP to the XS2A callback interface provided by the TPP.

## 5.9 Sorting

The records returned are sorted chronologically (on a reversed basis) as per the transaction date.

## 5.10 Filtering

Filtering in the AIS service takes place after setting the appropriate properties in the TransactionInfoRequest class object:

a) transactionIdFrom – transactions from the given transaction ID 'chronologically'
b) transactionDateFrom – initial transaction date of the data range requested
c) transactionDateTo – final transaction date of the data range requested
d) bookingDateFrom – initial booking date of the data range requested
e) bookingDateTo – final booking date of the data range requested
f) type – CREDIT or DEBIT
g) minAmount – minimum operation amount within the data range requested

h) maxAmount – maximum operation amount within the data range requested

## 5.11 Paging

Results of requests containing many records (where many is > 100) should be paged. Subsequent pages will be retrieved by setting the pageId attribute in the TransactionInfoRequest class. The pageId attribute should be set to the value returned in the PageInfo class of the previous request. Forward navigation - nextPage, backward navigation - previousPage. The number of records per page is defined by the perPage attribute in the TransactionInfoRequest class. The maximum settable value for the perPage parameter is suggested to be 100. Page numbers start from 1. Skipping the pageId parameter will return the first page.

## 5.12 Response Statuses

The technical statuses will be returned by the following http codes:

| STATUS | DESCRIPTION |
|---|---|
| 200 OK | The operation was successful |
| 304 Not Modified | Used when the cache headers have been used |
| 400 Bad Request | The request is syntactically incorrect |
| 401 Unauthorized | Incorrectly authenticated user |
| 403 Forbidden | Authorisation error (no rights to access the resource) |
| 404 Not Found | Reference to a non-existing resource |
| 405 Method Not Allowed | Use of an inappropriate method – the method used in the request is not allowed for the resource indicated (Only POST is used) |
| 406 Not Acceptable | Incorrect Accept heading in the request (the server does not support it) |
| 415 Unsupported Media Type | If an incorrect content type was set in the request |
| 422 Unprocessable Entity | Validation error |
| 429 Too Many Requests | Request rejected due to the fact that the maximum number of requests to access the resource has been exceeded |

## 5.13 HTTP Header

The following HTTP headers will be used in the requests:

| HEADER | TYPE | DESCRIPTION |
|---|---|---|
| Authorization | String | Authentication header (used when sending a token). The value of the Authorization header should comprise the 'type' + 'credentials', where, in case the 'type' token approach is applier, the 'type' should have the value of 'Bearer'. |
| Date | Date | Request timestamp in the RFC 5322 date and time format. |
| Accept | Content type | Should be set to application/json Otherwise the application should return 406 Not Acceptable HTTP. |
| Accept-Encoding | Gzip, deflate | The operation should support GZIP and DEFLATE coding, it may also return non-compressed data. |
| Accept-Language | 'pl', 'en', etc. | Defined the preferred language in which the response is to be returned. The operation does not have to support this header |
| Accept-Charset | Charset type like 'UTF-8' | UTF-8 |
| Content-Type | application/json | Should be set to application/json. Otherwise, the operation returns: 415 Unsupported Media Type HTTP status code |
| X-JWS-SIGNATURE | String | JWS Signature (Detached) |

Response headers:

| HEADER | REQUIRED | DESCRIPTION |
|---|---|---|
| Date | Yes | Timestamp on the basis of the GMT server time as per RFC 5322 |
| Content-Type | Yes | application/json |
| Content-Encoding | Yes | GZIP or DEFLATE |
| Expires | No | Defines the cache policy for slowly varying objects e.g. Expires: Mon, 25 Jun 2012 21:31:12 GMT |
| Size | Yes | Response size in bytes |
| ETag | No | Resource version identifier |
| Last-Modified | No | Last resource modification date |
| X-JWS-SIGNATURE | Yes | JWS Signature (Detached) |
| Location | No | URI, to which the authorization request should be redirected in case of the 302 response from the ASPSP |

## 5.14 Message format

The data exchange format will be JSON with the UTF-8 coding. All messages have a defined JSON schema draft #4. The parameter names will be saved camelCase.

## 5.15 Basic Data Formats

| SIZE | JSON FORMAT | DESCRIPTION |
|---|---|---|
| Text | String | Text coded in UTF-8 |
| Dates | String | Pursuant to ISO8601.<br>Date and time will be represented in the form of YYYY-MM-DD to YYYY-MM-DDThh:mm:ss.ccczzzzzz with the mandatory specification of the time zone<br>Designations:<br>YYYY – year, MM – month, DD – day, hh – hour, mm – minute, ss – second, ccc – millisecond (optional) zzzzzz – e.g. +02:00 or Z to denote universal time<br>For example: 2016-10-10T12:00:05.342+01:00 |
| Amounts | String | Written as digits with a sign separating the integer part from the fractional part up to the second decimal place (the dot sign). In case of positive value, no additional signs are given. In case of negative value, the '-' sign is added before the number |
| Integer | Number | The integer numbers are represented without group separators |
| Real number | String | Real numbers are represented without group separators and with the '.' sign as a decimal separator |
| Country codes | String | In accordance with ISO 3166 |
| Currencies | String | Currency symbols in accordance with ISO 4217 |
| Account numbers | String | IBAN numbers in accordance with ISO 13616 |
| Bank identifiers | String | Bank Identifier Codes (BIC) in accordance with ISO 9362 |
| Logical value | Boolean | Flags and logical tags which may take one of two values: true or false |

# 6 Security of information

This chapter presents general security requirements vital from the perspective of standard creation and its designing on the basis of the IT solution ecosystem compliant with the PolishAPI. Detailed security requirements comprising additionally the problems of security of PolishAPI-based system implementation, operation and maintenance will be described in a separate document and its development will be preceded by a preparation of a detailed risk model. In consequence, they will be an answer to specific identified threats, places where the threats may potentially materialise as well as the assessment of the level of materiality and probability and impact of the cases when such threats should materialise on the safety and operational continuity of the PolishAPI ecosystem.

Particular PolishAPI-based IT system components should have a clearly defined separation between the data layer, the controller's layer and the presentation layer. The components should be separated from each other by a defined security measures such as network segmentation or the firewall rules.

## 6.1 TPP's Authentication

TPP entities must be properly authenticated before they are granted access to the XS2A interface so as to ensure a high level of protection both from an impersonation on the part of unauthorised users of lawful TPPs and from an unauthorised escalation of the authorisation level by TPPs having a legal access to the interface. The authentication takes place on the basis of public key certificates during a mutual authentication process via the TLS 1.2+ protocol.

Authentication errors must result on the denial of access to the XS2A interface.
The user and session authentication data as well as operation authentication tokens may not be transferred in the form of URI parameters.

## 6.2 TPP's Authorisation

The TPP's authorisation must be based on the RBAC model (*Role Based Access Control*), where the level and scope of access to particular API resources depends on the role of the PolishAPI user.

The use of particular methods must be authorised so that the rights depended on the user's role. In particular, the level and scope of authorisation should be different for TPPs depending on the scope of their rights.

## 6.3 PSU's Authorisation for Operations made by a TPP

Irrespective of the PSU's authentication mechanism applied within the AIS/PIS/CAF services, it is assumed that the process will end by the issue by the ASPSP of an access token as defined in Chapter 5.3 of the specification. The operations are always requested by the TPP via a valid access token.

## 6.4 Security in case of Mobile Apps

For security reasons in a model using the authentication mechanism on the ASPSP's side, the redirection to the ASPSP's site and back to the TPP's site will take place in the browser (browsers other than the system browse will not be allowed, the application of the WebView will not be allowed) and not in the mobile app itself. The TPP may register the appropriate URL in the device's operating system so that after the redirection back to the TPP the mobile app be automatically resumed.

## 6.5 Data Validation and Integrity Assurance

The data must be subject to validation procedures in the context of variable types, the scope and the model of limit values. In particular, the structured JSON data must be parsed in accordance with the formal validation procedures, using a white list-based approach. The validation must also be made with regard to the Content-type and Accept (application/json) headers for the compliance of the header value with the actual text of the HTTP message.

During the validation, the digital signature in the header (X-JWS-SIGNATURE) must be validated in the context of the data provided both in the requests and in the http protocol responses as listed in the ASPSP-TPP communication. It should be stressed that this rule applies also in the case of communication initiated by the ASPSP side, in case of a use of the XS2A callback interface provided by the TPP.

In case of Content-type and Accept text validation errors, http message 406 should be returned (Not Acceptable).

Input data validation errors must be registered in logs.

The validation errors must be signalled by the HTTP 400 message (Bad Request) and the data must be rejected.

Not validated or incorrectly validated data must be rejected.

## 6.6 Cryptography

The communication using the PolishAPI must ensure a cryptographic security at two levels:

a) At the level of transmission via (https/TLS). The TLS connection parameter renegotiation must be made in a secure way in accordance with RFC 5746
b) At the message level, to ensure the non-repudiation, it is obligatory to use JSON Web Signature (JWS https://tools.ietf.org/html/rfc7515), the signature reference must be in the X-JWS-SIGNATURE header

Each party to the communication (TPP, ASPSP) must have its own unique two pairs of keys (for transmission and for signature).

Separate certificates must be used to secure the transaction at the https level and at the JWS-SIGNATURE level. For https, the certificate must have an expanded key use (Client Authentication) for signature (Digital signature).

The certificates used to combine the transmission and the signature must be validated in terms of:

a) Validity (certificate validity date from and to)
b) No cancellation (crl/ocsp)
c) Path verification (https://tools.ietf.org/html/rfc4158)

Particularly sensitive information, including identification confirmations and authorization keys, may not be buffered or registered in logs.

Certificates should be issued taking into account the ETSI TS 119 495 specification.

## 6.7 Protection against API Abuse

The API implementation should take into account the mechanisms of protection against excessive requests from the part of the users (both authorised and unauthorised ones), in particular those

generated on purpose with the intention to render the resource unavailable (DoS/DDoS), by an application of mechanisms limiting the number of requests supported over a given time unit. The limit values should be determined after examination of the specific operational conditions. Limits of this kind should be parametrized. The count of the number of resource access requests should base on a key that unambiguously identifies the given TPP (the RequestHeader.tppID class) and the meters implemented per TPP on the server side. The limit excess must be signalled by HTTP message number 429 (Too Many Requests).

The security should be ensured on the basis of OWASP Guidelines - REST Security Cheat Sheet (https://www.owasp.org/index.php/REST_Security_Cheat_Sheet).

## 6.8 Audit Information Logging

It is recommended that the time sources for all the parties using the PolishAPI should be synchronized in order to ensure that the log entries have the correct time stamp.

The key business operation logging should ensure non-repudiation and integrity of entries by using the data from JWS Signature.

A log should contain necessary information that will allow a precise time analysis in case of an incident so that it would be possible to combine particular entries into a single transaction. The element combining particular entries may be, for example, an abbreviation from the authorisation token.

# 7 Technical Description of the Authentication and Authorisation Process

## 7.1 Scope and scope_details Parameters

The *scope* parameter defines the access scopes (corresponding to particular resources (paths) of the specification:

- ais:accounts: Authorisation to effect AIS-Accounts
- ais:account: Authorisation to effect AIS-Account
- ais:holds: Authorisation to effect AIS-Holds – requires that a simultaneous consent to grant account access be requested, i.e. ais:account
- ais:transactionsDone: Authorisation to effect AIS-TransactionsDone – requires that a simultaneous consent to grant account access be requested, i.e. ais:account
- ais:transactionsPending: Authorisation to effect AIS-TransactionsPending – requires that a simultaneous consent to grant account access be requested, i.e. ais:account
- ais:transactionsRejected: Authorisation to effect AIS-TransactionsRejected – requires that a simultaneous consent to grant account access be requested, i.e. ais:account
- ais:transactionDetail: Authorisation to effect AIS-TransactionDetail – requires a simultaneous request for the account access, i.e. ais:account, and one or many consents for the transaction history access, i.e. ais:holds, ais:transactionsDone, ais:transactionsPending, ais:transactionsRejected
- pis:multiplePayments: Authorisation to effect PIS-MultiplePayments
- pis:payment: Authorisation to effect PIS-Payment
- pis:domestic: Authorisation to effect PIS-Domestic
- pis:EEA: Authorisation to effect PIS-EEA
- pis:nonEEA: Authorisation to effect PIS-NonEEA
- pis:tax: Authorisation to effect PIS-Tax
- CAF:confirmationOfFunds: Authorisation to effect CAF-ConfirmationOfFunds

In case, many consent identifiers are provided in the scope parameter, they should be separated by a space character, e.g.:

```
{
        ‘scope’ : ‘ais:account ais:transactionsDone’
}
```

The scope_details parameter defines the time ranges, limitations and details of the given authorisation:

a) as to the scope of resources which are made available (e.g. a list of accounts)
b) time for which they are made available
c) limit of the number of uses
d) list of operations it concerns
e) selected operation parameters, e.g. length of back history, transfer parameters etc.

A specification of the scope_details parameter structure is given in Annex No. 1.

The above parameters are sent by the TPP as POST (due to the possible size of scope_details) in the JSON format - encoded and signed using the JSON Web Signature in accordance with RFC 7515.

## 7.2 Authentication Mechanism on the ASPSP's Side

The process of the PSU's authentication on the ASPSP's side was developed on the *authorization code* method as defined in the OAuth 2.0 standard. The high level business aspect of this mechanism was presented in the diagram below while a detailed procedure of the authentication process and of the process of obtaining authorization for the ASPSP's resources were described in Chapter 11.1.

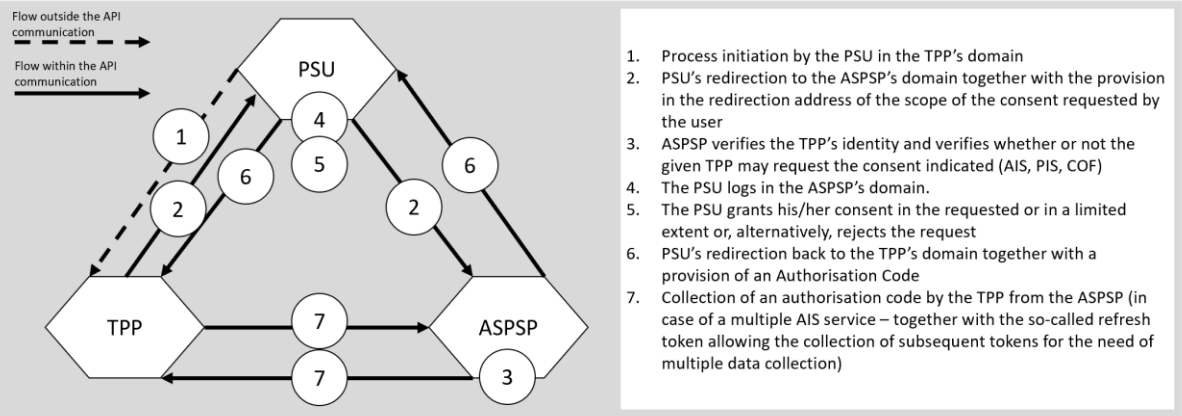



***Figure 19: Authentication Mechanism on the ASPSP's Side***

Below are described the steps and changes with regard to the OAuth 2.0 standard implemented by the PolishAPI in relation with legal and security requirements.

### 7.2.1 Redirection from the TPP to the ASPSP

The redirection comprises the following parameters:

| PARAMETER | REQUIRED | COMMENT |
|---|---|---|
| response_type | Required | 'Code' value |
| client_id | Required | TPP's unique identifier |
| redirect_uri | required | |
| scope | Required | |
| scope_details | Required | |
| state | Required | A random value unique for the TPP – protection against the Cross-Site Request Forgery attack |

### 7.2.2 PSU's authentication and authorisation

Performance on the ASPSP's side.

### 7.2.3 Reverse redirection of the PSU's browser to the TPP

When the PSU is granting an authorization for the TPP, the authorization server delivers this information to the TPP by sending a one-time authorization code, which means that it may be used by the TPP to obtain access to the ASPSP's resources (obtain an access token) exactly one time only. The code is sent within the request to redirect the PSU's browser to the redirect_uri address (using the Content-Type parameter with the value of 'application/x-www-form-urlencoded'). Additionally, the request may also contain the state parameter, which is required only if it was previously sent by the TPP in the authorization request.

A sample reverse redirection to the TPP after the PSU's authentication and the authorization of the TPP's access to the ASPSP's resources:

HTTP/1.1 302 Found

Location: https://[redirect_uri]?code=[authorization_code]

&state=[state]

where:

[redirect_uri] – TPP-side address sent in the authorization request to which the PSU's browser is redirected after the process of that PSU's authentication and the authorization of access to the ASPSP's resources has been completed

[authorization_code] – one-time authorization code confirming a correct authentication of the PSU and the PSU's grant of authorization for the TPP to access the ASPSP's resources

[state] – additional parameter allowing the adjustment of the authorization request with the redirection request after the completion of the PSU's authentication and after the grant by the PSU of authorization to the TPP to access the ASPSP's resources, used to prevent the *'cross-site request forgery*' type attacks.

### 7.2.4 Token collection on the basis of the Authorization Code

| PARAMETER | REQUIRED | COMMENT |
|---|---|---|
| grant_type | required | Value of the 'authorization_code' |
| Code | required | In accordance with the value given in step 7.1.3 |
| redirect_uri | required | Value in accordance with the value in step 7.1.1 |
| client_id | required | TPP's unique identifier |

The TPP's authentication takes place on the basis of a certificate used for the TLS connection

Data returned, also allowing for the *scope_details* field, containing details of consents granted by the PSU.

| PARAMETER | REQUIRED | COMMENT |
|---|---|---|
| access_token | required | |
| token_type | required | |
| expires_in | required | |
| refresh_token | optional | |
| scope | required | |
| scope_details | required | |

### 7.2.5 Consent Withdrawal

The consent withdrawal is made using the /v1.0/accounts/v1.0/deleteConsent method

### 7.2.6 Use of the scope_details structure

- The one-time consent is supported using the scopeUsageLimit parameter.

## 7.3 Authentication Mechanism in an External Authorisation Tool (Decoupled)

The basis assumption of the PSU's authentication method described is the use of EAT (*External Authorization Tool*). It is a tool the minimum functionality of which is the capacity to carry out a strong authentication of the PSU within the understanding of the technical requirements of the PSD2 Directive. Additionally, EAT may constitute software that is external in relation to the ASPSP's technical infrastructure, which ensures a safe exchange of the authorization information.

A session between the TPP and the ASPSP, allowing for the strong authentication of the PSU and based on the *Decoupled* method in order to allow the TPP to use the XS2A interface, must be established in accordance with the process as described below. The process described was developed on the basis of the assumptions of the OAuth 2.0 protocol, which means that it uses the terms defined therein (e.g. *'authorization code*', '*access token*'), but constitutes a separate way of gaining access to the XS2A interface in view of a lack of the use of redirections within the understanding of the http protocol, which are a mechanism this standard requires in the '*Authorization Code Grant*' method. This approach was applied in order to ensure a coherence of the process of gaining access to the XS2A interface, irrespective of the PSU's authentication method selected, and its objective is to facilitate integration activities related to the use of the XS2A interface by the TPP.

1. The TPP initiates a process of establishment of a session with the XS2A interface on the ASPSP's side by calling the following XS2A interface method:

   **POST /[VER_A]/auth/[VER_B]/authorizeExt**

The data sent in the request should be compliant with the XS2A interface technical specification as described in the Annex No. 1. It should be stressed that these parameters in an overwhelming majority are identical with the parameters of the request initiating a session with the XS2A interface using the ASPSP-side authentication mechanism as described in section 7.1.1. The most important parameters of that request were described in the table below.

| PARAMETER | REQUIRED | COMMENT |
|---|---|---|
| response_type | Required | Constant value: code |
| eatCode | Required | One-time authorization code generated by the EAT tool |
| client_id | Required | TPP's unique identifier |
| callbackURL | Required | Address of the callback function in the TPP's interface to which the request containing a result of the PSU's authentication will be sent |
| apiKey | | A key securing and adjusting the response to the request send in the form of a callback function<br>The key value has two functions: It constitutes a value identifying the ASPSP, on the basis of which the TPP determines whether or not the party sending the callback request is the party to which the original request was sent it allows one to match the callback request and the request sent originally by the TPP. Necessary in case of many requests sent to the ASPSP, responses to which are sent in the form of requests to the TPP's callback interface<br>The specific character of the provision of the apiKey attribute, both in the requests to the XS2A interface and to the callback interface, was described in the *swagger* format (version 2.0) in Annexes No. 1 and 2. |

| | | |
|---|---|---|
| scope_details | Required | Parameter structure described in Annex No. 1 |

In result of the calling of this method and after a positive verification by the ASPSP's of the TPP (Mutual TLS Authentication, client_id verification) and after the confirmation of non-repudiation of the message received (JWS Signature), information on the confirmation of the process of the PSU's authentication will be returned.

Notes:

The one-time authorization code, which is required as an input parameter of the authorizeExt method, must be generated by the EAT tool at the request of the PSU who has been earlier authenticated by that tool.

The PSU must previously activate access to the EAT tool in accordance with procedure developed and required by each of the ASPSPs.

The EAT tool will ensure a procedure of the PSU's strong authentication. The SCA procedure result must be send by a message to the appropriate ASPSP. The way of provision to the ASPSP of the PSU's strong authentication result, obtained in the EAT tool is not discussed in this PolishAPI specification.

The result of the PSU's strong authentication procedure must be then provided by the ASPSP to the TPP using the following TPP-side callback interface method: **[callBackURL]/[VER_A]/auth/[VER_B]/authorizeExtCallBack**

The scope of the request data was described in detail in Annex No. 2. The most important parameters of this request are:

| PARAMETER | REQUIRED | COMMENT |
|---|---|---|
| authorized | Required | Logic tag designating the result of the PSU's strong authentication by the EAT tool.<br>- true – the PSU has been authenticated<br>- false – the PSU has not been authenticated |
| code | Conditional | It is a value of the authorization code within the understanding of the OAuth 2.0 standard and the 'authorization code' method generated by the ASPSP only and exclusively in result of the PSU's authentication in the EAT tool. |

On the basis of the authorization code obtained in the previous step, the TPP should initiate an XS2A interface session by using the following method of this interface, in which one of the parameters required is an authorization code and the feedback is, among other things, the so-called 'Access token' (within the understanding of the OAuth 2.0 standard): **/[VER_A]/auth/[VER_B]/token**

The way this method is called is compliant with sec. 7.2.4, which describes the way of session initiation in the ASPSP-side PSU's authentication method. A detailed technical specification of this method is given in Annex No. 1.

## 7.4 Access token taking on the basis of the refresh token

After the access token has expired, the TPP may take a new access token using the refresh token (provided is has been issued). Such a situation will take place in case of a multiple AIS service.

Below is presented a TPP's request and a response from the ASPSP's server

| PARAMETER | REQUIRED | COMMENT |
|---|---|---|
| grant_type | required | Value of 'refresh_token' |
| refresh_token | required | In accordance with the value provided by the ASPSP in step 7.1.4 |
| scope | optional | The requested scope may not be larger than the one provided in step 7.1.4 |
| scope_details | optional | The requested scope may not be larger than the one provided in step 7.1.4 |
| is_user_session | optional | Defines whether or not the given session is related with an interaction with the PSU – true/false values. Expansion of the OAuth2 standard |
| user_ip | Required, if is_user_session=true | IP of the user's browser (information for fraud detection needs) Expansion of the OAuth2 standard |
| user_agent | Required, if is_user_session=true | Information concerning the version of the user's browser (information for fraud detection needs) Expansion of the OAuth2 standard |

The response sent by the ASPSP is the same as in item 7.2.4

## 7.5 New access token taking on the basis of the exchange token

It is a method of establishment of a communications session with the XS2A interface the purpose of which is to provide an opportunity to exchange the access token without a necessity to re-run the SCA procedure in case of a change to the scope of consents pursuant to a scenario described in sec. 1.4.4.2.3 of the specification. This scenario assumes the access is obtained to a precisely defined subset of PSU's accounts as indicated by the PSU on the basis of a list of all its accounts with the given ASPSP, which was previously obtained by the TPP on the basis of another type of the PSU's consent and after the SCA procedure was carried out.

To effect this session establishment method, it is necessary to use a dedicated authorization method indicated in the 'grant_type' attribute of the method/token with the 'exchange_token' value, and the provision in the dedicated attribute of the same name (exchange_token) of the value of access token obtained during the earlier request for the consent to take a list of accounts associated with a valid XS2A interface communications session.

Below is presented a TPP's request and a response from the ASPSP's server

| PARAMETER | REQUIRED | COMMENT |
|---|---|---|
| grant_type | required | Value of 'exchange_token' |
| exchange_token | required | Access token obtained when requesting a consent to take a list of accounts |
| scope | optional | The scope requested must be narrowed down to the list of accounts selected by the PSU and to authorizations concerning the scope of information |

| | | |
|---|---|---|
| | | requested, e.g. account details, transaction history or transaction details |
| scope_details | optional | The scope requested must be narrowed down to the list of accounts selected by the PSU and to authorizations concerning the scope of information requested, e.g. account details, transaction history or transaction details |
| is_user_session | optional | Defines whether or not the given session is related with an interaction with the PSU – true/false values. Expansion of the OAuth2 standard |
| user_ip | Required, if is_user_session=true | IP of the user's browser (information for fraud detection needs) Expansion of the OAuth2 standard |
| user_agent | Required, if is_user_session=true | Information concerning the version of the user's browser (information for fraud detection needs) Expansion of the OAuth2 standard |

The response sent by the ASPSP is the same as in item 7.2.4

# 8 Technical description of the PIS Service

This chapter constitutes a summary of the API specification in the swagger format defined in Annex No. 1 and Annex No. 2.

## 8.1 Diagram of Activity in the PIS Service

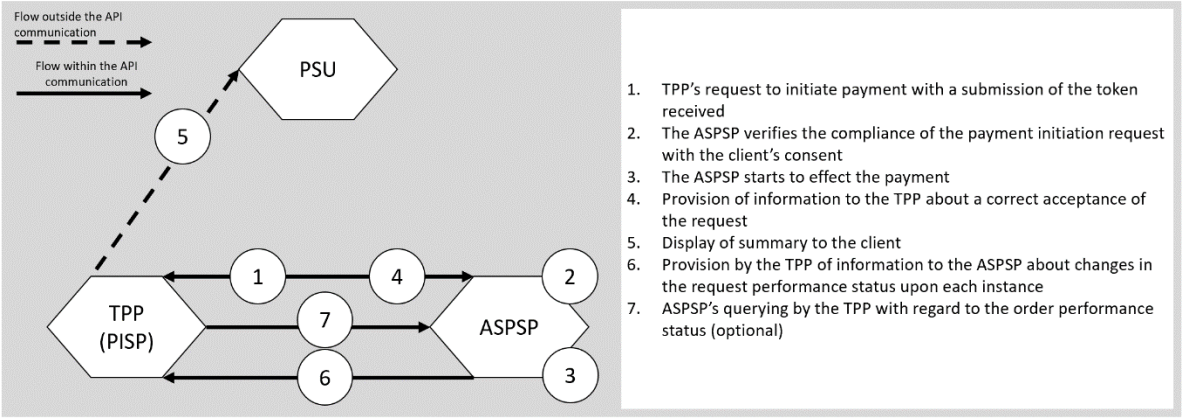



***Figure 20: High-level diagram of activity in the PIS Service***

## 8.2 XS2A Interface Request Structure

The table below contains basic information about all the PIS service methods of the XS2A interface, including the classes of objects of the cannon model of data provided in the request and obtained in the responses.

| INTERFACE METHOD | DESCRIPTION | KMD OBJECT CLASS |
|---|---|---|
| /payments/{version}/domestic | Initiates a domestic transfer | PaymentDomesticRequest/ AddPaymentResponse |
| /payments/{version}/EEA | Initiates a SEPA foreign transfer | PaymentEEARequest/ AddPaymentResponse |
| /payments/{version}/nonEEA | Initiates a non-SEPA foreign transfer | PaymentNonEEARequest / AddPaymentResponse |
| /payments/{version}/tax | initiates a tax payment | PaymentTaxRequest / AddPaymentResponse |
| /payments/{version}/getPayment | Collects the transfer status | PaymentRequest/ GetPaymentResponse |
| /payments/{version}/getMultiple Payments | Collects statuses of multiple payments. Calling does not require token reading. | PaymentsRequest/ PaymentResponse |

## 8.3 Structure of call back interface requests - CallBack

The PIS service specification comprises also a definition of CallBack interface, owing to which the ASPSP has a possibility to notify the TPP, in an asynchronous way, about changes in the status of the payment initiated using the selected PIS service method of the XS2A interface. For this purpose, a single CallBack

interface method was defined - paymentCallBack. A detailed technical specification of the CallBack interface for the PIS service was defined in Annex No. 2, therefore the table below describes only basic elements of this interface.

| INTERFACE METHOD | DESCRIPTION | KMD OBJECT CLASS |
|---|---|---|
| /payments/{version}/paymentCallBack | Provides the status of performance of a single payment | PaymentStatusInfoRequest/ CallBackResponse |

The method used to secure the API is the 'apiKey' type (https://swagger.io/docs/specification/2-0/authentication/) and, additionally, the fingerprint of the TPP's server certificate used to make the TLS connection of the CallBack - sent in the keyID parameter - is verified. In the PISs called, the TPP transfers the apiKey value and the callbackURL used in the CallBacks to the ASPSP.

In case of a failed call, the TPP may renew it and the number of repeated calls will be defined by the ASPSP in the implementation documentation.

# 9 Technical description of the AIS Service

This chapter constitutes a summary of the API specification in the swagger format defined in Annex No. 1 and Annex No. 2.

## 9.1 Diagram of Activity in the AIS Service

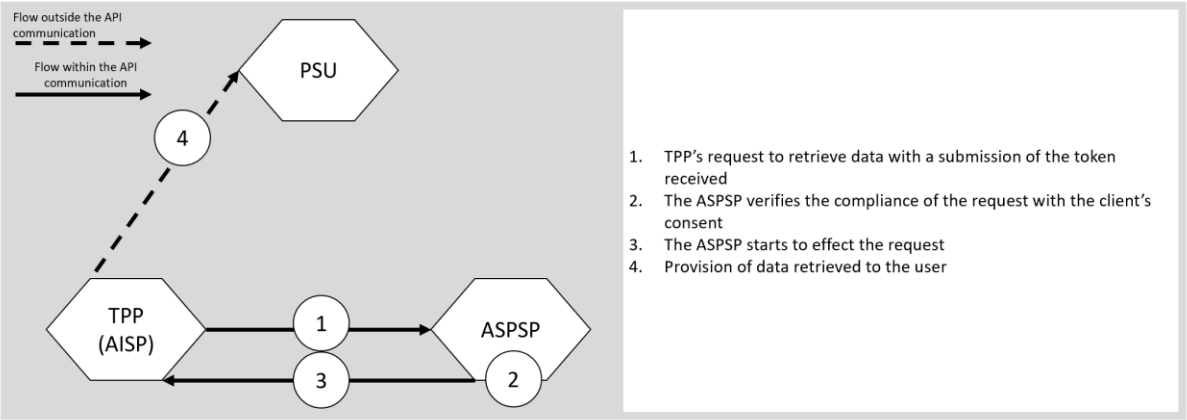



***Figure 21: High-level diagram of activity in the AIS Service***

## 9.2 XS2A Interface Request Structure

The table below contains basic information about all the AIS service methods of the XS2A interface, including the classes of objects of the cannon model of data provided in the request and obtained in the responses.

| INTERFACE METHOD | DESCRIPTION | KMD OBJECT CLASS |
|---|---|---|
| /accounts/{version}/deleteConsent | Deletes/invalidates the consent | DeleteConsentRequest/ string |
| /accounts/{version}/getAccounts | Collects all accounts of the PSU | AccountsRequest/ AccountsResponse |
| /accounts/{version}/getAccount | Collects a single payment account | AccountInfoRequest/ AccountInfo |
| /accounts/{version}/getTransactionsDone | Collects all transactions made at the account | TransactionInfoRequest/ TransactionDoneInfoResponse |
| /accounts/{version}/getTransactionsPending | Collects all pending transactions at the account | TransactionInfoRequest/ TransactionPendingInfoResponse |
| /accounts/{version}/getTransactionsRejected | Collects all rejected transactions at the account | TransactionInfoRequest/ TransactionRejectedInfoResponse |
| /accounts/{version}/getHolds | Collects all account holds | TransactionInfoRequest/ TransactionHoldInfoResponse |
| /accounts/{version}/getTransationDetail | Collects data of a single transaction/hold | TransactionDetailRequest/ TransactionDetailResponse |

## 9.3 Structure of call back interface requests - CallBack

The AIS service specification comprises also a definition of CallBack interface, owing to which the ASPSP has a possibility to provide to the TPP, in an asynchronous way, the information about the account, transactions and holds, the provision of which was requested by the TPP by calling the appropriate methods of the XS2A interface. To this end, a number of the CallBack interface methods has been defined, a detailed technical specification of which was defined in Annex No. 2. The table below describes only the basic elements of the CallBack interface for the AIS service.

| INTERFACE METHOD | DESCRIPTION | KMD OBJECT CLASS |
|---|---|---|
| /accounts/{version}/accountsCallBack | Provides information about details of the selected payment account | AccountsRequest / CallBackResponse |
| /accounts/{version}/transactionsDoneCallBack | Provides information about transactions done for the given payment account | TransactionDoneInfoRequest / CallBackResponse |
| /accounts/{version}/transactionsPendingCallBack | Provides information about pending transactions for the given payment account | TransactionPendingInfoRequest / CallBackResponse |
| /accounts/{version}/transactionsRejectedCallBack | Provides information about rejected transactions for the given payment account | TransactionRejectedInfoRequest / CallBackResponse |
| /accounts/{version}/transactionsHoldCallBack | Provides information about holds for the given payment account | TransactionHoldInfoRequest / CallBackResponse |

# 10 Technical Description of the CAF Service

This chapter constitutes a summary of the API specification in the swagger format defined in Annex No. 1.

## 10.1 Diagram of Activity in the CAF Service

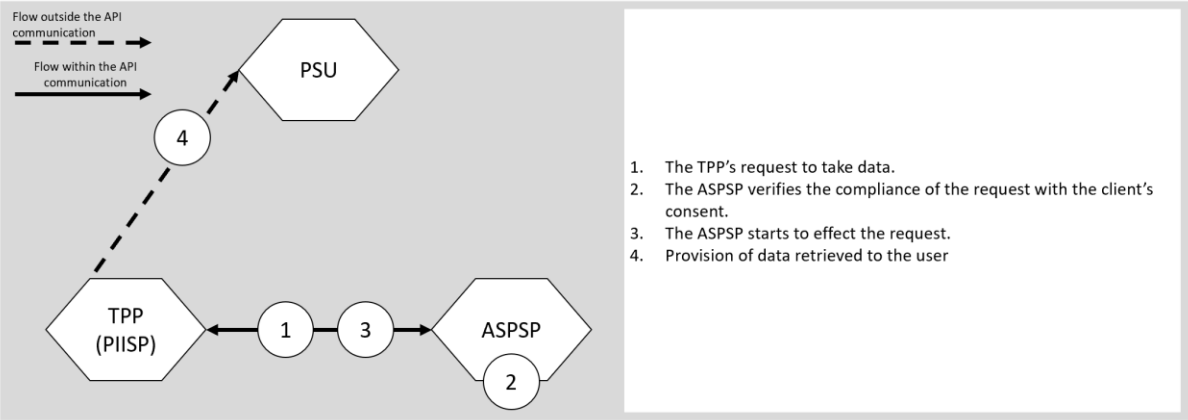



***Figure 22: High-level diagram of activity in the CAF Service***

## 10.2 XS2A Interface Request Structure (including a description of fields and information if required)

| INTERFACE METHOD | DESCRIPTION | KMD OBJECT CLASS |
|---|---|---|
| /confirmation/{version}/getConfirmationOfFunds | Confirmation of fund availability | confirmationOfFundsRequest/ confirmationOfFundsResponse |

# 11 Use of the XS2A interface methods and authorization services – sequence diagrams

The sequence diagrams presented in the UML notation describe interactions occurring between the PSU, the TPP, ASPSP-side systems and external systems which reflect the full scope of XS2A interface use scenarios, which constitutes the subject matter of the PolishAPI specification. The diagrams present only basic sequence and intervention paths leading to the achievement of the goal intended. This means that particular interactions may end in failure leasing to error messages and codes returned by the XS2A interface methods or authorization service methods and which were not included in the diagrams for picture clarity. For the same reason, particular interactions contain only some parameters of requests and responses that are important for the correctness of the sequences presented (a full specification with all the parameters of requests and responses of the XS2A interface services and the authorization services was described in the technical specification of those interfaces).

The following abbreviations, acronyms and designations were used in the diagrams:

**ASPSP Auth** – a communications interface provided by the ASPSP, pursuant to the PolishAPI specification (AS - Authorization Service), the role of which is to provide methods to authorise the TPP's access to the XS2A interface services and, in effect, establishment of sessions with that interface.

**ASPSP XS2A** – a communications interface provided by the ASPSP, the role of which is to ensure the performance of business services described in the PolishAPI specification (AIS - Account Information Service, PIS – Payment Initiation Service i CAF – Confirmation of the Availability of Funds).

**EAT** – *External Authorization Tool*, a system ensuring the SCA, i.e. a procedure of strong authentication of the PSU.

**eatCode** – a one-time code generated in the EAT tool at the PSU's request used to authorise access to the XS2A as one of the SCA procedure factors.

**sync / async** – designation of the type of communication (synchronous, asynchronous), between the actors presented in the diagram.

**tpp_redirect_url** – a URL address to which the PSU's browser should be redirected after the completion of the process of the PSU's authentication and the TPP's access authorization to the ASPSP-side resources of that PSU, in case of the *Redirection* method of the PSU's authentication.

**auth_redirect_url** – an address to which the PSU's Internet browser should be redirected. In order to authenticate the PSU using the *Redirection method.*

**authorization_code** – a one-time authorisation code which constitutes a confirmation of the TPP's authorization to access the PSU's resources.

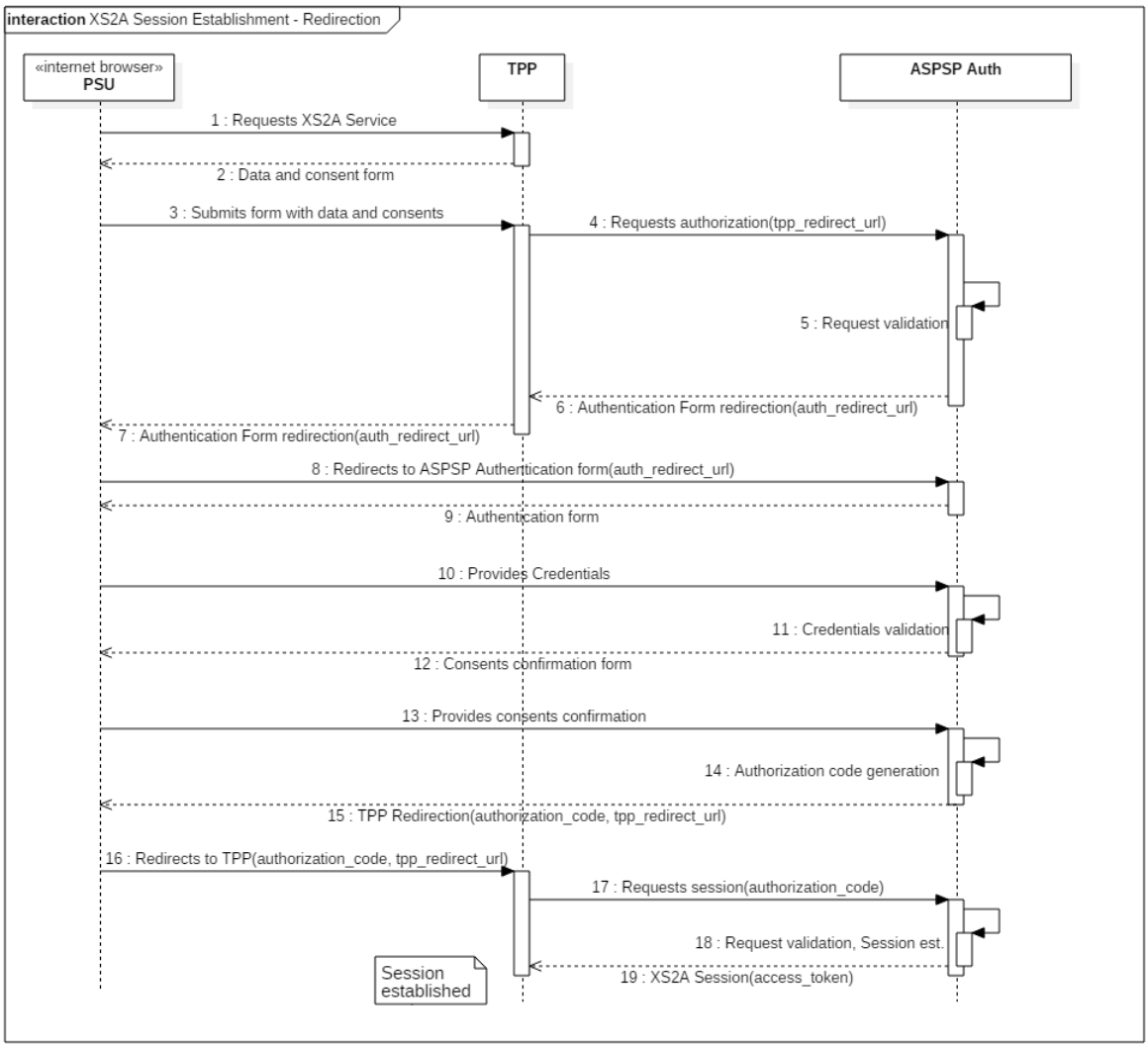
**access_token** – a token allowing access to the use of the XS2A interface services, described in more detail in sec. 5.3 Definition of Access Token.

**callback_url** – an address of the TPP-side callback interface indicating where the asynchronous responses should be sent.

**apiKey** – a type of token sent in the request in order to secure an asynchronous communication with the XS2A interface.

## 11.1 Establishment of an XS2A session with the PSU's authentication using the *redirection method*

The diagram presents a communications sequence leading to the establishment of a session with the XS2A interface, allowing for the PSU's authentication using the *redirection* method as described in Chapter 7.1 Authentication Mechanism on the ASPSP's Side




Description of interactions as per the sequence of their occurrence:

1: The PSU initiates the use of the selected XS2A interface service at the TPP's application side

2: The TPP presents a form with data required to identify the ASPSP, call an XS2A interface service and obtain access to that interface

3: The PSU inserts and confirms the data required in the TPP's form

4: The TPP requests authorisation of access to the XS2A interface by calling the following authorization service method:

**/[VER_1]/auth/[VER_2]/authorize**

One of the parameters of this method is an url address (tpp_redirect_url) redirecting to the TPP's interface after the completion of the procedure of the PSU's authentication and authorization of the TPP's access to the PSU's resources with the ASPSP.

5: The ASPSP validates the correctness of the authorization request received in terms of various aspects, including the correctness of signature, TPP's identity, compliance of the consents granted with the TPP's authorization

6: The ASPSP, in case of a positive outcome of the authorization request validation, returns a response in the form of http redirection to its own interface (auth_redirect_url) used for the PSU's authentication and authorization in the context of the request sent by the TPP

7: The TPP interprets the response from the ASPSP and returns a response to the PSU's browser in the form of a redirection to the interface of the ASPSP that obtained a response to the authorization request

8: The PSU's browser automatically redirects to the ASPSP's interface using the auth_redirect_url received

9: The ASPSP returns to the browser a page containing the PSU's authentication form

10: The PSU inserts authentication data to the form and the data, after confirmation by the PSU, are sent to the ASPSP

11: The ASPSP validates the correctness of the authentication data received as part of the SCA procedure

12: After the confirmation of the PSU's identity, the ASPSP returns to the browser a page containing a description of the scope of consents the TPP requested in order to perform the XS2A interface services, together with a form used to confirm the TPP's request

13: The PSU accepts the consents requested by the TPP by approving the form presented and by sending this information to the ASPSP

14: Having obtained the consent acceptances, the ASPSP generates and retains a one-time authorization code

15: The ASPSP returns to the browser a response in the form of redirection to the TPP's interface, i.e. to the url address of return to the TPP received in the authorization request (tpp_redirect_url), and sends the value of the one-time authorization code generated as the parameter of this response

16: The PSU's browser automatically redirects to the TPP's interface by means of the return url address received (tpp_redirect_url), together with the one-time authorization code

17: On the basis of the request received with the one-time authorization code, the TPP requests the ASPSP to start a session with the XS2A interface in the context of the authorization received from the PSU. To this end, the following authorization service method is called and one of the required elements of this method is a one-time authorization code (authorization_code):

**/[VER_1]/auth/[VER_2]/token**

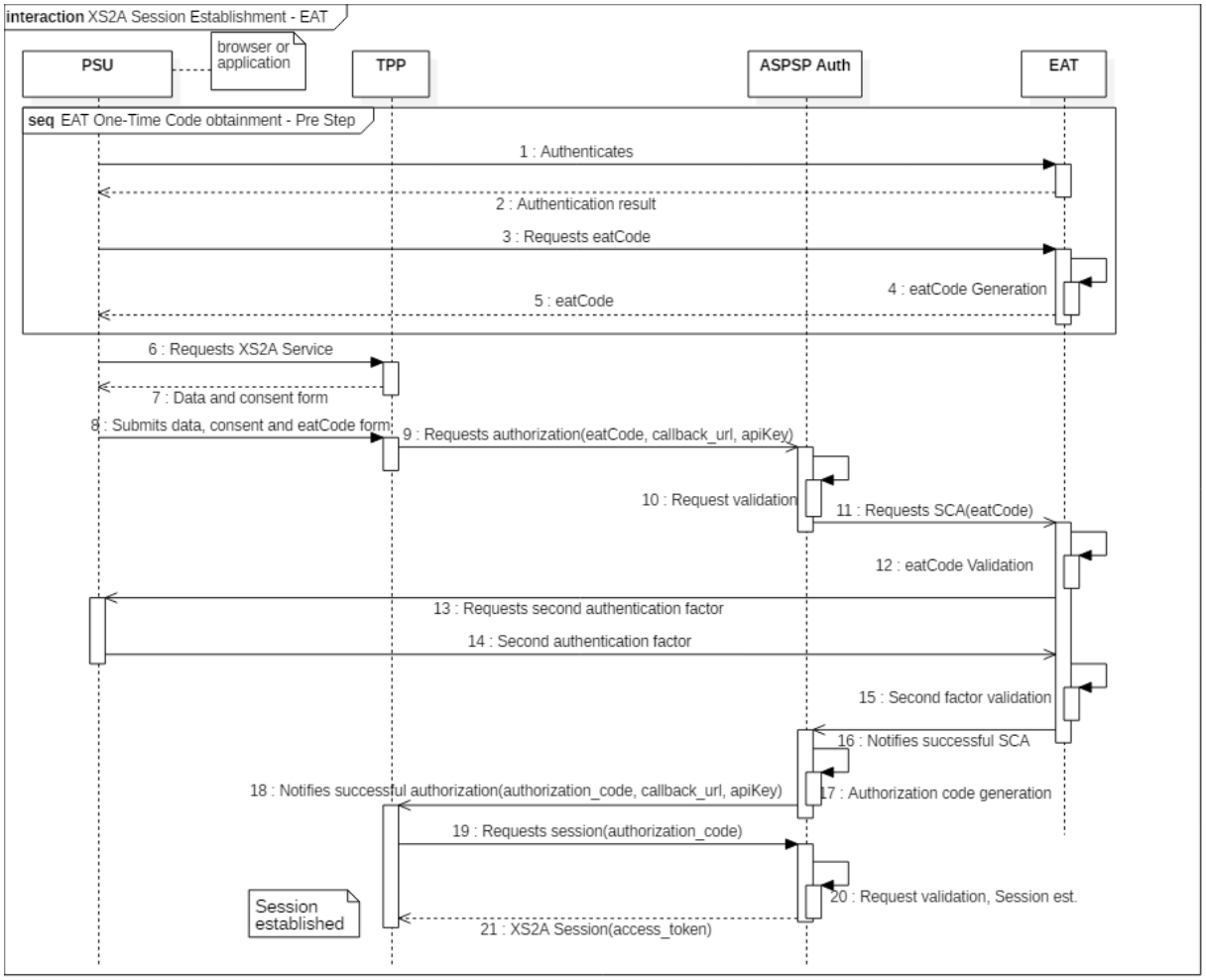
18: The ASPSP validates the XS2A session establishment request received by verification of the authorization code received (authorization_code) and the data concerning the PSU's consents

granted. After a positive verification, the ASPSP establishes a new session of the XS2A interface in result of which a unique access token (access_token) is generated.

19: The ASPSP returns a response to the session establishment request to the TPP, which contains, without limitation, the value of the access token generated, confirming thus the establishment of a session with the XS2A interface

## 11.2 Establishment of an XS2A session with the PSU's authentication using the *decoupled method*

The diagram presents a communications sequence leading to the establishment of a session with the XS2A interface, allowing for the PSU's authentication using the *decouple* method as described in Chapter 7.2 Authentication Mechanism in an External Authorisation Tool (Decoupled)




Description of interactions as per the sequence of their occurrence:

1: Via a browser or an application, the PSU sends the data to be authenticated in the EAT tool

2: The EAT tool verifies the authentication data and grants access to its interface to the PSU

3: The PSU requests that a one-time code be issued (eatCode)

4: The EAT tool generates a one-time code (eatCode)

5: The EAT tool returns the one-time code to the PSU's browser or application

6: The PSU initiates the use of the selected XS2A interface service at the TPP's application side

7: The TPP presents a form with data required to identify the ASPSP, call an XS2A interface service and obtain access to that interface (including, but not limited to, an insertion of the eatCode value)

8: The PSU inserts and confirms the data required in the TPP's form

9: The TPP requests authorisation of access to the XS2A interface by calling the following authorization service method:

**/[VER_1]/auth/[VER_2]/authorizeExt**

Due to the asynchronous character of the response to the request, among the parameters of the method, it is required to provide a url address (callback_url) of the XS2A callback interface and a security token (apiKey). Furthermore, in order to obtain an authorization, it is required to provide the one-time code received from the EAT tool (eatCode).

10: The ASPSP validates the correctness of the authorization request received in terms of various aspects, including the correctness of signature, TPP's identity, compliance of the consents granted with the TPP's authorization

11: The ASPSP sends a request to the EAT tool in order to carry out the SCA procedure in relation with the PSU, including a verification of the correctness of the one-time code (eatCode) received from the PSU and generated by EAT

12: EAT verifies the correctness of the one-time code (eatCode) received from the ASPSP

13: In case, the one-time code is correct, the EAT tool requests that the PSU provided a second factor in order to complete the SCA procedure

14: The PSU executes a second factor in the EAT tool

15: The EAT tool verifies the correctness of the second factor provided by the PSU

16: The EAT notifies the ASPSP about the result of the SCA procedure carried out

17: In case of a positive result of the strong authentication of the PSU and the TPP's authorization to access the PSU's resources (including consents obtained from the PSU), the ASPSP generates and retains a one-time authorization code (authorization_code)

18: The ASPSP notifies the TPP about the result of the request to authorize access to the PSU's resources by calling the following TPP-side callback interface method:

**/[VER_1]/auth/[VER_2]/authorizeExtCallBack**

In case the TPP has obtained authorization to access the PSU's resources, the request contains a one-time authorization code (authorization_code).

19: On the basis of the request received with the one-time authorization code, the TPP requests the ASPSP to start a session with the XS2A interface in the context of the authorization received from the PSU. To this end, the following authorization service method is called and one of the required elements of this method is a one-time authorization code (authorization_code):

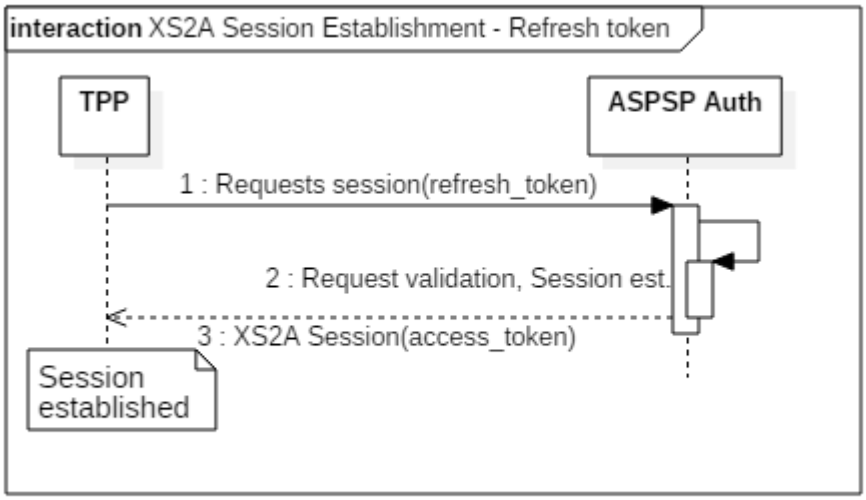
**/[VER_1]/auth/[VER_2]/token**

20: The ASPSP validates the XS2A session establishment request received by verification of the authorization code received (authorization_code) and the data concerning the PSU's consents granted. After a positive verification, the ASPSP establishes a new session of the XS2A interface in result of which a unique access token (access_token) is generated.

21: The ASPSP returns a response to the session establishment request to the TPP, which contains, without limitation, the value of the access token generated, confirming thus the establishment of a session with the XS2A interface

## 11.3 Establishment of an XS2A session with the PSU's authentication using the *refresh token method*

The diagram presents a communications sequence leading to the establishment of a session with the XS2A interface, using the refresh token method.




Description of interactions as per the sequence of their occurrence:

1: The TPP sends a request to the ASPSP to start a session with the XS2A interface in the context of the session established earlier which was cancelled and which was related to an additional token (refresh_token), returned to the TPP during the original session establishment procedure that was cancelled. To this end, the TPP calls the following authorization service method, sending an additional token (refresh_token):

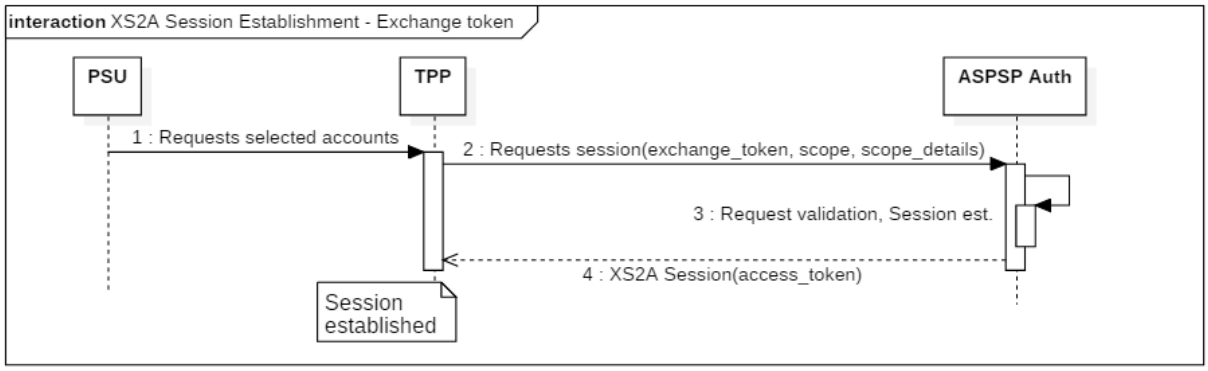
**/[VER_1]/auth/[VER_2]/token**

2: The ASPSP validates the XS2A session establishment request received by verification of the additional token received (refresh_token) and the data concerning the PSU's consents granted. After a positive verification, the ASPSP establishes a new session of the XS2A interface in result of which a unique access token (access_token) is generated.

3: The ASPSP returns a response to the session establishment request to the TPP, which contains, without limitation, the value of the access token generated, renewing thus the session with the XS2A interface

## 11.4 Establishment of an XS2A session with the PSU's authentication using the *exchange token method*

The diagram presents a communications sequence leading to the establishment of a session with the XS2A interface, using the exchange token method.




Description of interactions as per the sequence of their occurrence:

1: The PSU specifies the details of consents for the TPP by selecting a subset of accounts from the accounts previously retrieved by the TPP from the ASPSP, and by determining the scope of authorizations to the data related to those accounts, such as account details, account history and the time scope or transaction details.

2: The TPP sends a request to the ASPSP to start a session with the XS2A interface in the context of a previously established session that was established in order to retrieve a list of PSU's accounts and to which the access token (access_token) returned to the TPP during the original session establishment procedure with a strong authorization of the PSU is related. To this end, the TPP calls the following authorization service method, providing the said access token and using the exchange_token parameter:

**/[VER_1]/auth/[VER_2]/token**

The required parameters of that request are also the *scope* and *scope_details* parameters, which must contain a detailed scope of consents, including the numbers of accounts selected by the PSU.

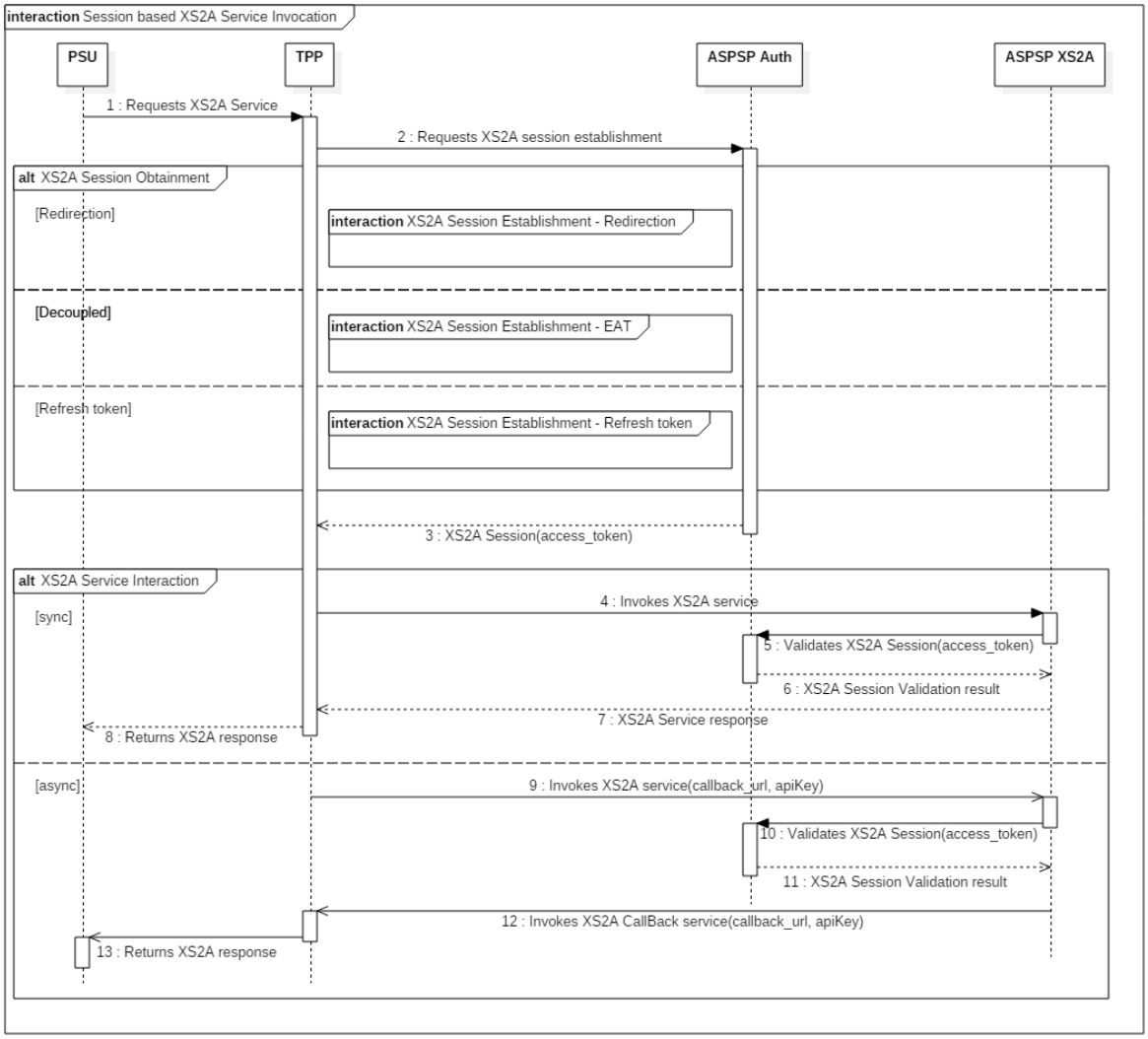
3: : The ASPSP validates the XS2A session establishment request received by verification of the access token received (provided in the exchange_token attribute) and the data concerning the PSU's consents granted. After a positive verification, the ASPSP establishes a new session of the XS2A interface in result of which a unique new access token (access_token) is generated in the context of the new scope of consents.

4: The ASPSP returns a response to the session establishment request to the TPP, which contains, without limitation, the value of the access token generated, confirming thus the establishment of a new session with the XS2A interface.

## 11.5 XS2A Interface Method Calling with the Use of a Session

The diagram presents a communication sequence allowing one to call the XS2A interface services for which a valid session of that interface is required. The table below contains a list of methods within the framework of the AIS and PIS services, for which the sequence presented is obligatory.

| **AIS** |
|---|
| /[VER_1]/accounts/[VER_2]/ getAccounts |
| /[VER_1]/accounts/[VER_2]/ getAccount |
| /[VER_1]/accounts/[VER_2]/ getTransactionsDone |
| /[VER_1]/accounts/[VER_2]/ getTransactionsPending |
| /[VER_1]/accounts/[VER_2]/ getTransactionsRejected |
| /[VER_1]/accounts/[VER_2]/ getHolds |
| /[VER_1]/accounts/[VER_2]/ getTransactionDetail |
| **PIS** |
| /[VER_1]/payments/[VER_2]/ domestic |
| /[VER_1]/payments/[VER_2]/ EEA |
| /[VER_1]/payments/[VER_2]/ nonEEA |
| /[VER_1]/payments/[VER_2]/ tax |
| /[VER_1]/payments/[VER_2]/ getPayment |




Description of interactions as per the sequence of their occurrence:

1: The PSU initiates the use of the selected XS2A interface service at the TPP's application side

2: The TPP requests that a session with the XS2A interface be established. The session establishment procedure may be carried out on the basis of each one of the variants available, for which the sequence diagrams were described in the previous items of this chapter.

3: The ASPSP returns a response to the session establishment request to the TPP, which contains, without limitation, the value of the access token generated, confirming thus the establishment of a session with the XS2A interface

**Variant 1 – synchronous services of the XS2A interface**

4: The TPP sends a request to the XS2A interface in order to use a service of this interface selected by the PSU (one of the methods listed in the table above). In the request parameters, the input data required to perform the service and an access token (access_token) are provided in order to verify the obtained authorization to use the service.

5: The ASPSP validates the correctness and validity of the access token obtained (access_token) by means of an authorization service communication.

6: The ASPSP receives the access token validation result

7: In case of a positive result of the access token validation, the ASPSP returns the outcome of the XS2A service performance in the form of a response to the request sent by the TPP to the XS2A interface.

8: The TPP presents a result of the XS2A interface service performance to the PSU

**Variant 2 – asynchronous services of the XS2A interface**

9: The TPP sends a request to the XS2A interface in order to use a service of this interface selected by the PSU (one of the methods listed in the table above). The request parameters include input data - required to perform the service, an access token (access_token) - required to verify the obtained authorization to perform the service, , and values of the callback_url and apiKey parameters - required to send a response to the request in the form of a request to the TPP-side callback interface.

10: The ASPSP validates the correctness and validity of the access token obtained (access_token) by means of an authorization service communication.

11: The ASPSP receives the access token validation result

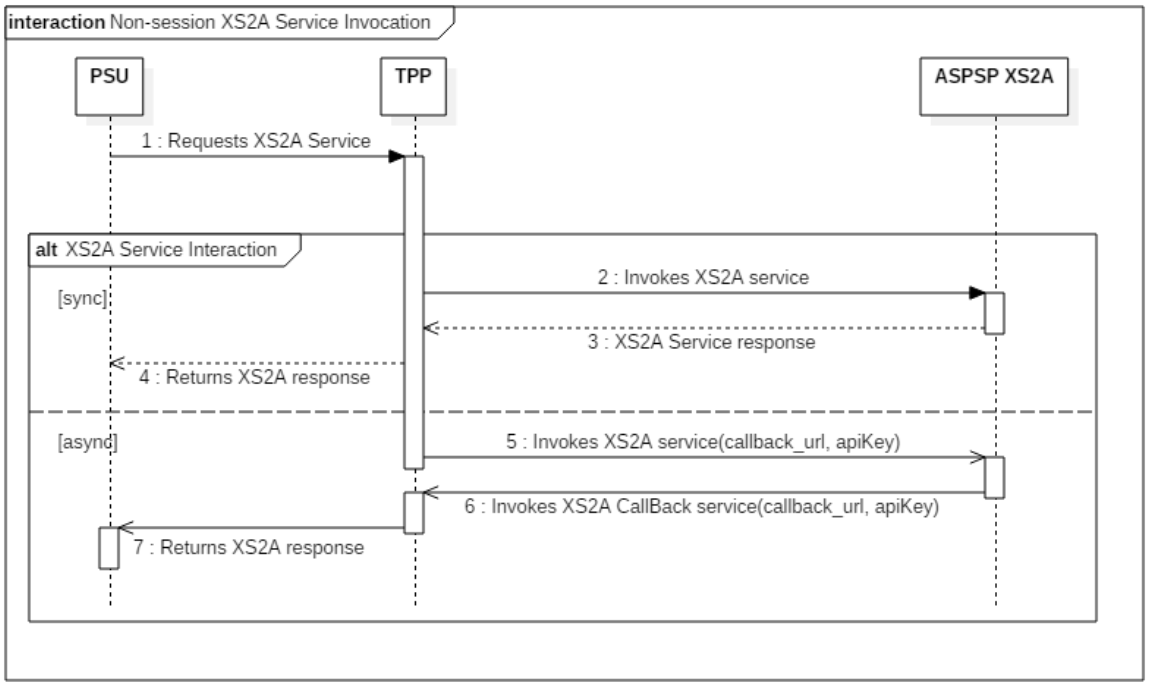
12: In case of a positive result of the access token validation, the ASPSP returns the outcome of the XS2A service performance by sending a request to the callback interface of the TPP-side XS2A interface (to the address indicated in callback_url).

13: The TPP presents a result of the XS2A interface service performance to the PSU

## 11.6 XS2A Interface Method Calling without the Use of a Session

The diagram presents a communication sequence allowing one to call the XS2A interface services for which a valid session of that interface is not required. The table below contains a list of methods within the framework of the AIS, PIS and CAF services, for which the sequence presented is obligatory.

| **AIS** |
|---|
| /[VER_1]/accounts/[VER_2]/deleteConsent |
| **PIS** |
| /[VER_1]/payments/[VER_2]/getMultiplePayments |
| **CAF** |
| /[VER_1]/confirmation/[VER_2]/getConfirmationOfFunds |




Description of interactions as per the sequence of their occurrence:

1: The PSU initiates the use of the selected XS2A interface service at the TPP's application side. This service does not require any session at the XS2A interface's side. The services of this type were listed in the table above.

**Variant 1 – synchronous services of the XS2A interface**

2: The TPP sends a request to the XS2A interface in order to use a service of this interface selected by the PSU. The request parameters contain input data required to perform the service.

3: The ASPSP returns the outcome of the XS2A service performance in the form of a response to the request sent by the TPP to the XS2A interface.

4: The TPP presents a result of the XS2A interface service performance to the PSU

**Variant 2 – asynchronous services of the XS2A interface**

5: The TPP sends a request to the XS2A interface in order to use a service of this interface selected by the PSU. The request parameters include input data - required to perform the service, and values of the callback_url and apiKey parameters - required to send a response to the request in the form of a request to the TPP-side callback interface.

6: The ASPSP returns the outcome of the XS2A service performance by sending a request to the callback interface of the TPP-side XS2A interface (to the address indicated in callback_url).

7: The TPP presents a result of the XS2A interface service performance to the PSU

# 12 Error codes

| STAGE | ERROR | HTTP CODE | HOW SUPPORTED |
|---|---|---|---|
| Redirection of the client to the ASPSP's domain together with a provision in the redirection address of parameters of the initiated payment | Provision of incorrect parameters | 400 | Reverse redirection of the client to the TPP's domain and process abortion (pursuant to RFC6749 4.1.2.1) |
| The ASPSP verifies the TPP's identity in the Identity Hub and verifies whether or not the given TPP may request the consents indicated (AIS or PIS) | Incorrect verification of the TPP's identity | 401 | Display of a message to the user |
| | Incorrect verification of the TPP's licence (e.g. only AIS) | 403 | Reverse redirection of the client to the TPP's domain and process abortion (pursuant to RFC6749 4.1.2.1) |
| The client logs in the ASPSP's domain | Incorrect logging - 1 time | 401 | Waiting for a correct client logging |
| | Multiple incorrect logging | 401 | Waiting for a correct client logging |
| | No service activation by the TPP (opt-out) | 403 | Reverse redirection of the client to the TPP's domain and process abortion (pursuant to RFC6749 4.1.2.1) |
| | No funds | 422 | Reverse redirection of the client to the TPP's domain and process abortion (appropriate error code returned when requesting a payment service) |
| The client expresses the consent in the requested or limited scope or else rejects the request | Request rejection by the client | 403 | Reverse redirection of the client to the TPP's domain and process abortion (pursuant to RFC6749 4.1.2.1) |
| | Incorrect authorisation by the client | 403 | Reverse redirection of the client to the TPP's domain and process abortion (pursuant to RFC6749 4.1.2.1) |
| Reverse redirection of the client to the TPP's domain | The client's end device shutdown before the redirection | - | The TPP has an option to complete the operation by taking the token using contextId |
| Authorisation token taking from the ASPSP by the TPP | Unknown TPP | 401 | Process abortion without the request being admitted |
| | Incorrect token request parameters | 400 | Process abortion without the request being admitted |
| | Communication with the ASPSP impossible | - | Process abortion without the request being admitted |

| | | | |
|---|---|---|---|
| The ASPSP verifies the compliance of the payment initiation request with the client's consent | No compliance with the consent granted | 401 | Process abortion without the request being admitted |
| | Limit of requests for the desired service has been exceeded | 429 | Process abortion without the request being admitted |
| The ASPSP starts to effect the payment | No funds | 422 | End of process (error code returned) |
| Provision of information to the TPP about a correct acceptance of the request | Communication with the TPP impossible | - | Message renewal x3 Failure information Possibility to repeat all messages |
| Provision by the TPP of information to the ASPSP about changes in the request performance status | Communication with the TPP impossible | - | Message renewal x3 Failure information Possibility to repeat all messages |
| The ASPSP verifies the compliance of the payment account data request with the client's consent | No compliance with the consent granted | 403 | End of process |
| ASPSP receives a request | Access to the service blocked by the ASPSP | 403 | End of process |
| ASPSP receives a request | Access to the service blocked by the PSU | 403 | End of process |

## 12.1 Error codes for the HTTP 403 response code

| CODE | COMMUNICATION |
|---|---|
| 1 | Incorrect verification of TPP licenses |
| 2 | No activation of TPP services |
| 3 | Rejection of the request by the client |
| 4 | Incorrect authorization by the client |
| 5 | Non-compliance with the consent given |
| 6 | Access to the service blocked by ASPSP |
| 7 | Access to the website blocked by the PSU |
| 8 | Blocked TPP's access to bank services by the bank |
| 9 | Blocked TPP's access to bank services by the client |

# 13 Standard Implementation Recommendations

## 13.1 Timeout Support

Due to the timeout type events which may occur during the http request processing, the ASPSP must ensure uniqueness verification at the server layer at the requestId level. Having identified a non-uniqueness of the request, the ASPSP returns the 400.1 error (Request repeated).

The recommended timeout value is 30 seconds.

## 13.2 TPP verification

The TPP authentication should be made based on the communications certificate (tls) and signature certificate (JSON Web Signature), with a simultaneous verification whether or not the certificates correspond to the TPP's ID (tppId) in the ASPSP's database. The tppId value is determined by the ASPSP during the TPP technical registration; the suggested value is TPP's EUNIP .

## 13.3 Authorization server

It is recommended that in its configuration of the given client_id, the ASPSP should have a list of redirect_uri which may be used. Thus, the ASPSP will not redirect the client to an incorrect URL address which may be submitted by an untrusted party.

## 13.4 Fraud Prevention

In order to prevent potential frauds, a dedicated RequestHeader Class was implemented which is provided in each request and contains the following information about the PSU: IP address and userAgent. The structure will be useful for the ASPSP during the implementation of security mechanisms.

Additionally, it is recommended that the entities participating in the project exchanged information about suspected unauthorised transactions/compromised IPs etc.

# 14 List of Annexes

Annex No. 1: PolishAPI-ver2.0.yaml

Annex No. 2: PolishAPI-CallBack-ver2.0.yaml