SLOVAK BANKING ASSOCIATION

{ Slovak Banking API Standard }

Version: 1.0

Bratislava, 20.11. 2017

1. License

Slovak banking association grants to users of Slovak banking API standard a non-exclusive, royalty free, worldwide copyright license to reproduce, prepare derivative works from, distribute, perform and display, this Slovak banking API standard solely for the purposes of developing and implementing relevant applications and APIs.

Provided that attribution be made to Slovak banking association as the source of the material but that such attribution does not indicate an endorsement by Slovak banking association.

2. Responsibility

Permission is hereby granted to use the document solely for the purpose of implementing the Slovak banking API standard subject to the following conditions: (i) that Slovak banking association nor any contributor to the Slovak banking API standard shall have any responsibility or liability whatsoever to any other party from the use or publication of the Slovak banking API standard; (ii) that one cannot rely on the accuracy or finality of the Slovak banking API standard; and (iii) that the willingness of Slovak banking association to provide the Slovak banking API standard does not in any way convey or imply any a responsibility for any product or service developed in accordance with the Slovak banking API standard and Slovak banking association as well as the contributors to the Slovak banking API standard specifically disclaim any such responsibility to any party. Implementation of certain elements of this Slovak banking API standard may require licenses under third party intellectual property rights, including without limitation, patent rights. Slovak banking association and any other contributors to the Slovak banking API standard are not, and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

If asked by the entitled users, the Slovak banking association will set out the names of the relevant third parties participating on Slovak banking API standard, which intellectual property rights could be affected by the implementation of certain elements of Slovak banking API standard.

This Slovak banking API standard is provided "as is", "where is" and "with all faults", and Slovak banking association does not makes any warranty of any kind, express or implied, including any implied warranties of merchantability, non-infringement of third party intellectual property rights (whether or not third parties have been advised, have reason to know, or are otherwise in fact aware of any information), and fitness for a particular purpose (including any errors and omissions in the Slovak banking API standard). To the extent permitted by applicable law, neither Slovak banking association nor any contributor to the Slovak banking API standard shall be liable to any user of the Slovak banking API standard for any damages (other than direct actual out-of-pocket damages) under any theory of law, including, without limitation, any special damages, nor any damages for loss of business profits, business interruption, loss of business information, or other monetary loss, nor any damages arising out of third party claims (including claims of intellectual property infringement) arising out of the use of or inability to use the Slovak banking API standard, even if advised of the possibility of such damages. Slovak banking association does not imply either endorsement of any of the solutions identified in this Slovak banking API standard or a commitment to implement them.

Any contributor of comments or feedback to the Slovak banking API standard does so in the knowledge that the specifications are open data and no rights or interest shall arise as a result of such contributions.

3. Entry into force

This document enters into force on the date specified in the Directive (EU) 2015/2366 of the European Parliament and of the Council on payment services in the internal market.

Table of Contents

1 Terminology

For the purposes of this document, the following terms have the following meanings:

Term	Meaning
AISP	Account Information Service Provider.
Alternative implementation	The ASPSP is required to implement at least one of the alternatives.
ASPSP	Account Servicing Payment Service Provider.
Authentication	TPP Identity confirmation.
Authorization	Verification of access to ASPSP resources.
Certificate	After the SCA RTS has been applied, it means a qualified certificate in the sense of e-IDAS.
Directive	PSD2 Directive. Directive of the European Parliament and of the Council (EU) 2015/2366.
EV	Extended Validation certificate
IBAN	International Bank Account Number.
JOSE	JSON Object Signing and Encryption.
OIDC	OpenID Connect
Optional implementation	The ASPSP may implement this functionality or process.
Optional input parameter	TPP can ignore this parameter.
Optional output Parameter	The ASPSP may fill the parameter value.
PIISP	Payment Instrument Issuer Service Provider
PISP	Payment Initiation Service Provider.
PSU	Payment Service User.
Resource	All access points of the ASPSP API for TPP access within PSD2.
RTS	Regulatory technical standards of the Europan Banking Authority
SBA	Slovak banking association.
SCA	Strong Customer Authentication. Authentication of a payment service user means authentication based on the use of two or more elements that are categorized as knowledge (something the user knows only), ownership (something that only the user has), and inherence (something, the user is) and are independent in the sense that the violation of one element does not impair the reliability of the other elements, while being created in such a way as to protect the confidentiality of the authentication data.
The Slovak Banking API Standard	Common initiative of Slovak banking association and its members. The aim of this initiative is to develop common specifications for the communication interface between ASPSPs and third party providers within the meaning of Directive (EU) 2015/2366.
TPP	Third Party Provider, i.e., a third party that is a payment service provider providing payment service users with a payment initiation or account information service or a payment service provider issuing card based payment facilities.

All HTTP requests in the examples are labeled with the number given in the individual data flow diagrams.

2 Table of Figures

3 Introduction

This document defines secure communication between the TPP and the ASPSP and between the PSU and the ASPSP, in particular to ensure the integrity of the transmitted data and the identity of the communicating entities.

The document does not describe the process of strong authentication of the ASPSP's (PSU) customer's payment service user (SCA) with the ASPSP itself. The SCA process drawn in the process flow of the individual processes diagrams serves for demonstration purposes and a better understanding of process flow. The SCA process is not part of this standard.

List of services described by the standard:

Service Provider	Service	Optionality	Description
AISP	**Accounts Information**	Mandatory	**Account information** – service provide information and balances related to an account
AISP	**Accounts Transactions**	Mandatory	**Account transactions** – service provide list of transactions in defined date range related to an account
AISP	**Accounts List**	Optional	**List of accounts** - service returns the list of accounts to which the client has given a consent to specific TPP (not a list of all client accounts) without balances
PISP	**Standard Payment Initialization (XML)**	Mandatory	**Standard payment initialization** – service allows to initialize payment in XML format (PAIN.001)
PISP	**Standard Payment Submission**	Mandatory	**Standard payment submission** – service allows to authorization of initialized payment
PISP	**Payment Order Status**	Mandatory	**Payment order status** – service provide actual information about initialized payment
PISP	**Standard payment initialization (JSON)**	Optional	**Standard payment initialization** – service allows to initialize payment in JSON
PISP	**Ecommerce payment initialization (XML)**	Optional	**Ecommerce payment initialization –** service allows to initialize immediate payment in XML format (PAIN.001)
PISP	**Ecommerce payment initialization (JSON)**	Optional	**Ecommerce payment initialization** – service allows initialize immediate payment in JSON format
PIISP	**Balance check**	Mandatory	**Balance check** – service provide information about sufficient balance with the yes/no answer

4 Common Design

4.1 Recommended form of ASPSP web services extension

a) The Slovak Banking API Standard (hereinafter referred as "standard") represents only minimum requirements for API implementation. In general the standard is voluntary for SBA members (banks or ASPSP).

b) The standard is a kind of binding for members which have joined it. It means that the ASPSPs must implement API service operations that are mentioned in this document as mandatory and may implement API service operations which are mentioned as optional.

c) An ASPSP may extend its provided web service by publishing its new service operation on a new endpoint denoted with the base path "/api/extend/v1".

- The standard uses the base path "/api/v1".
- The versions parts in both base paths are independent from each other, so that the extended API can have different version than the standard has, e.g., /api/extend/v2.
- The standard uses the semantic versioning [13] and in its base path the major part of the entire version. This approach should follow an extended API as well.

4.1.1 Design principles for APIs

The standard adheres to following list of principles and rules, to which an extended APIs of ASPSPs should adhere as well.

a) Every mandatory service operation is related just to one customer's bank account. None of the service operations can provide response for a bulk of accounts.

b) An account identifier, especially IBAN, should not appear in an internet address of a service operation. It should be located in the body of a HTTP request, or at least in a HTTP header field.

- This principle ensures that for instance IBAN as a sensitive data item cannot be used neither as a path template parameter nor as a query parameters of a service operation.

c) The HTTP method GET cannot be used with a message body with semantic meaning in order to follow the HTTP specification [14]

- Preferably the message body should be empty.
- If needed, the message body can have non-empty content, however, it can contain data without any semantic in relation to request as a whole. Such content can be used for analytical or statistical purposes.
- If a message body with semantically relevant content is required, the HTTP method POST should be used at least.

d) The data model of the standard and all extended APIs should utilized data elements, terms, and semantics from ISO 20022 [15] as much as reasonable.

e) The semantic messaging is prohibited. Particular data element of the API data model should have always the same semantics regardless of the context it is used in.

- This principle ensures that the meaning of a data element does not depend on combination of values of other data elements, not on its place in the data model or service operation address parameters, and so on.

f) The only semantic versioning [13] is allowed. If whatever information object in the API needs to be versioned (usually base path), the semantic versioning scheme should be used.
   - Preferably all part of semantic version scheme should be used, e.g., 1.2.3.
   - In some context just the major part (the very first one) of the version scheme may be used, e.g., v1.
   - Usage of major and minor version of the version scheme without patch part is prohibited. Instead, the patch value 0 should be used at least, e.g., 1.3.0.

g) Naming convention for data elements.
   - The names of data elements in service operation parameters and in the data model should be in lower camel case.
   - The data element starts with a meaningful word in lower case followed by words with the first capital letters, e.g., accountNumber.
   - Non-alphabetic characters should not be used as word delimiters in data elements names, e.g., account_number.

h) For better reuse or sharing data definitions or semantics it is preferable to create custom data types in the API data model as much as possible and reasonable.
   - Custom data types should be referenced at particular data elements in the data model as their definitions.

i) Naming convention for custom data types.
   - The names of custom data types in the data model should be in upper camel case (a.k.a. Pascal case)
   - The custom data type starts with a meaningful word with the first capital letter followed by words with the first capital letters, e.g., AccountNumberType.
   - Non-alphabetic characters should not be used as word delimiters in custom data types names, e.g., account_number_type.
   - The name of a custom data type should always ends with the word "Type" regardless the adjacent previous word is "Type" as well, e.g., AccountTypeType, AccountNumberType.
   - The name of a custom data type should be in singular even if it denotes a collection (array). The custom data type can be considered as a class name.

4.2 Securing communication

A TLS version 1.2+ is required to secure the communication layer. In order to reduce the vulnerability of block ciphers, only AEAD (Authenticated Encryption with Additional Data) is allowed, specifically:

AES_GCM (128,256)

NIST	OpenSSL equivalent
TLS_RSA_WITH_AES_128_GCM_SHA256	AES128-GCM-SHA256
TLS-RSA-WITH-AES-256-GCM-SHA384	AES256-GCM-SHA384
TLS-RSA-WITH-CAMELLIA-128-GCM-SHA256	NA
TLS-RSA-WITH-CAMELLIA-256-GCM-SHA384	NA
TLS-DHE-RSA-WITH-AES-128-GCM-SHA256	DHE-RSA-AES128-GCM-SHA256
TLS-DHE-RSA-WITH-AES-256-GCM-SHA384	DHE-RSA-AES256-GCM-SHA384
TLS-DHE-RSA-WITH-CAMELLIA-128-GCM-SHA256	NA
TLS-DHE-RSA-WITH-CAMELLIA-256-GCM-SHA256	NA
TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	ECDHE-RSA-AES128-GCM-SHA256
TLS-ECDHE-RSA-WITH-AES-256-GCM-SHA384	ECDHE-RSA-AES256-GCM-SHA384
TLS-ECDH-RSA-WITH-AES-128-GCM-SHA256	ECDH-RSA-AES128-GCM-SHA256
TLS-ECDH-RSA-WITH-AES-256-GCM-SHA384	ECDH-RSA-AES256-GCM-SHA384
TLS-ECDH-RSA-WITH-CAMELLIA-128-GCM-SHA256	NA
TLS-ECDH-RSA-WITH-CAMELLIA-256-GCM-SHA384	NA
TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256	ECDHE-ECDSA-AES128-GCM-SHA256
TLS-ECDHE-ECDSA-WITH-AES-256-GCM-SHA384	ECDHE-ECDSA-AES256-GCM-SHA384
TLS-ECDHE-ECDSA-WITH-CAMELLIA-128-GCM-SHA256	NA
TLS-ECDHE-ECDSA-WITH-CAMELLIA-256-GCM-SHA384	NA
TLS-ECDH-ECDSA-WITH-AES-128-GCM-SHA256	ECDH-ECDSA-AES128-GCM-SHA256
TLS-ECDH-ECDSA-WITH-AES-256-GCM-SHA384	ECDH-ECDSA-AES256-GCM-SHA384
TLS-ECDH-ECDSA-WITH-CAMELLIA-128-GCM-SHA256	NA
TLS-ECDH-ECDSA-WITH-CAMELLIA-256-GCM-SHA384	NA

AES_CCM (128,256)

NIST	OpenSSL equivalent
TLS-RSA-WITH-AES-128-CCM	AES128-CCM
TLS-RSA-WITH-AES-256-CCM	AES256-CCM
TLS-RSA-WITH-AES-128-CCM-8	AES128-CCM8
TLS-RSA-WITH-AES-256-CCM-8	AES256-CCM8
TLS-DHE-RSA-WITH-AES-128-CCM	DHE-RSA-AES128-CCM
TLS-DHE-RSA-WITH-AES-256-CCM	DHE-RSA-AES256-CCM
TLS-DHE-RSA-WITH-AES-128-CCM-8	DHE-RSA-AES128-CCM8
TLS-DHE-RSA-WITH-AES-256-CCM-8	DHE-RSA-AES256-CCM8
TLS-ECDHE-ECDSA-WITH-AES-128-CCM	ECDHE-ECDSA-AES128-CCM
TLS-ECDHE-ECDSA-WITH-AES-256-CCM	ECDHE-ECDSA-AES256-CCM
TLS-ECDHE-ECDSA-WITH-AES-128-CCM-8	ECDHE-ECDSA-AES128-CCM8
TLS-ECDHE-ECDSA-WITH-AES-256-CCM-8	ECDHE-ECDSA-AES256-CCM8

CHACHA20_POLY1305

NIST	OpenSSL equivalent
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	ECDHE-RSA-CHACHA20-POLY1305
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	ECDHE-ECDSA-CHACHA20-POLY1305
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	DHE-RSA-CHACHA20-POLY1305

4.3 TPP and ASPSP authentication

For the authentication of the ASPSP as a resource provider, the eIDAS-based site authentication certificate will be used or EV certificate. For the authentication of the TPP as a client, the eIDAS-based site authentication certificate will be used or EV certificate.

Because under the EVCG EV Certificate conditions, subjects other than those specified in the EVCG document may not be listed in the Subject EV certificate, it is necessary to state attributes in another location - we suggest it in the SubjectAlternativeName extension.

Technical parameters of the certificate RSA2048 + / SHA256 or ECC certificate [prime256v1, secp256r1, NIST P-256, secp384r1, NIST P-384]

All TPP requests, where technically possible, must be protected by TLS protocol with mutual authentication where PKI certificates are used.

Extension	Description
SubjectAlternativeName	
.Description (2.5.4.13)	**ASPSP role text tag** • UnboundedDirectoryString, max. 1024 chars
.DN Qualifier (2.5.4.46)	**Regulator name** • PrintableString, max. 64 chars
.DMDName (2.5.4.54)	**the ASPSP license number available in the public registerASPSP** • UnboundedDirectoryString, 64 chars
CertificatePolicies	
.PolicyInformation (1)	EV policy of certificate issuer mandatory for EVC
.PolicyInformation (2)	**QCP-w (0.4.0.194112.1.4)** recommended by EN 319411-2
qcStatements	
.Id-etsi-qcs-QcCompliance (0.4.0.1862.1.1)	Mandatory
.id-etsi-qcs-QcPDS (0.4.0.1862.1.5)	Optional; link (URI, https) to user report (PDS)
.QcType (0.4.0.1862.1.6)	Mandatory **id-etsi-qct-web (0.4.0.1862.1.6.3)**

.pkixQCSyntax-v2 (1.3.6.1.5.5.7.11.2)	a link to the website (i) of the Regulator (NBS), (ii) the PSP public register, (iii) the register of banks
	URI to Registration Authority (NBS)
	ASN.1 definition (for qcStatement-2):
	statementId::= {id-qcs 2} = 1.3.6.1.5.5.7.11.2
	statementInfo (SemanticsInformation)
	::=SEQUENCE{
	NameRegistrationAuthorities
	::=SEQUENCE{ SIZE (1..MAX) OF
	GeneralName
	::=CHOICE{
	uniformResourceIdentifier [6]
	IA5String }
	}
	where in the uniformResourceIdentifier entry is the https address of the Registrar's of the Regulator's Website (NBS) with reference to public register

TPP may also use an eIDAS eSeal Certificate for application purposes. For the certificate in X.509v3 format, we recommend using the simple text type standard features described in X.520 below.

Subject	Description
Subject	
.organizationIdentifier (2.5.4.97)	TPP Registration number issued by regulator (NBS, paragraph 2, article.34 RTS) in format XX:CC-nnnnnnn where: • XX is chosen by NBS – e.g. PS:= payment service or IC:= id. code etc. • CC is country code [SK] • nnnnnnn TPP registration number published in regulator's public regiustry
.Description (2.5.4.13)	**ASPSP role text tag** • UnboundedDirectoryString, max. 1024 zn.
.DN Qualifier (2.5.4.46)	**Regulator name** • PrintableString, 64zn.
Extension	**Description**
qcStatements	

.pkixQCSyntax-v2 (1.3.6.1.5.5.7.11.2)	a link to the website (i) of the Regulator (NBS), (ii) the PSP public register, (iii) the register of banks
	URI to Registration Authority (NBS)
	ASN.1 definition (pre qcStatement-2):
	statementId::= {id-qcs 2} = 1.3.6.1.5.5.7.11.2
	statementInfo (SemanticsInformation)
	::=SEQUENCE{
	NameRegistrationAuthorities
	::=SEQUENCE{ SIZE (1..MAX) OF
	GeneralName
	::=CHOICE{
	uniformResourceIdentifier [6]
	IA5String }
	}
	where in the uniformResourceIdentifier entry is the https address of the Registrar's of the Regulator's Website (NBS) with reference to public register

4.3.1 Restrictions

During the transition period, until the validity of relevant RTS, any EV certificates as well as eSeal certificates can be used. The ASPSP reserves the right to define the process of registering such a certificate.

4.4 General Design aproach

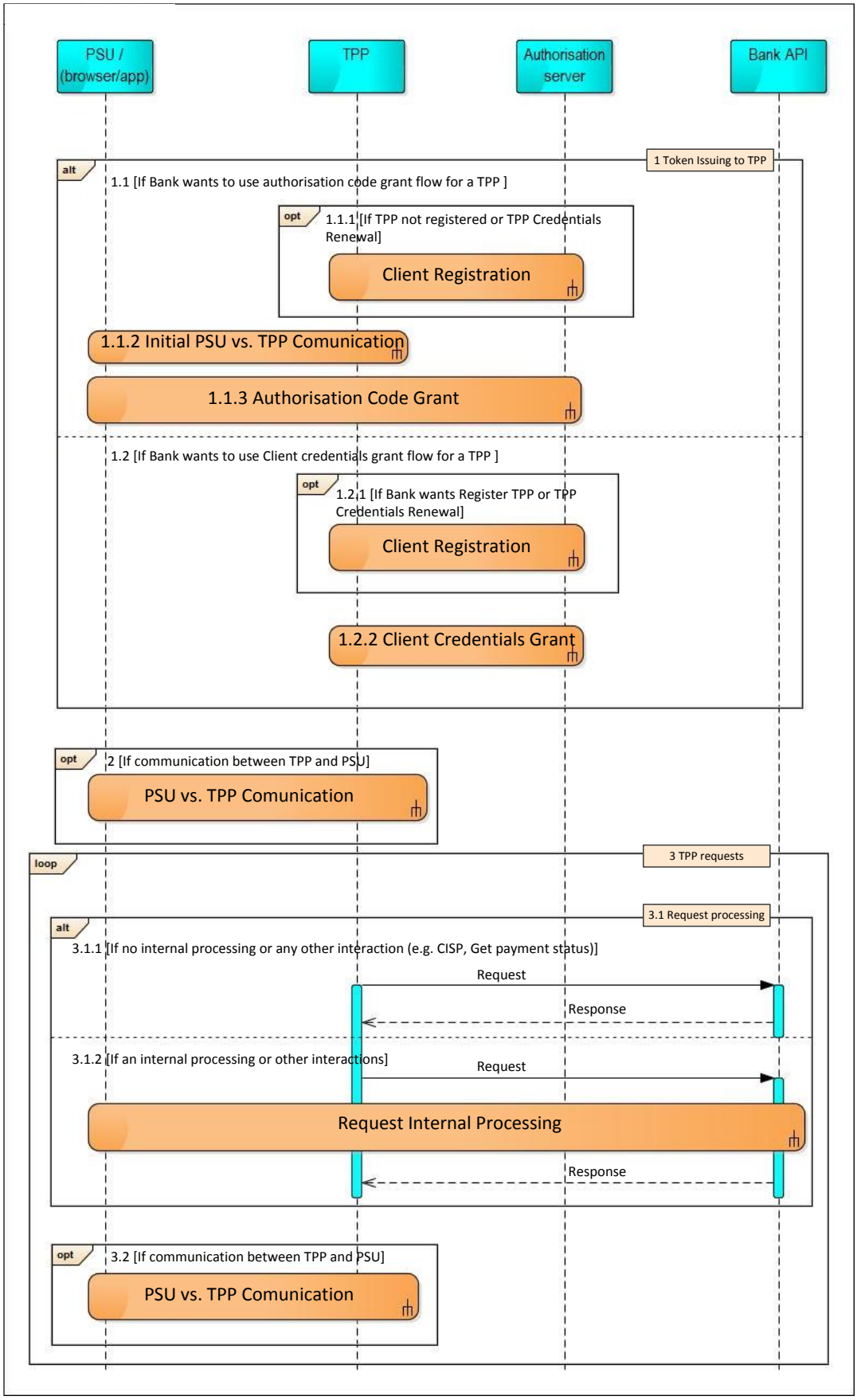

Figure 1: General Design Aproach

After successful TPP PSD2 registration, the TPP is given by license number and PKI certificate which contains the license number. Since this point, the TPP is allowed to perform both communication with an ASPSP, use development portal and access the API documentation.

ASPSP will manage communication with TPP using the IETF RFC 6749 - The OAuth 2.0 Authorization Framework („OAuth framework“, hereafter only). Therefore, in order to access ASPSP's API, TPP must be given by an access token which must be presented when performing a call. The access token usage is defined by the ITEF RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage („**access_token“**, hereafter only). All TPP requests, where technically possible, must be protected by TLS protocol with mutual authentication, where PKI certificates used are in accordance to definitions in the Section TPP and ASPSP authentication. If such a TLS communication is not possible, requests should be protected at least by PKI signatures performed using PKI certificates in accordance to definitions in the Section TPP and ASPSP authentication.

4.4.1 Alternatives for Token Issuing to TPP

ASPSP can decide whether it will use Authorization code grant flow according to RFC 6749, Section 4.1 of OAuth framework, or it will use rather Client credentials grant flow according to RFC 6749, Section 4.4. of OAuth framework. For PIISP Client credentials grant flow only is allowed.

4.4.1.1 If ASPSP wants to use Authorization code grant flow for a TPP

If ASPSP intends to use Authorization code grant flow for a TPP access, the TPP client must be given by client credentials according to the requirements of the flow. The client credentials are given by a Client registration process which must be implemented in compliance with Section 2 of OAuth framework.

If TPP is not registered or it needs to renew its credentials, the TPP uses Client registration process (*Module opt 1.1.1 If TPP not registered or TPP Credentials Renewal* in the Sequence diagram), see the Section Assigning a technical identifier.

The technical identifier consists of **client_id** and **client_secret** and is used for automated communication with the ASPSP to obtain valid **access_token** and **refresh_token**. Assigning a technical identifier is not required. In the absence of a technical identifier, only the client credentials grant method with a valid PKI certificate can be used.

The technical identifier can also be assigned by business process of the ASPSP, according to RFC 6749, Section 2.3.2. The specific process of assigning a technical identifier in a different manner is not part of the standard. To obtain a technical identifier automatically, TPP can use the automatic enrollment process specified in the Section Optional implementation: Enrollment.

Afterwards the TPP client registration, an Authorization code grant flow begins as a consequence of a PSU request to TPP (*Module 1.1.2 Initial PSU vs. TPP Communication* in the Sequence diagram), eg. PSU install mobile application of TPP.

The Authorization code grant flow (*Module 1.1.3 Authorization Code Grant* in Sequence diagram) must be implemented in compliance with RFC 6749, Section 4.1 of OAuth framework, where **access_token** and optionally **refresh_token** are issued as the finale step of the flow. Further implementations details, such as lifetime of tokens, are not defined by this document.

4.4.1.2 If ASPSP wants to use Client credentials grant flow for a TPP

If the ASPSP wants to use Client credentials grant flow for a TPP access instead of Authorization code grant flow (*Item 1.1*), the ASPSP must implement the Client credentials grant flow in compliance with RFC 6749, Section 4.4. Client Credentials Grant of OAuth framework. In this case, the ASPSP can decide whether it will require TPP to register (*Module opt 1.2.1 If ASPSP wants Register TPP or TPP Credentials Renewal* in the Sequence diagram) in terms of RFC 6749, Section 4.4. Client Credentials Grant of OAuth framework, or the ASPSP will rely on a PKI certificate that must be in accordance with Section TPP and ASPSP authentication, and will not require TPP to register.

In this case, a TPP is given by access token by using Client credentials grant flow (see *Module 1.2.2 Client Credentials Grant* in Sequence diagram). Further implementations details, such as lifetime of tokens, are not defined by this document.

4.4.2 Optional PSU vs. TPP Communication

After the token is being issued to the TPP, a PSU instructs the TPP to perform requests on behalf of the PSU (*Module opt 2 If communication between TPP and PSU* in Sequence diagram). TPP is not obliged to receive a token should TPP act on behalf on PSU based on PSUs concent given at earlier stages. The communication between PSU and TPP is not specified by this document.

4.4.3 TPP requests

Upon TPP being issued a valid access token, it may call only the services defined by this Standard and made available by a particular ASPSP.

If the access token becomes invalid, the TPP must perform actions to obtain new access token according to Item 1 above in this Section (*Module alt 1 Token Issuing to TPP* in Sequence diagram).

There are two possibilities how the request of TPP can be processed by ASPSP. If the request does not need any other interaction with PSU or TPP in order to be processed, the response is returned such as it is figured by *Module 3.1.1 If no internal processing or any other interaction* (e.g. PIISP, Get payment status) in Sequence diagram. Otherwise, if there is an interaction with PSU or TPP needed to process the request of TPP, like the request must be authorized using SCA by PSU and/or there are more endpoints on the side of ASPSP which must be called by TPP to acquire the desired action, the flow must be implemented such as it is figured by *Module 3.1.2 If an internal processing or other interactions* in Sequence diagram. The Request Internal Processing specification and implementation is left on the decision of ASPSP.

Then optionally, next PSU vs. TPP commination can occur, for instance to show result of an action which TPP was instructed to perform on behalf of PSU in ASPSP, or another instructions are given to TPP by PSU.

4.4.4 Assigning a technical identifier

The technical identifier consists of **client_id** and **client_secret** and is used for automated communication with the ASPSP to obtain valid **access_token** and **refresh_token**. Assigning a technical identifier could be required by ASPSP. In the absence of a technical identifier, only the client credentials grant method with a valid PKI certificate can be used.

The technical identifier can also be assigned by business process of the ASPSP, according to RFC 6749, Section 2.3.2. The specific process of assigning a technical identifier in a different manner is not part of the standard. To obtain a technical identifier automatically, ASPSP can use the automatic enrollment process specified in the Section Optional implementation: Enrollment.

4.5 Optional implementation: Enrollment

This section defines non mandatory part of ASPSP implementation. ASPSP does not have to use and implement the definitions bellow.

4.5.1 Automated assigning of a technical identifier

The technical identifier can be assigned by the following automated process. The automatic process is no mandatory.

By calling this resource, a TPP with a valid PKI certificate can request the automatic assignment of **client_id** and **client_secret**. The output is **client_id** and **client_secret**, which the TPP needs to get **access_token** and **refresh_token**.

Endpoint: POST https://ib.banka.sk**/enroll**

Request

Attribute	Optionality	Type	Description
redirect_uris	Mandatory	Array of strings e.g. URL [Max 3x 2047 B]	A list of URLs to which the authentication flow is redirected at the end. The authorization request must contain just one of these registered URIs in the exact format.
client_name	Mandatory	string [Max 255 B]	TPP application name
client_type	Mandatory	string	OAuth defines two client types, based on their ability to authenticate securely with the authorization server (Confidential/Public). ASPSP does accept confidential clients only.
client_name#en-US	Optional	string [Max 1024 B]	TPP name in the appropriate language / encoding.
logo_uri	Optional	URI [Max 2047 B]	Application logo URI (or where to download it at registration)
contacts	Mandatory	Array of strings e-mail [Max 10x 255 B]	E-mails as a contact to a responsible person on the TPP side.
scopes	Optional	Array of strings [Max 10x 255 B]	Array of the required scopes by application. At registration, scopes are validated against the content of the certificate used.
license number	Mandatory	string [Max 1024 B]	License number obtain by national regulator

Response (only new fields are listed)

Attribute	Optionality	Type	Description
client_id	Mandatory	string	The **client_id** assigned to application. This ID starts the authentication process and the communication process when replacing the code and **refresh_token**.
client_secret	Mandatory	string	**client_secret** - password / token issued by the ASPSP for the application (**client_id**) of the TPP
client_secret_expires_at	Optional	DateTime	The default value is 0 (**client_id** never expires). Otherwise, the value is in seconds from 1970-01-01T0: 0: 0Z
api_key	Optional	string	The API key that the app uses to communicate with the ASPSP's API. If API does not support API keys, returns "NOT_PROVIDED"

Error codes

HTTP Status	Error code	Description
400	**invalid_request**	Invalid request. The query is missing a required field or is in an inappropriate / invalid format.
400	**invalid_scope**	Invalid scope of request.
400	**invalid_redirect_uri**	The value of one or more redirect URI is not valid.
401	**invalid_client**	Invalid client_id.
401	**unauthorized_client**	The TPP is not authorized to execute this demand.
401	**access_denied**	The authorization server denied access.
403	**insufficient_scope**	E.g. insufficient authorization to use the required scope.
500, 503	**server_error**	Authorization server error.

HTTP Request:

```
POST /enroll HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: ib.banka.sk

{

   "redirect_uris":
     ["https://www.multipay.sk/start",
         "https://www.multipay.sk/start2"],
   "client_name": "Môj platobný portál",
   "client_name#en-US": "My payment portal",
   "logo_uri": "https://www.multipay.sk/logo.png",
   "contacts": ["admin@multipay.sk"],
   "scopes": ["aisp", "pisp"],
   "client type": "confidential",
   "licence number": "v@Nn123456"
}
```

HTTP Response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
   "client_id": "0187862545465",
   "client_secret":"AAjkk45suiyui564568712_4555g5g5g5gg",
   "client_secret_expires_at": 0,
   "api_key": "00000000-1212-0f0f-a0a0-123456789abc",
   "redirect_uris":
     ["https://www.mymultipay.sk/start",
      "https://www.mymultipay.sk/start2"],
   "client_name": "Môj platobný portál",
```

```
    "client_name#en-US": "My payment portal",
    "logo_uri": "https://www.multipay.sk/logo.png",
    "contacts": ["admin@multipay.sk"],
    "scopes": ["aisp", "pisp"],
    "client type": "confidential",
    "licence number": "v@Nn123456"

}
```

4.5.2 Change of registration data

By calling this resource, the TPP may request to change the application-specific registration details. To call a resource, TPP must use a valid PKI certificate and **client_id** that is issued to this TPP. Output is a reviewed changed data

Endpoint: PUT https://ib.banka.sk**/enroll/{client_id}**

Request

Attribute	Optionality	Type	Description
redirect_uris	Mandatory	Array of strings e.g. URL [Max 3x 2047 B]	A list of URLs to which the authentication flow is redirected at the end. The authorization request must contain just one of these registered URIs in the exact format.
client_name	Mandatory	string [Max 255 B]	TPP application name
client_name#en-US	Optional	string [Max 1024 B]	TPP name in the appropriate language / encoding.
client_type	Mandatory	string	OAuth defines two client types, based on their ability to authenticate securely with the authorization server (Confidential/Public). ASPSP does accept confidential clients only.
logo_uri	Optional	URI [Max 2047 B]	Application logo URI (or where to download it at registration)
contacts	Mandatory	Array of strings e-mail [Max 10x 255 B]	E-mails as a contact to a responsible person on the TPP side.
scopes	Optional	Array of strings [Max 10x 255 B]	Array of the required scopes by application. At registration, scopes are validated against the content of the certificate used.

Response

Attribute	Optionality	Type	Description
client_id	Mandatory	string	The unique identifier of the TPP application issued by the ASPSP.
redirect_uris	Mandatory	Array of strings e.g. URL	A list of URLs to which the authentication flow is redirected at the end. The authorization request must contain just one of these registered URIs in the exact format.
client_name	Mandatory	string	TPP application name
client_name#en-US	Optional	string	TPP name in the appropriate language / encoding.
logo_uri	Optional	URI	Application logo URI (or where to download it at registration)
contacts	Mandatory	Array of strings emails	E-mails as a contact to a responsible person on the TPP side.
scopes	Optional	Array of strings	Array of the required scopes by application. At registration, scopes are validated against the content of the certificate used.

Error codes

HTTP Status	Error code	Description
400	**invalid_request**	Invalid request. The query is missing a required field or is in an inappropriate / invalid format.
400	**invalid_scope**	Invalid scope of request.
400	**invalid_redirect_uri**	The value of one or more redirect URI is not valid.
401	**invalid_client**	Invalid client_id.
401	**unauthorized_client**	The TPP is not authorized to execute this demand.
401	**access_denied**	The authorization server denied access.
403	**insufficient_scope**	E.g. insufficient authorization to use the required scope.
500, 503	**server_error**	Authorization server error.

HTTP request:

```
PUT /enroll/a0b25291f0 HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: ib.banka.sk

{
   "application_type": "web",
```

```
    "redirect_uris":
      ["https:// https://www.multipay.sk/start",
       "https:// https://www.multipay.sk/start2"],
    "client name": " Môj platobný portál",
    "client_name#en-US": "My payment portal",
    "logo_uri": "https:// https://www.multipay.sk/logo.png",
    "contact": "info@multipay.sk",
    "scopes": ["aisp", "pisp"],
    "client type": "confidential"
}
```

HTTP response:

```
HTTP/1.1 200
Content-Type: application/json

{
   "client_id": "a0b25291f0",
   "client_secret_expires_at": 0,
   "application_type": "web",
   "redirect_uris":
     ["https://www.multipay.sk/start",
      "https://www.multipay.sk/start2"],
   "client_name": "Moja univerzálna banka",
   "client_name#en-US": "My cool bank",
   "logo_uri": "https://www.multipay.sk/logo.png",
   "contact": "info@multipay.sk",
   "scopes": ["aisp", "pisp"],
   "client type": "confidential"

}
```

4.5.3 Delete application specific credentials

By calling this resource, the TPP may request to remove data and application-specific credentials. To call a resource, TPP must use a valid PKI certificate and **client_id** that is issued to this TPP. Output is confirmation of deletion.

Endpoint: DELETE https://ib.banka.sk/enroll/{client_id}

Request

Payload is empty.

Response

Payload is empty.

Error codes

HTTP Status	Error code	Description
400	**invalid_request**	Invalid request. The query is missing a required field or is in an inappropriate / invalid format.
401	**invalid_client**	Invalid client_id.
401	**unauthorized_client**	The TPP is not authorized to execute this demand.
401	**access_denied**	The authorization server denied access.
500, 503	**server_error**	Authorization server error.

HTTP request:

```
DELETE /register/a0b25291f0 HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: ib.banka.sk
```

HTTP response:

```
HTTP/1.1 204 No content
```

4.5.4 Request a new client_secret

By calling this resource, TPP can request a new **client_secret**. To call a resource, TPP must use a valid PKI certificate and **client_id** that is issued to this TPP. The original **client_secret** will be invalidated by this request.

Endpoint: POST https://ib.banka.sk**/enroll/{client_id}/renewSecret**

Request

Payload is empty.

Response

Attribute	Optionality	Type	Description
client_id	Mandatory	string	The **client_id** assigned to application. This ID starts the authentication process and the communication process when replacing the code and **refresh_token**.
client_secret	Mandatory	String	**client_secret** - password / token issued by the ASPSP for the application (**client_id**) of the TPP.
client_secret_ expires_at	Optional	DateTime	The default value is 0 (**client_id** never expires). Otherwise, the value is in seconds from 1970-01-01T0: 0: 0Z

Error codes

HTTP Status	Error code	Description
400	**invalid_request**	Invalid request. The query is missing a required field or is in an inappropriate / invalid format.
401	**invalid_client**	Invalid client_id.
401	**unauthorized_client**	The TPP is not authorized to execute this demand.
401	**access_denied**	The authorization server denied access.
500, 503	**server_error**	Authorization server error.

HTTP request:

```
POST /register/a0b25291f0/renewSecret HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: ib.banka.sk
```

HTTP response:

```
HTTP/1.1 200 OK

{
   "client_id": "a0b25291f0",
   "client_secret": "BBjkk45sd78ad454gddd8712_4555g5g5g5gg",
   "client_secret_expires_at": 0
}
```

5 AISP

Chapter defines list of methods and alternative of flows provided for AISPs.

Prerequisites:

a) The TPP is registered for the AISP role and valid AISP scope
b) The TPP has been successfully checked and authenticated
c) The TPP has presented its "OAuth2 Authorization Code Grant" access token which allows the ASPSP to identify the relevant PSU

5.1 Endpoints definition

Following sections describes technical definition of provided endpoints for AISPs.

Endpoint	Method	Optionality	Description
/api/v1/**accounts/information**	POST	Mandatory	**Account information** – service provide information and balances related to an account
/api/v1/**accounts/transactions**	POST	Mandatory	**Account transactions** – service provide list of transactions in defined date range related to an account
/api/v1/**accounts**	GET	Optional	**List of accounts** - service returns the list of accounts to which the client has given a long-term consent to specific TPP (not a list of all client accounts) without balances

5.1.1 Standard header definition

Recommended set of request and response headers for AISP endpoints

Request header definition

Attribute	Optionality	Type	Description
Content-Type	Mandatory	String	application/json or application/xml
Authorization	Mandatory	String	Authorization is defined in RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage
Request-ID	Mandatory	String	An unique identifier of a particular request message. Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
Correlation-ID	Optional	String	An unique correlation identifier correlates the request and the response messages as a pair especially useful for audit logs. Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
Process-ID	Optional	String	Identifier of a business or technical process to what the set of requests and response pairs are organized (e.g. paging of transaction history should have same Process-ID). Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
PSU–IP-Address	Mandatory	String	Identifier of a customer's IP address from which he/she is connected to the TPP infrastructure. It might be in the format of IPv4 o IPv6 address. ASPSP shall indicate which values are acceptable.
PSU-Device-OS	Mandatory	String	A customer's device and/or operating system identification from which he/she is connected to the TPP infrastructure.
PSU-User-Agent	Mandatory	String	A customer's web browser of other client device identification from which he/she is connected to the TPP infrastructure. Agent header field of the http request between PSU and TPP.)
PSU-Geo-Location	Optional	String	The GPS coordinates of the current customer's location in the moment of connection to the TPP infrastructure. (Required GPS format: Latitude, Longitude)
PSU-Last-Logged-Time	Optional	DateTime	Last date and time when user was logged to TPP app (RFC3339 format)

Response header definition

Attribute	Optionality	Type	Description
Content-Type	Mandatory	String	application/json or application/xml
Response-ID	Mandatory	String	An unique identifier of a particular request message. Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
Correlation-ID	Optional	String	An unique correlation identifier correlates the request and the response messages as a pair especially useful for audit logs. Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
Process-ID	Optional	String	Identifier of a business or technical process to what the set of requests and response pairs are organized (e.g. paging of transaction history should have same Process-ID). Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).

HTTP AISP Request header example:

Header

```
   Content-Type: application/json
   Authorization: Bearer          IDWJJBCHQ5DZJWEMO7ZWM4DLYWOFWKXXRequest-ID:
c2c48fc8-1f79-4934-a47b-56d61a28f351
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
   PSU-Last-Logged-Time: 2017-07-31T14:54:32+01:00
   PSU-IP-Address: 192.168.0.100
   PSU-Device-OS: iOS 11
   PSU-User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2228.0 Safari/537.36
   PSU-Geo-Location: 48.145745, 17.116062
```

HTTP AISP Response header example:

Header

```
   Content-Type: application/json
   Response-ID: ac30869e-29e2-40f7-83fb-ed1c6bdde216
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
```

5.1.2 AISP Operation: Account information

The operation provides the relevant data about PSU account identified by IBAN and two types of account balances: Interim booked and interim available balance. Only AISP is allowed to use current endpoint.

Endpoint: POST /api/v1/accounts/information

Request

Attributes structure Level 1	Optionality	Type	Description
iban	Mandatory	String [34]	International Bank Account Number (IBAN)

Response (if no error)

Attributes sturcture			Optionality	Type	Description
Level 1	Level 2	Level 3			
account	*name*		Mandatory	String [70]	**Account name** - usually client name
account	*productName*		Optional	String [70]	**Product name** - commercial product designation
account	*type*		Optional	Enum	**Account type** is enumeration: ISO 20022 - Cash Account Type Code e.g. (CACC - Current account)
account	*baseCurrency*		Mandatory	String [3]	**Account currency** (currency code according to ISO 4217 - 3 capital letters)
balances	*typeCodeOrProprietary*		Mandatory	Enum	**Balance type** is enumeration: ISO 20022 - Balance Type Code. Following balances mandatory are published: - ITBD (Interim booked balance) - ITAV (Interim available balance)
balances	*amount*	*value*	Mandatory	Number Float [12.2]	**Balance amount.** Numeric value of the amount as a fractional number. The fractional part has a maximum of two digits
balances	*amount*	*currency*	Mandatory	String [3]	**Balance currency** (currency code according to ISO 4217 - 3 capital letters)
balances	*creditDebitIndicator*		Mandatory	Enum	**Credit/Debit indicator** is enumeration: - CRDT (Credit) - DBIT (Debit)
balances	*dateTime*		Mandatory	DateTime	**Timestamp of balances** (official local date and time of Slovak republic in RFC 3339 format)

Links to ISO 20022 enumerations:

- Account types: https://www.iso20022.org/standardsrepository/public/wqt/Description/mx/dico/codesets/_a3ed5tp-Ed-ak6NoX_4Aeg_-1826678245
- Balance type: https://www.iso20022.org/standardsrepository/public/wqt/Description/mx/dico/codesets/_bbFhQNp-Ed-ak6NoX_4Aeg_142948041

Error codes

Recommended set of HTTP Status codes and corresponding custom error codes:

HTTP Status	Error code	Description
400	**parameter_missing**	Mandatory parameter is missing
400	**parameter_invalid**	Value of input parameter is not valid
500, 503	**server_error**	Authorization server error.
Rest of HTTP Status codes and error codes are defined according to RFC 6749, Section 5.2		

5.1.3 AISP Operation: Account transactions

The operation provides the list of financial transactions performed on a customer's bank account within a date period. Transaction history will only include transactions that affect the balance (reserved and booked transaction). Transactions will be ordered from the most recent to the oldest. The range of attributes provided for transactions is based on ISO 20 022 - CAMT.054.

Only AISP is allowed to use current operation.

Endpoint: POST /api/v1/accounts/transactions

Request

Attributes structure Level 1	Optionality	Type	Description
iban	Mandatory	String [34]	International Bank Account Number (IBAN)
dateFrom	Optional	Date	The starting date of a date period for transaction history. Default value is actual day.
dateTo	Optional	Date	The end date of a date period for transaction history. ASPSPs provide transaction's history for at least 13 months. Default value is actual day.
pageSize	Optional	Integer	The number of records included in one page for displaying. Default value is 50 records. ASPSP has to supports at least 100 records on page.
page	Optional	Integer	The sequence number of a page in regards to page size for a record set. Because it starts at number 0, it should be considered as an offset from the beginning from a page set. Default value is 0.
status	Optional	Enum	Transaction status indicator is enumeration: - BOOK (booked transactions) - INFO (settled transactions) - ALL (all transactions) Default value is ALL

Response (if no error)

Collection of information sets about customer's financial transactions executed at their bank account.

Attributes structure					Optionality	Type	Description
Level 0	Level 1	Level 2	Level 3	Level 4			
pageCount					Optional	Number	Number of pages in the selected range
transactions	*amount*	*value*			Mandatory	Number Float [12.2]	**Transaction amount value** in account currency. Numeric value of the amount as a fractional number.
	amount	*currency*			Mandatory	String [3]	**Transaction amount currency**. Formated in Alphabetic codes from ISO 4712.
	creditDebitIndicator				Mandatory	Enum	**Credit/Debit indicator is enumeration:** - CRDT (Credit) - DBIT (Debit)
	reversalIndicator				Optional	boolean	The flag determining that it is the **reversal transaction** for some previous one.
	status				Mandatory	Enum	**The status of a transaction**, related to the query parameter 'transactionStatus'. Transaction status indicator is enumeration: - BOOK (booked transactions) - INFO (settled transactions
	bookingDate				Mandatory for booked tnx.	Date	**Transaction booking date.** The date of the execustion of the transaction.
	valueDate				Mandatory	Date	**Transaction value date.** The requested date by a bank customer to execute the transaction.
	bankTransactionCode				Optional	String [11]	**The category code** of the transaction type from the SBA's code list.
	transactionDetails	*references*	*accountServicerReference*		Optional	String [35]	**The unique identifier of the transaction generated by a ASPSP** that it should be considered as a ASPSP reference.
	transactionDetails	*references*	*instructionIdentification*		Optional	String [35]	Technical identification of the payment generated by a client.
	transactionDetails	*references*	*endToEndIdentification*		Mandatory in case this attribute is provided by client	String [35]	Unique identification defined by a requestor.
	transactionDetails	*references*	*transactionIde*		Optional	String [35]	The payment reference for related fees.

			ntification				
	transactionDetails	*references*	*mandateIdentification*		Mandatory for Direct debit tnx.	String [35]	The **mandate referece** as its reference number.
	transactionDetails	*references*	*chequeNumber*		Optional	String [35]	**For card transactions**, this is the card number in format **** **** **** 1111
	transactionDetails	*counterValueAmount*	*amount*	*value*	Optional	Number Float [12.2]	**Transaction amount value** in account currency.
	transactionDetails	*counterValueAmount*	*amount*	*currency*	Optional	String [3]	**Transaction amount currency**. Formated in Alphabetic codes from ISO 4712.
	transactionDetails	*counterValueAmount*	*currencyExchange*	*exchangeRate*	Optional	Number Float [12.2]	The used **exchange rate for conversion** from the instructed currency to the target account currency.
	transactionDetails	*relatedParties*	*debtor*	*name*	Optional	String [140]	**Name of the debtor**
	transactionDetails	*relatedParties*	*debtorAccount*	*identification*	Optional	String [34]	Unique identification of the **debtor account**, usually IBAN.
	transactionDetails	*relatedParties*	*creditor*	*name*	Optional	String [140]	**Name of the creditor**
	transactionDetails	*relatedParties*	*creditor*	*identification*	Optional	String [35]	The **creditor identifier** (CID) in the direct debit transaction.
	transactionDetails	*relatedParties*	*creditorAccount*	*identification*	Optional	String [34]	Unique identification of the **creditor account**, usually IBAN.
	transactionDetails	*relatedParties*	*tradingParty*	*name*	Optional	String [140]	Name of a third party. For card transaction, this is the name of merchant.
	transactionDetails	*relatedParties*	*tradingParty*	*identification*	Optional	String [35]	Unique identification of a third party. For card transaction, this is ID of merchant.
	transactionDetails	*relatedParties*	*tradingParty*	*merchantCode*	Optional	String [4]	A **Merchant Category Code** (MCC) coordinated by MasterCard and Visa.
	transactionDetails	*relatedAgents*	*debtorAgent*	*financialInstitutionIdentification*	Optional	String [11]	Corresponding identification of a **debtor bank** managing the account, usually Bank Identification Code (BIC).
	transactionDetails	*relatedAgents*	*creditorAgent*	*financialInstitutionIdentification*	Optional	String [11]	Corresponding identification of a **creditor bank** managing the account, usually Bank Identification Code (BIC).

	transaction Details	*remittanceInformation*			Mandatory in case this attribute is provided by client	String [140]	The text aimed as the information for a receiver of the transaction.
	transaction Details	*related Dates*	*acceptanceDateTime*		Optional	Date	**Transaction entry date**. The date of receiving the transaction in a bank.
	transaction Details	*additionalTransactionInformation*			Optional	String [140]	Bank transaction description.

Links to enumerations:

- The category code of the transaction type from the SBA's code list can be find in document *„xmlstatement_sk_v2.4_2016.docx“* in section *„4.3.3 Transaction Codes“*: http://www.sbaonline.sk/files/subory/KPS/verejne/xmlstatement_sk_v2.4_2016.docx

Error codes

Recommended set of HTTP Status codes and corresponding custom error codes:

HTTP Status	Error code	Description
400	**parameter_missing**	Mandatory parameter is missing
400	**parameter_invalid**	Value of input parameter is not valid
500, 503	**server_error**	Authorization server error.
Rest of HTTP Status codes and error codes are defined according to RFC 6749, Section 5.2		

5.1.4 Optional AISP Operation: List of accounts

The operation provides the list of accounts to which the client has given a long-term consent to specific TPP (not a list of all client accounts) without balances.

Only AISP is allowed to use current operation.

Endpoint: GET /api/v1/accounts

Request

Payload is empty.

Response (if no error)

Attributes sturcture			Optionality	Type	Description
Level 1	**Level 2**	**Level 3**			
creationDateTime			Mandatory	DateTime	The **date and time** in RFC3339 format at which a particular action has been requested or executed.
accounts	*identification*	*iban*	Mandatory	String	International Bank Account Number (IBAN)
accounts	*name*		Mandatory	String [70]	**Account name** - usually client name
accounts	*productName*		Optional	String [70g]	**Product name** - commercial product designation
accounts	*type*		Optional	Enum	**Account type** is enumeration: ISO 20022 - Cash Account Type Code e.g. (CACC - Current account)
accounts	*baseCurrency*		Mandatory	String [3]	**Account currency** (currency code according to ISO 4217 - 3 capital letters)
accounts	*servicer*	*financialInstitutionIdentification*	Mandatory	String [11]	Corresponding identification of a servicing bank managing the account, usually **Bank Identification Code** (BIC).
accounts	*consent*		Mandatory	Array [String]	Consent containts set of particular account's scopes for TPP. Formated as array of following enumerations: AISP, PISP, PIISP.

Error codes

Recommended set of HTTP Status codes and corresponding custom error codes:

HTTP Status	Error code	Description
400	**parameter_missing**	Mandatory parameter is missing
400	**parameter_invalid**	Value of input parameter is not valid
500, 503	**server_error**	Authorization server error.
Rest of HTTP Status codes and error codes are defined according to RFC 6749, Section 5.2		

5.2 Alternative flow implementation

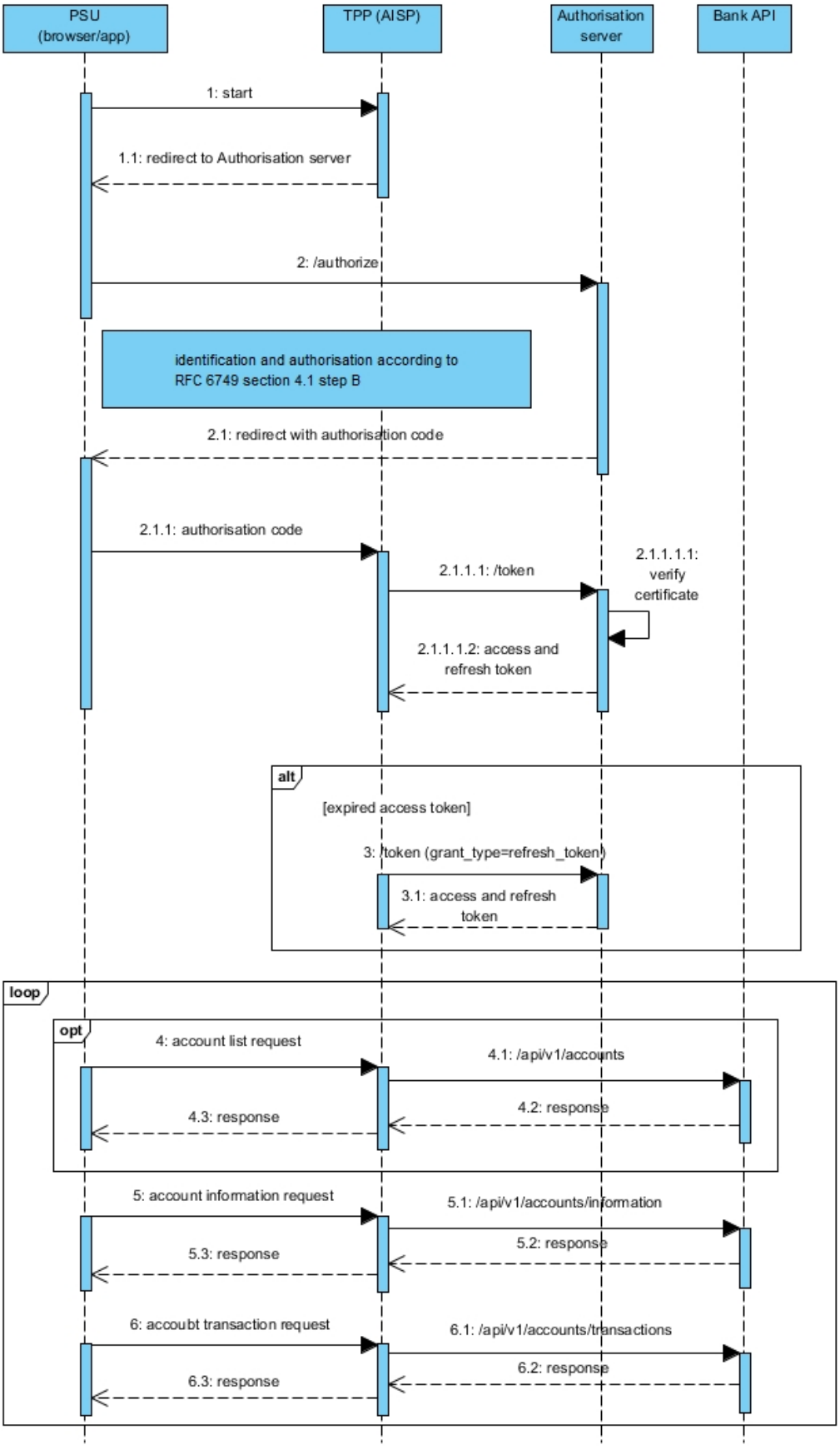

Figure 2: Implementation of AISP Services

5.2.1 Token for AISP services

To access ASPSP API, the TPP must use a valid **access_token** with AISP scope. According to OAuth framework, valid **access_token** and **refresh_token** are used to access ASPSP and PSU' s resources.

TPP uses a short-term **access_token** to communicate with the API of the ASPSP if the ASPSP requires it and it MAY use **refresh_token** to request a new **access_token**.

The OAuth2 PKCE extension (RFC 7636 https://tools.ietf.org/html/rfc7636) is used to issue **access_token** using **code_challenge** and **code_verifier** technique.

Basic properties

- **access_token** is issued as short-term (e.g. 3600 s) and MAY be canceled (by PSU, TPP or ASPSP)
- **refresh_token** can not be directly used to communicate with the API, it has a long validity (e.g. 90 days) and the ASPSP has the option to cancel it and this option can also allow to the PSU.
- The ASPSP and the TPP application share a common "secret" **client_secret**
- The result of identification and authentication according to RFC 6749, Section 4.1, step B is the code that the TPP application must replace with the client secret for **refresh_token** and **access_token**
- the code itself without **client_secret** knowledge CAN NOT be used
- Under the code grant flow, the ASPSP is not required to execute the SCA of PSU of the ASPSP to authorize the TPP's access to ASPSP resources related to that PSU

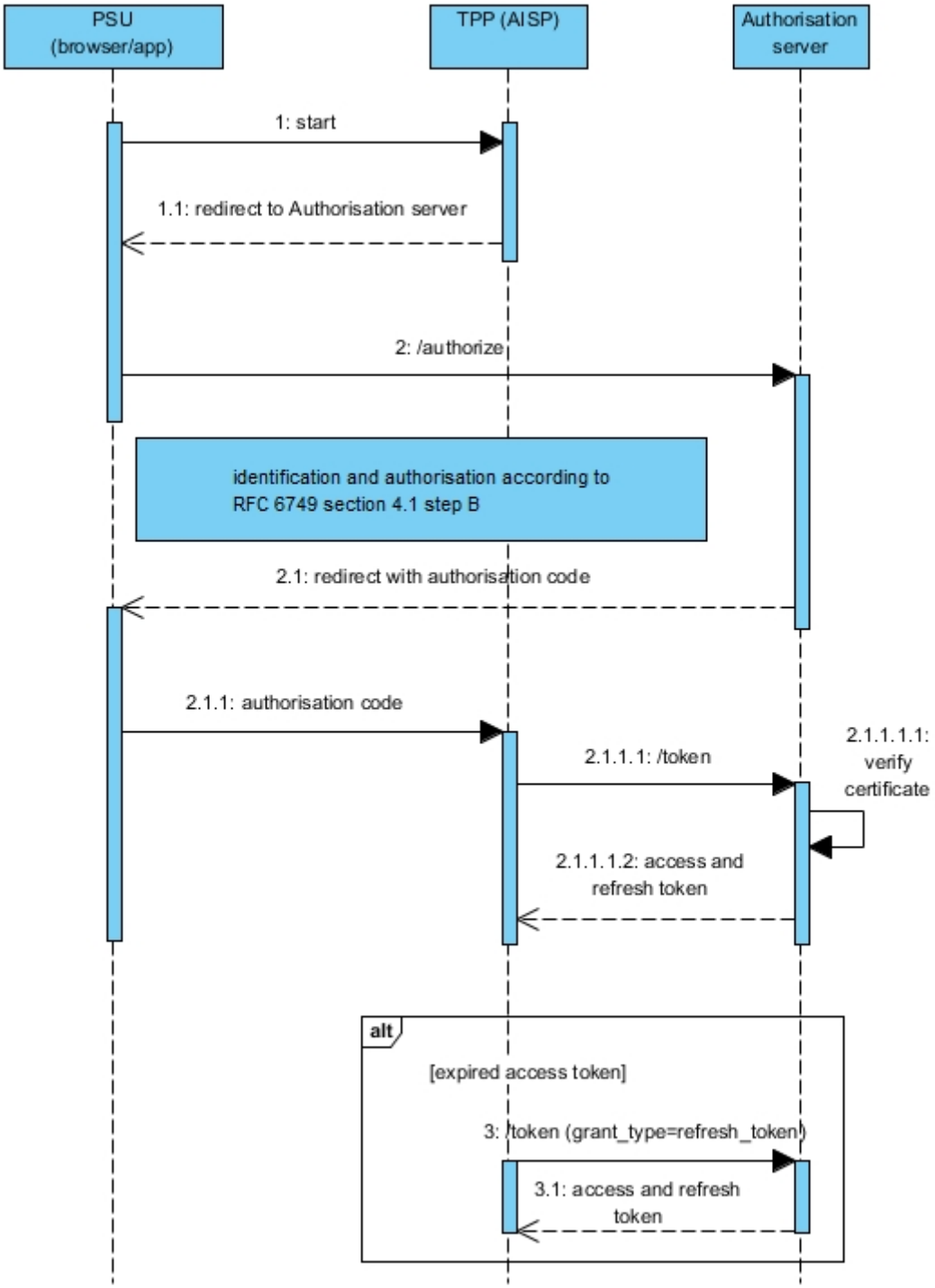

Figure 3: Token for AISP Services

5.2.2 Authorization

The AISP creates an Authorization request for the PSU to consent to the AISP request. The request is an Oauth 2.0. Authorization Code Grant with PKCE extension (requesting for Code)

Endpoint: GET https://ib.banka.sk**/authorize**

Request

Attribute	Optionality	Type	Description
response_type	Mandatory	Code	Mandatory parameter. Specifies the authentication flow used. In this case, a **code grant**. For the authentication process, this means that, as a result of successful identification and authentication, a one-time **auth_code** is expected instead of **access_token**.
client_id	Mandatory	String	Unique TPP application identifier issued by the ASPSP, eg. using the process defined in section 4.5.1
redirect_uri	Mandatory	URL	The URL to which the authentication flow is redirected at the end. This URL is set when **client_id** is issued, and this parameter is validated against the URL introduced to **client_id** in the ASPSP. The value should match one of the values introduced using registration e.g. using the process defined in Section **Chyba! Nenašiel sa žiaden zdroj odkazov.**
scope	Mandatory	String	Space separated string of attributes of the application required scope.
login_hint	Optional	User identification for automation	Hint to the Authorization Server about the login identifier the End-User might use to log in (http://openid.net/specs/openid-connect-core-1_0.html)
state	Mandatory	Random string [min 128 bits]	With this parameter, TPP needs to enrich **redirect_uri** when redirecting. It protects against CSRF attacks and passes information from the application through authentication flow. Requested CSRF token length is min. 128 bits
code_challenge	Mandatory	String	code_challenge = BASE64URL-ENCODE(SHA256(ASCII(code_verifier))) see. RFC 7636

code_challenge_ method	Mandatory	String	S256

Response (only new fields are listed)

Attribute	Optionality	Type	Description
code			Authorization code
State			Attribute state from TPP request

Error codes

Error codes are defined according to RFC 6749, Section 4.1.2.1

2: HTTP Request example: GET /authorize

```
GET /authorize HTTP/1.1
Host: ib.banka.sk
Content-Type: application/x-www-form-urlencoded

response_type=code&
scope=AISP&
client_id=CLIENT_ID&
state=STATE&
redirect_uri=https://www.mymultipay.sk/start&
login_hint=USER_ID
code_challenge=X&
code_challenge_method=S256
```

2.1: HTTP Response example: GET /authorize

```
HTTP/1.1 303 See Other
content-type: application/x-www-form-urlencoded
location: https://www.mymultipay.sk/start?
          code=AUTH_CODE&
          state=STATE
```

The PSU is redirected to the AISP with Authorization code and state parameters in URL.

5.2.3 Get token

The AISP will now possess the Authorization code and state parameter from the ASPSP. State parameter value must be identical as requested by AISP in the previous request otherwise, the response is invalid. AISP will proceed to obtain an Access Token from the ASPSP using the Authorization Code. The AISP will present its Authorization Code together with CLIENT_ID and CLIENT_SECRET in authorization header.

The Access Token is required by the AISP in order to access PSU Account information. The AISP scope should already be associated with the Authorization Code generated in the previous step.

Endpoint: POST https://ib.banka.sk**/token**

Request

Attribute	Optionality	Type	Description
code	Mandatory		Authorization code returned from the code grant
redirect_uri	Mandatory		The redirect URL matches the URL passed in the authentication request.
grant_type	Mandatory		Under the existing OAuth2 definition, this value will be the **authorization_code** if the TPP requested **refresh_token**.
code_verifier	Mandatory		**Code_verifier** used to generate **code_challenge** from a previous request

Response

Attribute	Optionality	Type	Description
access_token	Mandatory		Short-term (e.g. 3600 seconds, in some cases, one-time) token, which can be reissued using **refresh_token**. This token serves to authorize TPP request on ASPSP API.
refresh_token	Optional		Long-term token (e.g. 90days) issued as a replacement for **authorization_code**.
expires_in	Mandatory		The remaining time to expiration of **access_token** - in seconds.
token_type	Mandatory		Type of token „Bearer“
scope	Optional		List of permissions separated by the space for which the token is issued.

Error codes

Error codes are defined according to RFC 6749, Section 5.2

2.1.1.1: HTTP Request example: POST /token

```
POST /token HTTP/1.1
Host: ib.banka.sk
Content-Type: application/x-www-form-urlencoded
Authorization: Basic BASE64(CLIENT_ID + ":" + CLIENT_SECRET)

grant_type=authorization_code&
code=AUTH_CODE&
redirect_uri=REDIRECT_URI&   //[https://www.mymultipay.sk/start]
code_verifier=CODE_VERIFIER
```

2.1.1.1.2: HTTP Response example: POST /token

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
"access_token":"ACCESS_TOKEN",
"token_type":"bearer",
"expires_in":3600,
"refresh_token":"REFRESH_TOKEN",
"scope":"AISP PISP"
}
```

5.2.4 Access token renew

The TPP can save the **refresh_token** from the Get toke resource and ask for a new **access_token** after the expiration of **access_token** through this token. Therefore, TPP can use Get token resource with these parameters:

Endpoint: POST https://ib.banka.sk**/token**

Request

Attribute	Optionality	Type	Description
grant_type	Mandatory		According to the OAuth2 definition, this value will be **refresh_token** if the **access_token** is renewed by **refresh_token**.
refresh_token	Mandatory		Valid **refresh_token** for which the exchange takes place e.g. be9eef9b0af42c674d0b1c1128c37c2g
scope	Mandatory		The scope of the access request. If scope is used, then is checked with scope registered in authorization server. If scope is empty or not used, then is returned scope, which is defined in authorization server.

Response

Error codes

Error codes are defined according to RFC 6749, Section 5.2

3: HTTP Request example: (riadok atribútov je rozdelený pre lepšiu čitateľnosť):

```
POST /token HTTP/1.1
Host: ib.banka.sk
Content-Type: application/x-www-form-urlencoded
Authorization: Basic BASE64(CLIENT_ID + ":" + CLIENT_SECRET)

grant_type=refresh_token&
refresh_token=REFRESH_TOKEN&
scope= AISP PISP
```

3.1: HTTP Response example:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
   "access_token":"ACCESS_TOKEN2",
   "token_type":"bearer",
   "expires_in":3600,
   "refresh_token":"REFRESH_TOKEN2"
}
```

5.2.5 Usage Example of AISP Operation: Account information

Process flow is visible in Figure 2: Implementation of AISP Services

5.1: HTTP Request example: POST /api/v1/accounts/information

Header

```
Content-Type: application/json
Authorization: Bearer IDWJJBCHQ5DZJWEMO7ZWM4DLYWOFWKXX
Request-ID: c2c48fc8-1f79-4934-a47b-56d61a28f351
Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
PSU-Last-Logged-Time: 2017-07-31T14:54:32+01:00
PSU-IP-Address: 192.168.0.100
PSU-Device-OS: iOS 11
PSU-User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2228.0 Safari/537.36
PSU-Geo-Location: 48.145745, 17.116062
```

Body

```
{
  "iban": "SK6807200002891987426353"
}
```

*5.2: HTTP Response example **201**: POST /api/v1/accounts/information*

Header

```
Content-Type: application/json
Response-ID: 7deb90a9-9900-4c90-a91c-3ecc888c2c88
Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
```

Body

```
{
  "account": {
    "name": "John Doe",
    "productName": "BestAccount",
    "type": "CACC",
    "baseCurrency": "EUR"
  },
  "balances": [
    {
      "typeCodeOrProprietary": "ITBD",
      "amount": {
        "value": 1234.56,
        "currency": "EUR"
      },
      "creditDebitIndicator": "CRDT",
      "dateTime": "2017-09-19T17:18:45.727Z"
    },
    {
      "typeCodeOrProprietary": " ITAV ",
      "amount": {
        "value": 1214.06,
        "currency": "EUR"
      },
      "creditDebitIndicator": "CRDT",
      "dateTime": "2017-09-19T17:18:45.727Z"
    }
  ]
}
```

5.2.6 Usage Example of AISP Operation: Account transactions

Process flow is visible in Figure 2: Implementation of AISP Services

6.1: HTTP Request example: POST /api/v1/accounts/transactions

Header

```
Content-Type: application/json
Authorization: Bearer IDWJJBCHQ5DZJWEMO7ZWM4DLYWOFWKXX
Request-ID: c2c48fc8-1f79-4934-a47b-56d61a28f351
Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
PSU-Last-Logged-Time: 2017-07-31T14:54:32+01:00
PSU-IP-Address: 192.168.0.100
PSU-Device-OS: iOS 11
PSU-User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2228.0 Safari/537.36
PSU-Geo-Location: 48.145745, 17.116062
```

Body


```
{
  "iban": "SK6807200002891987426353",
  "transactionType": "ALL",
  "dateFrom": "2017-07-31",
  "dateTo": "2017-08-01",
  "pageSize": 50,
  "page": 0
}
```


*6.2: HTTP Response example **201**: POST /api/v1/accounts/transactions*

Header


```
Content-Type: application/json
Response-ID: 7deb90a9-9900-4c90-a91c-3ecc888c2c88
Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
```


Body


```
{
  "pageCount": 10,
  "transactions": [
    {
      "amount": {
        "value": 1234.56,
        "currency": "EUR"
      },
      "creditDebitIndicator": "CRDT",
      "reversalIndicator": false,
      "transactionStatus": "BOOK",
      "bookingDate": "2017-08-11",
      "valueDate": "2017-08-11",
      "bankTransactionCode": "CO11",
      "transactionDetails": {
        "references": {
          "accountServicerReference": "2c569b47-f402-4b47-8415-
498bfc5ba296",
          "instructionIdentification": "9b766084-57de-48b2-be53-
1bd2804ae0b7",
          "endToEndIdentification": "/VS123/SS456/KS0308",
          "transactionIdentification": "c3b783bb-134e-4d77-bbe0-
2925bdd699a3",
          "mandateIdentification": "c3b783bb-134e-4d77-bbe0-
2925bdd699a3",
          "chequeNumber": "123456******3456"
        },
        "counterValueAmount": {
          "value": 1234.56,
          "currency": "EUR",
          "exchangeRate": 1
        },
        "relatedParties": {
          "debtor": {
            "name": "John Doe"
          },
          "debtorAccount": {
            "identification": "SK6807200002891987426353"
          },
          "creditor": {
            "name": "John Doe",
              "identification": "string/CID",
```

```
            },
            "creditorAccount": {
              "identification": "SK6807200002891987426353"
            },
            "tradingParty": {
              "name": "Merchant name",
              "identification": "AAA-GG-SSSS",
              "merchantCode": "3370"
                 }
          },
          "relatedAgents": {
            "debtorAgent": {
               "financialInstitutionIdentification": "GIBASKBX"
            },
            "creditorAgent": {
               "financialInstitutionIdentification": "GIBASKBX"
            },
          },
          "remittanceInformation": "Message for the receiver.",
          "additionalTransactionInformation": "string",
          "relatedDates": {
            "acceptanceDateTime": "2017-08-11"
          },
        },
      }
    ]
  }
```


5.2.7 Usage Example of AISP Operation: List of accounts

Process flow is visible in Figure 2: Implementation of AISP Services

4.1: HTTP Request example: GET /api/v1/accounts

Header


```
   Content-Type: application/json
   Authorization: Bearer         IDWJJBCHQ5DZJWEMO7ZWM4DLYWOFWKXXRequest-ID:
c2c48fc8-1f79-4934-a47b-56d61a28f351
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
   PSU-Last-Logged-Time:           2017-07-31T14:54:32+01:00PSU-IP-Address:
192.168.0.100
   PSU-Device-OS: iOS 11
   PSU-User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2228.0 Safari/537.36
   PSU-Geo-Location: 48.145745, 17.116062
```


Body


```
   {

   }
```

*4.2: HTTP Response example **201**: GET /api/v1/accounts*

Header

```
Content-Type: application/json
Response-ID: 7deb90a9-9900-4c90-a91c-3ecc888c2c88
Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
```

Body

```
{
  "creationDateTime": "2017-07-31T14:54:32+01:00",
  "accounts": [
    {
      "iban": "SK6807200002891987426353",
      "name": "John Doe",
      "productName": "BestAccount",
      "type": "CACC",
      "baseCurrency": "EUR",
      "servicer": "",
      "consent": ["AISP", "PISP"]
    }
  ]
}
```

6 PISP

Chapter defines list of services and alternative of flows provided for PISPs.

Prerequisites:

a) The TPP is registered for the PISP role and valid PISP scope
b) The TPP has been successfully authenticated
c) The TPP has presented its access token to call PISP services.

6.1 Endpoints definition

In following sections describe technical definition of provided endpoints for PISPs.

Endpoints	Method	Optionality	Description
/api/v1/**payments/standard/iso**	POST	Mandatory	**Standard payment initialization** – service allows to initialize payment in XML format (PAIN.001)
/api/v1/**payments/submission**	POST	Mandatory	**Standard payment submission** – service allows to authorization of initialized payment
/api/v1/**payments/{orderId}/status**	GET	Mandatory	**Payment order status** – service provide actual information about initialized payment
/api/v1/**payments/standard/sba**	POST	Optional	**Standard payment initialization** – service allows to initialize payment in JSON format
/api/v1/**payments/ecomm/iso**	POST	Optional	**Ecommerce payment initialization –** service allows to initializeimmediate payment in XML format (PAIN.001)
api/v1/**payments/ecomm/sba**	POST	Optional	**Ecommerce payment initialization** – service allows initialize immediate payment in JSON format

6.1.1 Standard header definition

Recommended set of request and response headers for PISP endpoints

Request header definition

Attribute	Optionality	Type	Description
Content-Type	Mandatory	String	application/json or application/xml
Authorization	Mandatory	String	Authorization is defined in RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage
Request-ID	Mandatory	String	An unique identifier of a particular request message. Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
Correlation-ID	Optional	String	An unique correlation identifier correlates the request and the response messages as a pair especially useful for audit logs. Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
Process-ID	Optional	String	Identifier of a business or technical process to what the set of requests and response pairs are organized (e.g. paging of transaction history should have same Process-ID). Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
PSU–IP-Address	Mandatory	String	Identifier of a customer's IP address from which he/she is connected to the TPP infrastructure. It might be in the format of IPv4 o IPv6 address. ASPSP shall indicate which values are acceptable.
PSU-Device-OS	Mandatory	String	A customer's device and/or operating system identification from which he/she is connected to the TPP infrastructure.
PSU-User-Agent	Mandatory	String	A customer's web browser of other client device identification from which he/she is connected to the TPP infrastructure. Agent header field of the http request between PSU and TPP.)
PSU-Geo-Location	Optional	String	The GPS coordinates of the current customer's location in the moment of connection to the TPP infrastructure. (Required GPS format: Latitude, Longitude)
PSU-Last-Logged-Time	Optional	DateTime	Last date and time when user was logged to TPP app (RFC3339 format)

Response header definition

Attribute	Optionality	Type	Description
Content-Type	Mandatory	String	application/json or application/xml
Response-ID	Mandatory	String	An unique identifier of a particular request message. Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
Correlation-ID	Optional	String	An unique correlation identifier correlates the request and the response messages as a pair especially useful for audit logs. Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
Process-ID	Optional	String	Identifier of a business or technical process to what the set of requests and response pairs are organized (e.g. paging of transaction history should have same Process-ID). Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).

HTTP PISP Request header example:

Header

```
   Content-Type: application/json
   Authorization: Bearer IDWJJBCHQ5DZJWEMO7ZWM4DLYWOFWKXX
   Request-ID: c2c48fc8-1f79-4934-a47b-56d61a28f351
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
   PSU-Last-Logged-Time: 2017-07-31T14:54:32+01:00
   PSU-IP-Address: 192.168.0.100
   PSU-Device-OS: iOS 11
   PSU-User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2228.0 Safari/537.36
   PSU-Geo-Location: 48.145745, 17.116062
```

HTTP PISP Response header example:

Header

```
   Content-Type: application/json
   Response-ID: ac30869e-29e2-40f7-83fb-ed1c6bdde216
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
```

6.1.2 PISP Operation: Standard payment initialization (XML)

The operation allows initialize payment in XML format (PAIN.001). The PISP sends a ISO20022 pain.001 based structure that specifies the payment activation request that is related to a commercial transaction between a PSU and the merchant.

Endpoint: POST /api/v1/payments/standard/iso

Request

Message contains xml: pain.001.001.03

- Link to message definition: https://www.iso20022.org/documents/general/Payments_Maintenance_2009.zip
- Link to message examples: https://www.iso20022.org/documents/messages/pain/instances/pain.001.001.03.zip

Response (if no error)

Message contains xml: pain.002.001.03

Attribute	XML structure mapping	Optionality	Type	Description
orderId	TxInfAndSts/ AcctSvcrRef	Mandatory	String (35)	**OrderId is** Unique reference, as assigned by the account servicing institution, to unambiguously identify the instruction.
status	TxInfAndSts/ TxSts	Mandatory	Enum	**Transaction status indicator** is enumeration: - ACTC (AcceptedTechnicalValidation) - ACWC (AcceptedWithChange) - RJCT (Rejected)
reasonCode	TxInfAndSts/ StsRsnInf/Rsn	Optional	Enum	ISO 20022 **Rejected Status Reason Code**
statusDateTime	GrpHdr/CreDtTm	Optional	Enum	**Transaction entry date**. The date of receiving the transaction in a bank.

- Link to definitions: https://www.iso20022.org/documents/general/Payments_Maintenance_2009.zip
- Link to message examples: https://www.iso20022.org/documents/messages/pain/instances/pain.002.001.03.zip
- Links to enumerations:
  Rejected Status Reason Code
  https://www.iso20022.org/standardsrepository/public/wqt/Description/mx/dico/codesets/_ZfQtyNp-Ed-ak6NoX_4Aeg_99789863

Error codes

Recommended set of HTTP Status codes and corresponding custom error codes:

HTTP Status	Error code	Description
400	**parameter_missing**	Mandatory parameter is missing
400	**parameter_invalid**	Value of input parameter is not valid
500, 503	**server_error**	Authorization server error.
Rest of HTTP Status codes and error codes are defined according to RFC 6749, Section 5.2		

6.1.3 PISP Operation: Standard payment submission

The operation provides authorization of initialized payment.

Endpoint: POST /api/v1/payments/submission

Request

The authorization header will contain a "bearer token" that coresppond to "payment order".

Response (if no error)

Attributes structure Level 1	Optionality	Type	Description
orderId	Mandatory	String	**OrderId is** Unique reference, as assigned by the account servicing institution, to unambiguously identify the instruction.
status	Mandatory	Enum	**Transaction status indicator** is enumeration: - ACTC (AcceptedTechnicalValidation) - ACWC (AcceptedWithChange) - RJCT (Rejected)
reasonCode	Optional	Enum	ISO 20022 **Rejected Status Reason Code**
statusDateTime	Mandatory	DateTime	**The date and time** in RFC3339 format at which a particular action has been requested or executed.

- Links to enumerations:
  Rejected Status Reason Code
  https://www.iso20022.org/standardsrepository/public/wqt/Description/mx/dico/codesets/_ZfQtyNp-Ed-ak6NoX_4Aeg_99789863

Error codes

Recommended set of HTTP Status codes and corresponding custom error codes:

HTTP Status	Error code	Description
400	**parameter_missing**	Mandatory parameter is missing
400	**parameter_invalid**	Value of input parameter is not valid
500, 503	**server_error**	Authorization server error.
Rest of HTTP Status codes and error codes are defined according to RFC 6749, Section 5.2		

6.1.4 PISP Operation: Payment order status

The operation provides information about processing status of a received payment instruction based on payment orderId identification.

Endpoint: GET /api/v1/payments/{orderId}/status

Request

Payload is empty.

Response (if no error)

Attributes structure Level 1	Optionality	Type	Description
orderId	Mandatory	String	**OrderId is** Unique reference, as assigned by the account servicing institution, to unambiguously identify the instruction.
status	Mandatory	Enum	**Transaction status indicator** is enumeration: - ACTC (AcceptedTechnicalValidation) - ACWC (AcceptedWithChange) - RJCT (Rejected) - PDNG (Pending) - ACSP (AcceptedSettlementInProcess) - ACSC (AcceptedSettlementCompleted)
reasonCode	Optional	Enum	ISO 20022 **Rejected Status Reason Code**
statusDateTime	Mandatory	DateTime	**The date and time** in RFC3339 format at which a particular action has been requested or executed.

- Links to enumerations:
  Rejected Status Reason Code
  https://www.iso20022.org/standardsrepository/public/wqt/Description/mx/dico/codesets/_ZfQtyNp-Ed-ak6NoX_4Aeg_99789863

Error codes

Recommended set of HTTP Status codes and corresponding custom error codes:

HTTP Status	Error code	Description
400	**parameter_missing**	Mandatory parameter is missing
400	**parameter_invalid**	Value of input parameter is not valid
500, 503	**server_error**	Authorization server error.
Rest of HTTP Status codes and error codes are defined according to RFC 6749, Section 5.2		

Expected flow of payment's states:

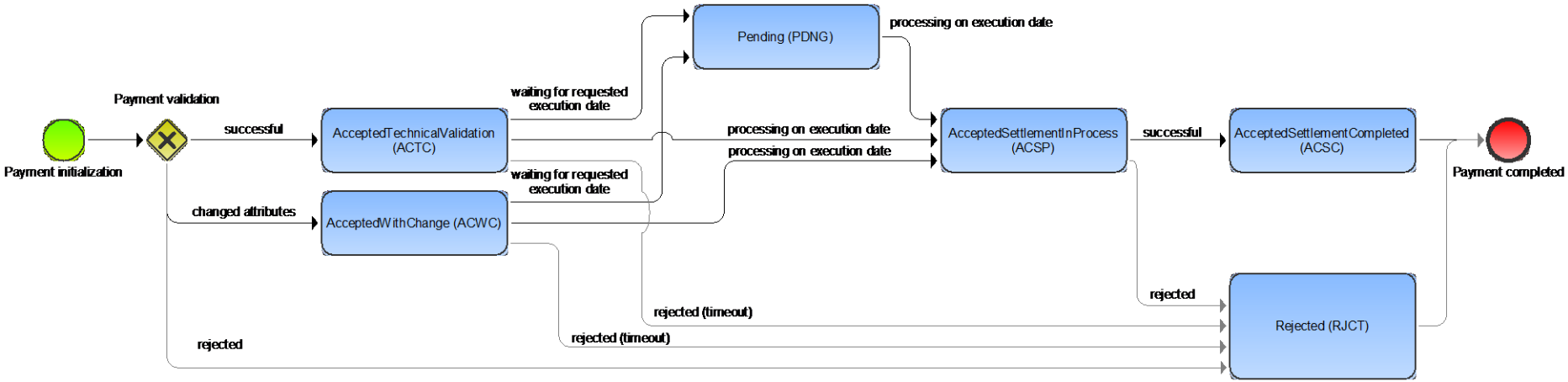

Figure 4: Flow of Payment's States

This operation provides following payment status codes:

Attribute	Description
ACTC	**AcceptedTechnicalValidation** - Authentication and syntactical and semantical validation are successful.
ACWC	**AcceptedWithChange** - Instruction is accepted but a change will be made, such as date or remittance not change
PDNG	**Pending** – payment initiation or individual transaction included in the payment initiation is pending. Further checks and status update will be perform.
ACSP	**AcceptedSettlementInProcess** - All preceding checks such as technical validation and customer profile were successful and therefore the payment initiation has been accepted for execution.
ACSC	**AcceptedSettlementCompleted** – Settlement on the debtor's account has been completed. Usage: this can by used by the first agent to reports to the debtor that the transaction has been completed. Warning: this status is provided for transaction status reasons, not for financial information. It can only be used after bilateral agreement.
RJCT	**Rejected** - Payment initiation or individual transaction included in the payment initiation has been rejected

- Link to definition: https://www.iso20022.org/standardsrepository/public/wqt/Description/mx/dico/codesets/_Z7RUV9p-Ed-ak6NoX_4Aeg_-481257913

6.1.5 Optional PISP Operation: Standard payment initialization (JSON)

The operation allows initialize payment in JSON format. The PISP sends JSON structure message based on ISO20022 pain.001.

Endpoint: POST /api/v1/payments/standard/sba

Request

Attributes structure		Optionality	Type	Description
Level 1	**Level 2**			
instructionIdentification		Mandatory	String [200]	**T echnical identification** of the payment generated by a PISP (or PSU).
creationDateTime		Optional	DateTime	**The date and time** in RFC3339 format at which a particular action has been requested or executed.
debtor	*name*	Mandatory	String [70]	**Debtor name** (first name and surname in case of individual persons or company name)
debtor	*iban*	Mandatory	String [34]	**Debtor account** International Bank Account Number (IBAN)
creditor	*name*	Mandatory	String [70]	**Creditor name** (first name and surname in case of individual persons or company name)
creditor	*iban*	Mandatory	String [34]	**Creditor account** International Bank Account Number (IBAN)
instructedAmount	*value*	Mandatory	Number Float [12.2]	**Transaction amount value** in account currency. Numeric value of the amount as a fractional number. The fractional part has a maximum of two digits.
instructedAmount	*currency*	Mandatory	String[3]	**Transaction amount currency**. Formated in Alphabetic codes from ISO 4712.
requestedExecutionDate		Mandatory	Date	**Expected execution date**
endToEndIdentification		Optional	String (35)	**Unique identification** defined by a requestor (PSU).
remittanceInformation		Optional	String (140)	The text aimed as the information for a receiver of the transaction.

Response (if no error)

Attributes structure Level 1	Optionality	Type	Description
orderId	Mandatory	String	**OrderId is** Unique reference, as assigned by the account servicing institution, to unambiguously identify the instruction.
status	Mandatory	Enum	**Transaction status indicator** is enumeration: - ACTC (AcceptedTechnicalValidation) - ACWC (AcceptedWithChange) - RJCT (Rejected) - PDNG (Pending) - ACSP (AcceptedSettlementInProcess) - ACSC (AcceptedSettlementCompleted)
reasonCode	Optional	Enum	ISO 20022 **Rejected Status Reason Code**
statusDateTime	Mandatory	DateTime	**The date and time** in RFC3339 format at which a particular action has been requested or executed.
request	Optional	String	Signed JWT - security mitigation for unauthorized payment request changes

- Links to enumerations:
  Rejected Status Reason Code
  https://www.iso20022.org/standardsrepository/public/wqt/Description/mx/dico/codesets/_ZfQtyNp-Ed-ak6NoX_4Aeg_99789863

Error codes

Recommended set of HTTP Status codes and corresponding custom error codes:

HTTP Status	Error code	Description
400	**parameter_missing**	Mandatory parameter is missing
400	**parameter_invalid**	Value of input parameter is not valid
500, 503	**server_error**	Authorization server error.
Rest of HTTP Status codes and error codes are defined according to RFC 6749, Section 5.2		

6.1.6 Optional PISP Operation: Ecommerce payment initialization (XML)

The operation allows initialize only payment with current values in XML format (PAIN.001). The PISP sends a ISO20022 pain.001 based structure that specifies the payment activation request. Successful authorizations of this type of payment lead to immediate transaction processing or funds reservation. This is the recommended payment type for performing e-commerce transactions.

Endpoint: POST /api/v1/payments/ecomm/iso

Request

Message contains xml: pain.001.001.03

- Link to message definition: https://www.iso20022.org/documents/general/Payments_Maintenance_2009.zip
- Link to message examples: https://www.iso20022.org/documents/messages/pain/instances/pain.001.001.03.zip

Response (if no error)

Message contains xml: pain.002.001.03

Attribute	XML structure mapping	Optionality	Type	Description
orderId	TxInfAndSts/ AcctSvcrRef	Mandatory	String (35)	**OrderId is** Unique reference, as assigned by the account servicing institution, to unambiguously identify the instruction.
status	TxInfAndSts/ TxSts	Mandatory	Enum	**Transaction status indicator** is enumeration: - ACTC (AcceptedTechnicalValidation) - ACWC (AcceptedWithChange) - RJCT (Rejected)
reasonCode	TxInfAndSts/ StsRsnInf/Rsn	Optional	Enum	ISO 20022 **Rejected Status Reason Code**
statusDateTime	GrpHdr/CreDtTm	Optional	Enum	**Transaction entry date**. The date of receiving the transaction in a bank.

- Link to definitions: https://www.iso20022.org/documents/general/Payments_Maintenance_2009.zip
- Link to message examples: https://www.iso20022.org/documents/messages/pain/instances/pain.002.001.03.zip
- Links to enumerations:
  Rejected Status Reason Code
  https://www.iso20022.org/standardsrepository/public/wqt/Description/mx/dico/codesets/_ZfQtyNp-Ed-ak6NoX_4Aeg_99789863

Error codes

Recommended set of HTTP Status codes and corresponding custom error codes:

HTTP Status	Error code	Description
400	**parameter_missing**	Mandatory parameter is missing
400	**parameter_invalid**	Value of input parameter is not valid
500, 503	**server_error**	Authorization server error.
Rest of HTTP Status codes and error codes are defined according to RFC 6749, Section 5.2		

6.1.7 Optional PISP Operation: Ecommerce payment initialization (JSON)

The operation allows initialize only payment with current values in JSON format. The PISP sends JSON structure message based on ISO20022 pain.001.that specifies the payment activation request. Successful authorization of this type of payment lead to immediate transaction processing or funds reservation. This is the recommended payment type for performing e-commerce transactions.

Endpoint: POST /api/v1/payments/ecomm/sba

Request

Attributes structure		Optionality	Type	Description
Level 1	Level 2			
instructionIdentification		Mandatory	String [200]	**Technical identification** of the payment generated by a PISP (or PSU).
creationDateTime		Optional	DateTime	**The date and time** in RFC3339 format at which a particular action has been requested or executed.
debtor	*name*	Mandatory	String [70]	**Debtor name** (first name and surname in case of individual persons or company name)
debtor	*iban*	Mandatory	String [34]	**Debtor account** International Bank Account Number (IBAN)
creditor	*name*	Mandatory	String [70]	**Creditor name** (first name and surname in case of individual persons or company name)
creditor	*iban*	Mandatory	String [34]	**Creditor account** International Bank Account Number (IBAN)
instructedAmount	*value*	Mandatory	Number Float [12.2]	**Transaction amount value** in account currency. Numeric value of the amount as a fractional number.
instructedAmount	*currency*	Mandatory	String [3]	**Transaction amount currency**. Formated in Alphabetic codes from ISO 4712.
endToEndIdentification		Optional	String (35)	**Unique identification** defined by a requestor (PSU).
remittanceInformation		Optional	String (140)	The text aimed as the information for a receiver of the transaction.

Response (if no error)

Attributes structure Level 1	Optionality	Type	Description
orderId	Mandatory	String	**OrderId is** Unique reference, as assigned by the account servicing institution, to unambiguously identify the instruction.
status	Mandatory	Enum	**Transaction status indicator** is enumeration: - ACTC (AcceptedTechnicalValidation) - ACWC (AcceptedWithChange) - RJCT (Rejected) - PDNG (Pending) - ACSP (AcceptedSettlementInProcess) - ACSC (AcceptedSettlementCompleted)
reasonCode	Optional	Enum	ISO 20022 **Rejected Status Reason Code**
statusDateTime	Mandatory	DateTime	**The date and time** in RFC3339 format at which a particular action has been requested or executed.

- Links to enumerations:
  Rejected Status Reason Code
  https://www.iso20022.org/standardsrepository/public/wqt/Description/mx/dico/codesets/_ZfQtyNp-Ed-ak6NoX_4Aeg_99789863

Error codes

Recommended set of HTTP Status codes and corresponding custom error codes:

HTTP Status	Error code	Description
400	**parameter_missing**	Mandatory parameter is missing
400	**parameter_invalid**	Value of input parameter is not valid
500, 503	**server_error**	Authorization server error.
Rest of HTTP Status codes and error codes are defined according to RFC 6749, Section 5.2		

6.2 Alternative flow implementation

Payment Initiation with Client Credentials Grant Type and OIDC Hybrid Flows:

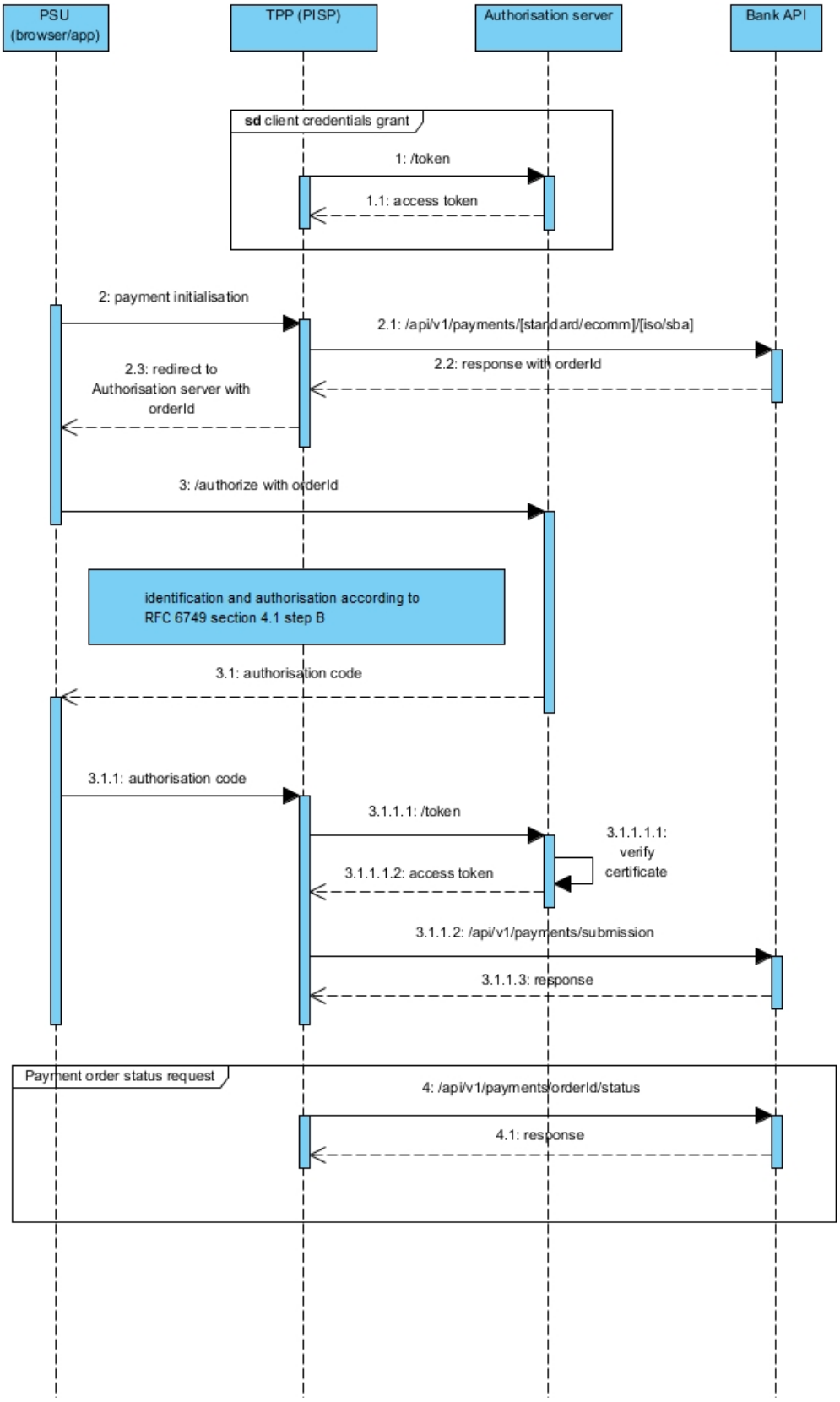

Figure 5: Implementation of PISP Services

6.2.1 Token for PISP services

STEP 1: To setup a single payment the Client Credentials Grant (according to RFC 6749, section 4.4) is used.The PISP initiates an Authorization request using valid Client Credentials Grant type and scope(s). The ASPSP Authorization Server validates the Client Authentication request from the PISP and generates an Access Token response where the request is valid

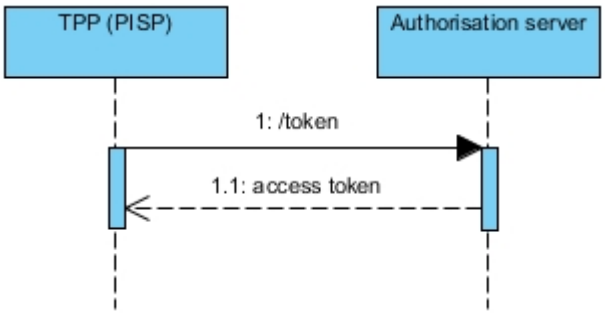

Figure 6: Token for PISP Services

PISP obtains an Access Token using a Client Credentials Grant Type with valid **client_id** and **client_secret** in authorization header. The scope PISP must be used. When an Access Token expires, the PISP will need to re-request for another Access Token using the same request below.

Request

Attribute	Optionality	Type	Description
grant_type	Mandatory		**client_credentials** exclusively to assign one-time **access_token**
scope	Mandatory		Required scope: "PISP"

Response

Attribute	Optionality	Type	Description
access_token			Short-term (one-time) token. This token is used to authorize the API request.
expires_in			The remaining time to expiration of **access_token** - in seconds.
token_type			Type of token „Bearer“
scope			"PISP"

Error codes

Error codes are defined according to RFC 6749, Section 5.2

1: HTTP Request example: POST /token

```
POST /token HTTP/1.1
Host: ib.banka.sk
Content-Type: application/x-www-form-urlencoded
Authorization: Basic BASE64(CLIENT_ID + ":" + CLIENT_SECRET)

grant_type=client_credentials&scope=PISP
```

1.1: HTTP Response example: POST /token

Headers

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Body

```
{
"access_token":"ACCESS_TOKEN_0",
"token_type":"bearer",
"expires_in":3600
"scope":"PISP"
}
```

The Client Credentials Grant may optionally be used by the PISP in Step 4 to retrieve the status of a Payment or Payment-Submission where no active Access Token is available

6.2.2 Usage Example of PISP Operation: Standard payment initialization (XML)

STEP 2: The PISP uses the Access Token (with PISP scope) from the ASPSP to invoke the Payments API resource against the ASPSP Resource Server. The ASPSP Resource server responds with the OrderId (and rest of data according specification).

2.1: HTTP Request example: POST /api/v1/payments/standard/iso

Header

```
Content-Type: application/xml
Authorization: Bearer IDWJJBCHQ5DZJWEMO7ZWM4DLYWOFWKXX
Request-ID: c2c48fc8-1f79-4934-a47b-56d61a28f351
Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
PSU-Last-Logged-Time: 2017-07-31T14:54:32+01:00
PSU-IP-Address: 192.168.0.100
PSU-Device-OS: iOS 11
PSU-User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2228.0 Safari/537.36
PSU-Geo-Location: 48.145745, 17.116062
```

Body

```
   <?xml version="1.0" encoding="UTF-8"?>
   <Document xmlns="urn:iso:std:iso:20022:tech:xsd:pain.001.001.03">
      <CstmrCdtTrfInitn>
            <GrpHdr>
                  <MsgId>MCCT1708164657382965</MsgId>
                  <CredDtTm>2017-08-16T14:08:36</CredDtTm>
                  <NbOfTxs>1</NbOfTxs>
                  <CtrlSum>1.75</CtrlSum>
                  <InitgPty>
                        <Nm>Company, a.s.</Nm>
                        <Id>
                              <OrgId>
                                    <Othr>
                                          <Id>ffdc2f2d-1288-4212-be38-
a011838ee051</Id>
                                    </Othr>
                              </OrgId>
                        </Id>
                  </InitgPty>
            </GrpHdr>
            <PmtInf>
                  <PmtInfId>17081600001</PmtInfId>
                  <PmtMtd>TRF</PmtMtd>
                  <PmtTpInf>
                        <InstrPrty>NORM</InstrPrty>
                        <SvcLvl>
                              <Cd>NURG</Cd>
                        </SvcLvl>
                        <CtgyPurp>
                              <Cd>SEPA</Cd>
                        </CtgyPurp>
                  </PmtTpInf>
                  <ReqdExctnDt>2017-08-16</ReqdExctnDt>
                  <Dbtr>
                        <Nm>Firm, a.s.</Nm>
                        <Id>
                              <OrgId>
                                    <Othr>
                                          <Id>123456</Id>
                                    </Othr>
                              </OrgId>
                        </Id>
                  </Dbtr>
                  <DbtrAcct>
                        <Id>
                              <Iban>SK6807200002891987426353</Iban>
                              <Othr>
                                    <Id>2891987426353/7200</Id>
                              </Othr>
                        </Id>
                        <Issr>Issuer</Issr>
                  </DbtrAcct>
                  <DbtrAgt>
                        <FinInstnId>
                              <BIC>SUBASKBX</BIC>
                        </FinInstnId>
                  </DbtrAgt>
                  <ChrgBr>SLEV</ChrgBr>
                  <CdtTrfTx>
                        <PmtId>
```

```
                        <InstrId>MCCT170816000005</InstrId>
                        <EndToEndId>NOTPROVIDED</EndToEndId>
                  </PmtId>
                  <Amt>
                        <InstdAmt>1.75</InstdAmt>
                        <Ccy>EUR</Ccy>
                  </Amt>
                  <CdtrAgt>
                        <FinInstnId>
                              <BIC>NOTPROVIDED</BIC>
                        </FinInstnId>
                  </CdtrAgt>
                  <Cdtr>
                        <Nm>NOTPROVIDED</Nm>
                        <Id>
                              <OrgId>
                                    <Othr>
                                          <Id>NOTPROVIDED</Id>
                                    </Othr>
                              </OrgId>
                        </Id>
                  </Cdtr>
                  <CdtrAcct>
                        <Id>
                              <Iban>SK6807200002891987426353</Iban>
                              <Othr>
                                    <Id>2891987426353/7200</Id>
                              </Othr>
                        </Id>
                        <Issr>Issuer</Issr>
                  </CdtrAcct>
                  <UltmtCdtr>
                        <Nm>Fero Skrutka</Nm>
                        <Id>
                              <OrgId>
                                    <Othr>
                                          <Id>654321</Id>
                                    </Othr>
                              </OrgId>
                        </Id>
                  </UltmtCdtr>
                  <Purp>
                        <Cd>ACCT</Cd>
                  </Purp>
                  <RmtInf>
                        <Ustrd>Payment for the goods</Ustrd>
                  </RmtInf>
            </CdtTrfTx>
      </PmtInf>
   </CstmrCdtTrfInitn>
</Document>
```

2.2: HTTP Response example: POST /api/v1/payments/standard/iso

Header

```
Content-Type: application/xml
Response-ID: 7deb90a9-9900-4c90-a91c-3ecc888c2c88
Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
```

Body

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns="urn:iso:std:iso:20022:tech:xsd:pain.002.001.03">
   <CstmrCdtTrfInitn>
         <GrpHdr>
               <MsgId>P002081617134122F1722800001731681</MsgId>
               <CredDtTm>2017-08-16T13:41:22+02:00</CredDtTm>
               <DbtrAgt>
                     <FinInstnId>
                           <BIC>SUBASKBX</BIC>
                     </FinInstnId>
               </DbtrAgt>
         </GrpHdr>
     <OrgnlGrpInfAndSts>
       <OrgnlMsgId> MCCT1708164657382965</OrgnlMsgId>
       <OrgnlMsgNmId>pain.001</OrgnlMsgNmId>
       <OrgnlCreDtTm>2017-08-16T14:08:36+02:00</OrgnlCreDtTm>
       <OrgnlNbOfTxs>1</OrgnlNbOfTxs>
       <OrgnlCtrlSum>1.75</OrgnlCtrlSum>
       <GrpSts>ACTC</GrpSts>
       <NbOfTxsPerSts>
         <DtldNbOfTxs>1</DtldNbOfTxs>
         <DtldSts>ACTC</DtldSts>
       </NbOfTxsPerSts>
     </OrgnlGrpInfAndSts>
     <OrgnlPmtInfAndSts>
       <OrgnlPmtInfId>17081600001</OrgnlPmtInfId>
       <OrgnlNbOfTxs>1</OrgnlNbOfTxs>
       <OrgnlCtrlSum>1.75</OrgnlCtrlSum>
       <PmtInfSts>ACTC</PmtInfSts>
       <TxInfAndSts>
         <StsId>1722810011766637</StsId>
         <OrgnlInstrId> MCCT170816000005</OrgnlInstrId>
         <OrgnlEndToEndId>NOTPROVIDED</OrgnlEndToEndId>
         <AcctSvcrRef>ffdc2f2d-1288-4212-be38-
a011838ee051</AcctSvcrRef>
         <TxSts>ACTC</TxSts>
         <StsRsnInf>
           <Orgtr>
             <Id>
               <OrgId>
                 <BICOrBEI>SUBASKBX</BICOrBEI>
               </OrgId>
             </Id>
           </Orgtr>
         </StsRsnInf>
         <OrgnlTxRef>
           <Amt>
             <InstdAmt Ccy="EUR">1.75</InstdAmt>
           </Amt>
           <ReqdExctnDt>2017-08-16</ReqdExctnDt>
           <PmtMtd>TRF</PmtMtd>
           <RmtInf>
```

```
            <Ustrd>Payment for the goods</Ustrd>
          </RmtInf>
          <Dbtr>
            <Nm>Company, a.s.</Nm>
          </Dbtr>
          <DbtrAcct>
            <Id>
              <IBAN>SK6807200002891987426353</IBAN>
            </Id>
          </DbtrAcct>
          <DbtrAgt>
            <FinInstnId>
              <BIC>SUBASKBX</BIC>
            </FinInstnId>
          </DbtrAgt>
          <CdtrAgt>
            <FinInstnId>
              <BIC>NOTPROVIDED</BIC>
            </FinInstnId>
          </CdtrAgt>
          <Cdtr>
            <Nm>NOTPROVIDED</Nm>
          </Cdtr>
          <CdtrAcct>
            <Id>
              <IBAN>SK6807200002891987426353</IBAN>
            </Id>
          </CdtrAcct>
        </OrgnlTxRef>
      </TxInfAndSts>
    </OrgnlPmtInfAndSts>
  </CstmrPmtStsRpt>
</Document>
```

6.2.3 Usage Example of PISP Operation: Standard payment submission

STEP 3: Payment authorization is initiated at the end of Step 2 by the PISP after the OrderId is generated by the ASPSP and returned to the PISP. This is used in a redirect across the PSU and ASPSP in Step 3 in order for the PSU to authorize the transaction..

The PISP creates an Authorization request (using a signed JWT Request containing the orderId as a claim) for the PSU to consent to the Payment directly with their ASPSP. The request is an OIDC Hybrid flow (requesting for Code and id_token) - for the PISP to proceed with the Payment by exchanging the Authorization Code for an Access Token in order to create the Payment-Submission.

3: HTTP Request example: GET /authorize

```
GET /authorize  HTTP/1.1
Host: ib.banka.sk
Content-Type: application/x-www-form-urlencoded

response_type=code id_token&
client_id=CLIENT_ID&
redirect_uri=REDIRECT_URI&
scope=payments openid&
state=STATE&
nonce=n-0S6_WzA2Mj&
code_challenge=BASE64URL-ENCODE(SHA256(ASCII(code_verifier)))&
code_challenge_method= S256&
request= CJleHAiOjE0OTUxOTk1ODd... JjVqsDuushgpwp0E.5leGFtcGxlI
iwianRpIjoiM...JleHAiOjE0.olnx_YKAm2J1rbpOP8wGhi1BDNHJjVqsDup0E
```

Note: All attributes are mandatory

Non-Base64 encoded example of the request parameter is defined by section

Signed JWT.

After the PSU has consented directly with the ASPSP via their web application (and confirmed the Debtor account) the ASPSP validates the Authorization request and generates an Auth Code and ID Token.

3.1: HTTP Response example: GET /authorize

```
   HTTP/1.1 303 See Other
   content-type: application/x-www-form-urlencoded
   Location:  REDIRECT_URI?code=AUTH_CODE&id_token=  eyJ0...NiJ9.eyJ1c  ...
I6IjIifX0.DeWt4Qu...ZXso&state=STATE
```

Note: Mandatory attributes: code, state

Non-Base64 encoded example of the id_token is defined by section Id_token.

The PSU is then redirected to the PISP. The PISP will now possess the Authorization Code and ID Token from the ASPSP. Note at this point, there is no Access Token. The PISP will now introspect the ID Token and use it as a detached signature to check:

- The hash of the Authorization Code to prove it hasn't been tampered with during redirect (comparing the hash value against the c_hash attribute in ID Token)
- The hash of the State to prove it hasn't been tampered with during redirect (comparing the state hash value against the s_hash attribute in the ID Token)

Once the state and code validations have been confirmed as successful by use of the ID token, the PISP will proceed to obtain an Access Token from the ASPSP using the Authorization Code they now possess. The PISP will present its Authorization Code.

3.1.1.1: HTTP Request example: POST /token

Headers

```
   POST /token HTTP/1.1
   Host: ib.banka.sk
   Content-Type: application/x-www-form-urlencoded
   Authorization: Basic BASE64(CLIENT_ID + ":" + CLIENT_SECRET)
```

Body

```
   grant_type=authorization_code&
   code=AUTH_CODE&
   redirect_uri=REDIRECT_URI&
   code_verifier=CODE_VERIFIER
```

Note: All attributes are mandatory

The Access Token is required by the PISP in order to submit the Payment on behalf of the PSU. The payments scope should already be associated with the Authorization Code generated in the previous step.

3.1.1.1.2: HTTP Response example: POST /token

Header

```
   HTTP/1.1 200 OK
   Content-Type: application/json;charset=UTF-8
```

Body

```
{
"access_token":"ACCESS_TOKEN_PAY",
"token_type":"bearer",
"expires_in": 600
}
```

Mandatory attributes: access_token, token_type, expires_in

The PISP has an Access Token which can be used to Create a Payment submission. The PISP must obtain the OrderId so that the Payment request is associated with the correct OrderId. OrderId is sourced from the **OrderId claim** of signed ID Token. The PISP will need to decode the ID Token JWT and locate the claim attribute associated with the OrderId.

The PISP can now invoke the payment submissions endpoint to commit the Payment using the Access Token and OrderId in the payload of the request.

3.1.1.2: HTTP Request example: POST /api/v1/payments/paymentSubmission

Header

```
   POST /api/v1/payments/paymentSubmission HTTP/1.1
   Host: ib.banka.sk
   Authorization: Bearer ACCESS_TOKEN_PAY

   Content-Type: application/json
   Request-ID: c2c48fc8-1f79-4934-a47b-56d61a28f351
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
   PSU-Last-Logged-Time: 2017-07-31T14:54:32+01:00
   PSU-IP-Address: 192.168.0.100
   PSU-Device-OS: iOS 11
   PSU-User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2228.0 Safari/537.36
   PSU-Geo-Location: 48.145745, 17.116062
```

Body

```
   {
   "orderId":"ffdc2f2d-1288-4212-be38-a011838ee051"
   }
```

3.1.1.3: HTTP Response: POST /api/v1/payments/paymentSubmission

Header

```
   HTTP/1.1 200 OK
   Content-Type: application/json;charset=UTF-8

   Response-ID: 7deb90a9-9900-4c90-a91c-3ecc888c2c88
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
```

Body

```
   {
   "orderId":"ffdc2f2d-1288-4212-be38-a011838ee051",
   "status":"ACTC",
   "statusDateTime":"2017-10-03T14:02:32.807Z"
   }
```

We provide the PaymentStatus method for completeness of the process.

6.2.4 Usage Example of PISP Operation: Payment order status

STEP 4: The PISP can query for the status of a Payment submission by invoking the payment submissions using the known OrderId. This can use an existing access token with payments scope or the PISP can obtain a fresh access token by replaying the client credentials grant request as per Step 1 – Setup Single Payment Initiation.

4: HTTP Request example: POST /api/v1/payments/paymentStatus

Header

```
POST /api/v1/payments/paymentStatus HTTP/1.1
Host: ib.banka.sk
Authorization: Bearer ACCESS_TOKEN_0

Content-Type: application/json;charset=UTF-8
Request-ID: c2c48fc8-1f79-4934-a47b-56d61a28f351
Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
PSU-Last-Logged-Time: 2017-07-31T14:54:32+01:00
PSU-IP-Address: 192.168.0.100
PSU-Device-OS: iOS 11
PSU-User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2228.0 Safari/537.36
PSU-Geo-Location: 48.145745, 17.116062
```

Body

```
{
"orderId":"ffdc2f2d-1288-4212-be38-a011838ee051"
}
```

All attributes are mandatory

4.1: HTTP Response example: POST /api/v1/payments/paymentStatus

Header

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Response-ID: 7deb90a9-9900-4c90-a91c-3ecc888c2c88
Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
```

Body

```
{
  "status": "RJCT",
  "reasonCode": "MONY",
  "statusDateTime": "2017-10-03T14:02:32.807Z"
}
```

6.2.5 Usage Example of PISP Operation: Standard payment initialization (JSON)

2.1: HTTP Request example: POST /api/v1/payments/standard/sba

Header

```
   Content-Type: application/json
   Authorization: Bearer IDWJJBCHQ5DZJWEMO7ZWM4DLYWOFWKXX
   Request-ID: c2c48fc8-1f79-4934-a47b-56d61a28f351
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
   PSU-Last-Logged-Time: 2017-07-31T14:54:32+01:00
   PSU-IP-Address: 192.168.0.100
   PSU-Device-OS: iOS 11
   PSU-User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2228.0 Safari/537.36
   PSU-Geo-Location: 48.145745, 17.116062
```

Body

```
   {
     "instructionIdentification": "9b766084-57de-48b2-be53-1bd2804ae0b7",
     "creationDateTime": "2017-07-31T14:54:32+01:00",
     "debtor": {
       "name": "John Doe",
       "iban": "SK6807200002891987426353"
     },
     "creditor": {
       "name": "John Doe",
       "iban": "SK6807200002891987426353"
     },
     "instructedAmount": {
       "value": 1234.56,
       "currency": "EUR"
     },
     "endToEndIdentification": "/VS123/SS456/KS0308",
     "remittanceInformation": "Payment for a utility service.",
     "requestedExecutionDate": "2017-08-11"
   }
```

2.2: HTTP Response example: POST /api/v1/payments/standard/sba

Header

```
   Content-Type: application/json
   Response-ID: 7deb90a9-9900-4c90-a91c-3ecc888c2c88
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
```

Body

```
   {
     "orderId": "ffdc2f2d-1288-4212-be38-a011838ee051",
     "status": "RJCT",
     "reasonCode": "MONY",
     "statusDateTime": "2017-10-04T11:59:27.350Z"
   }
```

6.2.6 Usage Example of PISP Operation: Ecommerce payment initialization (XML)

2.1: HTTP Request example: POST /api/v1/payments/ecomm/iso

Header

```
   Content-Type: application/xml
   Authorization: Bearer IDWJJBCHQ5DZJWEMO7ZWM4DLYWOFWKXX
   Request-ID: c2c48fc8-1f79-4934-a47b-56d61a28f351
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
   PSU-Last-Logged-Time: 2017-07-31T14:54:32+01:00
   PSU-IP-Address: 192.168.0.100
   PSU-Device-OS: iOS 11
   PSU-User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2228.0 Safari/537.36
   PSU-Geo-Location: 48.145745, 17.116062
```

Body

```
   <?xml version="1.0" encoding="UTF-8"?>
   <Document xmlns="urn:iso:std:iso:20022:tech:xsd:pain.001.001.03">
      <CstmrCdtTrfInitn>
            <GrpHdr>
                  <MsgId>MCCT1708164657382965</MsgId>
                  <CredDtTm>2017-08-16T14:08:36</CredDtTm>
                  <NbOfTxs>1</NbOfTxs>
                  <CtrlSum>1.75</CtrlSum>
                  <InitgPty>
                        <Nm>Company, a.s.</Nm>
                        <Id>
                              <OrgId>
                                    <Othr>
                                          <Id>4748027</Id>
                                    </Othr>
                              </OrgId>
                        </Id>
                  </InitgPty>
            </GrpHdr>
            <PmtInf>
                  <PmtInfId>17081600001</PmtInfId>
                  <PmtMtd>TRF</PmtMtd>
                  <PmtTpInf>
                        <InstrPrty>NORM</InstrPrty>
                        <SvcLvl>
                              <Cd>NURG</Cd>
                        </SvcLvl>
                        <CtgyPurp>
                              <Cd>SEPA</Cd>
                        </CtgyPurp>
                  </PmtTpInf>
                  <ReqdExctnDt>2017-08-16</ReqdExctnDt>
                  <Dbtr>
                        <Nm>Firm, a.s.</Nm>
                        <Id>
                              <OrgId>
                                    <Othr>
                                          <Id>123456</Id>
                                    </Othr>
                              </OrgId>
                        </Id>
```

```
                </Dbtr>
                <DbtrAcct>
                      <Id>
                            <Iban>SK6807200002891987426353</Iban>
                            <Othr>
                                  <Id>2891987426353/7200</Id>
                            </Othr>
                      </Id>
                      <Issr>Issuer</Issr>
                </DbtrAcct>
                <DbtrAgt>
                      <FinInstnId>
                            <BIC>SUBASKBX</BIC>
                      </FinInstnId>
                </DbtrAgt>
                <ChrgBr>SLEV</ChrgBr>
                <CdtTrfTx>
                      <PmtId>
                            <InstrId>MCCT170816000005</InstrId>
                            <EndToEndId>NOTPROVIDED</EndToEndId>
                      </PmtId>
                      <Amt>
                            <InstdAmt>1.75</InstdAmt>
                            <Ccy>EUR</Ccy>
                      </Amt>
                      <CdtrAgt>
                            <FinInstnId>
                                  <BIC>NOTPROVIDED</BIC>
                            </FinInstnId>
                      </CdtrAgt>
                      <Cdtr>
                            <Nm>NOTPROVIDED</Nm>
                            <Id>
                                  <OrgId>
                                        <Othr>
                                              <Id>NOTPROVIDED</Id>
                                        </Othr>
                                  </OrgId>
                            </Id>
                      </Cdtr>
                      <CdtrAcct>
                            <Id>
                                  <Iban>SK6807200002891987426353</Iban>
                                  <Othr>
                                        <Id>2891987426353/7200</Id>
                                  </Othr>
                            </Id>
                            <Issr>Issuer</Issr>
                      </CdtrAcct>
                      <UltmtCdtr>
                            <Nm>Fero Skrutka</Nm>
                            <Id>
                                  <OrgId>
                                        <Othr>
                                              <Id>654321</Id>
                                        </Othr>
                                  </OrgId>
                            </Id>
                      </UltmtCdtr>
                      <Purp>
                            <Cd>ACCT</Cd>
```

```
                        </Purp>
                        <RmtInf>
                              <Ustrd>Payment for the goods</Ustrd>
                        </RmtInf>
                  </CdtTrfTx>
            </PmtInf>
      </CstmrCdtTrfInitn>
   </Document>
```

2.2: HTTP Response example: POST /api/v1/payments/ecomm/iso

Header

```
   Content-Type: application/xml
   Response-ID: 7deb90a9-9900-4c90-a91c-3ecc888c2c88
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
```

Body

```
   <?xml version="1.0" encoding="UTF-8"?>
   <Document xmlns="urn:iso:std:iso:20022:tech:xsd:pain.002.001.03">
      <CstmrCdtTrfInitn>
            <GrpHdr>
                  <MsgId>P002081617134122F1722800001731681</MsgId>
                  <CredDtTm>2017-08-16T13:41:22+02:00</CredDtTm>
                  <DbtrAgt>
                        <FinInstnId>
                              <BIC>SUBASKBX</BIC>
                        </FinInstnId>
                  </DbtrAgt>
            </GrpHdr>
       <OrgnlGrpInfAndSts>
         <OrgnlMsgId> MCCT1708164657382965</OrgnlMsgId>
         <OrgnlMsgNmId>pain.001</OrgnlMsgNmId>
         <OrgnlCreDtTm>2017-08-16T14:08:36+02:00</OrgnlCreDtTm>
         <OrgnlNbOfTxs>1</OrgnlNbOfTxs>
         <OrgnlCtrlSum>1.75</OrgnlCtrlSum>
         <GrpSts>ACTC</GrpSts>
         <NbOfTxsPerSts>
           <DtldNbOfTxs>1</DtldNbOfTxs>
           <DtldSts>ACTC</DtldSts>
         </NbOfTxsPerSts>
       </OrgnlGrpInfAndSts>
       <OrgnlPmtInfAndSts>
         <OrgnlPmtInfId>17081600001</OrgnlPmtInfId>
         <OrgnlNbOfTxs>1</OrgnlNbOfTxs>
         <OrgnlCtrlSum>1.75</OrgnlCtrlSum>
         <PmtInfSts>ACTC</PmtInfSts>
         <TxInfAndSts>
           <StsId>1722810011766637</StsId>
           <OrgnlInstrId> MCCT170816000005</OrgnlInstrId>
           <OrgnlEndToEndId>NOTPROVIDED</OrgnlEndToEndId>
           <AcctSvcrRef>ffdc2f2d-1288-4212-be38-
a011838ee051</AcctSvcrRef>
           <TxSts>ACTC</TxSts>
           <StsRsnInf>
             <Orgtr>
               <Id>
                 <OrgId>
                   <BICOrBEI>SUBASKBX</BICOrBEI>
                 </OrgId>
```

```
            </Id>
          </Orgtr>
        </StsRsnInf>
        <OrgnlTxRef>
          <Amt>
            <InstdAmt Ccy="EUR">1.75</InstdAmt>
          </Amt>
          <ReqdExctnDt>2017-08-16</ReqdExctnDt>
          <PmtMtd>TRF</PmtMtd>
          <RmtInf>
            <Ustrd>Payment for the goods</Ustrd>
          </RmtInf>
          <Dbtr>
            <Nm>Company, a.s.</Nm>
          </Dbtr>
          <DbtrAcct>
            <Id>
              <IBAN>SK6807200002891987426353</IBAN>
            </Id>
          </DbtrAcct>
          <DbtrAgt>
            <FinInstnId>
              <BIC>SUBASKBX</BIC>
            </FinInstnId>
          </DbtrAgt>
          <CdtrAgt>
            <FinInstnId>
              <BIC>NOTPROVIDED</BIC>
            </FinInstnId>
          </CdtrAgt>
          <Cdtr>
            <Nm>NOTPROVIDED</Nm>
          </Cdtr>
          <CdtrAcct>
            <Id>
              <IBAN>SK6807200002891987426353</IBAN>
            </Id>
          </CdtrAcct>
        </OrgnlTxRef>
      </TxInfAndSts>
    </OrgnlPmtInfAndSts>
  </CstmrPmtStsRpt>
</Document>
```

6.2.7 Usage Example of PISP Operation: Ecommerce payment initialization (JSON)

2.1: HTTP Request example: POST /api/v1/payments/ecomm/sba

Header

```
   Content-Type: application/json
   Authorization: Bearer IDWJJBCHQ5DZJWEMO7ZWM4DLYWOFWKXX
   Request-ID: c2c48fc8-1f79-4934-a47b-56d61a28f351
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
   PSU-Last-Logged-Time: 2017-07-31T14:54:32+01:00
   PSU-IP-Address: 192.168.0.100
   PSU-Device-OS: iOS 11
   PSU-User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2228.0 Safari/537.36
   PSU-Geo-Location: 48.145745, 17.116062
```

Body

```
   {
     "instructionIdentification": "9b766084-57de-48b2-be53-1bd2804ae0b7",
     "creationDateTime": "2017-07-31T14:54:32+01:00",
     "debtor": {
       "name": "John Doe",
       "iban": "SK6807200002891987426353"
     },
     "creditor": {
       "name": "John Doe",
       "iban": "SK6807200002891987426353"
     },
     "instructedAmount": {
       "value": 1234.56,
       "currency": "EUR"
     },
     "endToEndIdentification": "/VS123/SS456/KS0308",
     "remittanceInformation": "Payment for a utility service."
   }
```

2.2: HTTP Response example: POST /api/v1/payments/ecomm/sba

Header

```
   Content-Type: application/json
   Response-ID: 7deb90a9-9900-4c90-a91c-3ecc888c2c88
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
```

Body

```
   {
     "orderId": "ffdc2f2d-1288-4212-be38-a011838ee051",
     "status": "RJCT",
     "reasonCode": "MONY",
     "statusDateTime": "2017-10-04T11:59:27.350Z"
   }
```

6.2.8 Signed JWT

JWT contains:

- JOSE Header – according to JWT(rfc7519), JWS(rfc7515)
- jwt
- signature – according to JWS(rfc7515)

Non-Base64 encoded example of the request parameter object:

```
{
  "alg": "RS256",
  "kid": "GxlIiwianVqsDuushgjE0OTUxOTk",
  "typ": "JWT"
}
.
{
   "iss": "s6BhdRkqt3",
   "aud": "https://login.bank123.com",
   "response_type": "code id_token",
   "client_id": "s6BhdRkqt3",
   "redirect_uri": "https://api.mytpp.com/cb",
   "scope": "openid payments",
   "state": "af0ifjsldkj",
   "nonce": "n-0S6_WzA2Mj",
   "max_age": 86400,
   "claims":
    {
     "id_token":
      {
       "orderId":  {"value":  "urn:  bank123:order:58923",  "essential":
true},
      }
    }
}
.
<<signature>>
```

6.2.9 Id_token

JWT contains:

- JOSE Header – according to JWT(rfc7519), JWS(rfc7515)
- jwt
- signature – according to JWS(rfc7515)

Non-Base64 encoded example of the id_token:

```
{
  "alg": "RS256",
  "kid": "GxlIiwianVqsDuushgjE0OTUxOTk",
  "typ": "JWT"
}
.
{
   "iss": "https://login.bank123.com",
   "aud": "s6BhdRkqt3",
   "iat": "1234569795",
   "sub": " bank123:order:58923",
   "orderId": "urn:bank123:order:58923",
   "aud": "s6BhdRkqt3",
   "nonce": "n-0S6_WzA2Mj",
   "exp": 1311281970,
   "s_hash": "76sa5dd",
   "c_hash": "asd097d"
  }
.
{
<<signature>>
}
```

7 PIISP

Chapter defines list of services and alternative of flows provided for PIISPs.

Prerequisites:

a) The TPP is registered for the PIISP role and valid PIISP scope
b) The TPP has been successfully authenticated
c) The TPP has presented its "OAuth2 Authorization Client Credential Grant" access token which allows the ASPSP to identify the TPP

7.1 Endpoints definition

In following sections describe technical definition of provided endpoints for PIISPs.

Endpoint	Method	Optionality	Description
/api/v1**/accounts/balanceCheck**	POST	Mandatory	**Balance check** – service provide information about sufficient balance with the yes/no answer

7.1.1 Standard header definition

Recommended set of request and response headers for PIISP endpoints

Request header definition

Attribute	Optionality	Type	Description
Content-Type	Mandatory	String	application/json or application/xml
Authorization	Mandatory	String	Authorization is defined in RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage
Request-ID	Mandatory	String	An unique identifier of a particular request message. Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
Correlation-ID	Optional	String	An unique correlation identifier correlates the request and the response messages as a pair especially useful for audit logs. Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
Process-ID	Optional	String	Identifier of a business or technical process to what the set of requests and response pairs are organized (e.g. paging of transaction history should have same Process-ID). Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
PSU–IP-Address	Mandatory	String	Identifier of a customer's IP address from which he/she is connected to the TPP infrastructure. It might be in the format of IPv4 o IPv6 address. ASPSP shall indicate which values are acceptable.
PSU-Device-OS	Mandatory	String	A customer's device and/or operating system identification from which he/she is connected to the TPP infrastructure.
PSU-User-Agent	Mandatory	String	A customer's web browser of other client device identification from which he/she is connected to the TPP infrastructure. Agent header field of the http request between PSU and TPP.)
PSU-Geo-Location	Optional	String	The GPS coordinates of the current customer's location in the moment of connection to the TPP infrastructure. (Required GPS format: Latitude, Longitude)
PSU-Last-Logged-Time	Optional	DateTime	Last date and time when user was logged to TPP app (RFC3339 format)

Response header definition

Attribute	Optionality	Type	Description
Content-Type	Mandatory	String	application/json or application/xml
Response-ID	Mandatory	String	An unique identifier of a particular request message. Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
Correlation-ID	Optional	String	An unique correlation identifier correlates the request and the response messages as a pair especially useful for audit logs. Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).
Process-ID	Optional	String	Identifier of a business or technical process to what the set of requests and response pairs are organized (e.g. paging of transaction history should have same Process-ID). Although it may be arbitrary string, it is strongly recommended to use a Universally Unique Ifentifier (UUID) version 4 form (RFC4122).

HTTP PIISP Request header example:

Header

```
   Content-Type: application/json
   Authorization: Bearer IDWJJBCHQ5DZJWEMO7ZWM4DLYWOFWKXX
   Request-ID: c2c48fc8-1f79-4934-a47b-56d61a28f351
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
   PSU-Last-Logged-Time: 2017-07-31T14:54:32+01:00
   PSU-IP-Address: 192.168.0.100
   PSU-Device-OS: iOS 11
   PSU-User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2228.0 Safari/537.36
   PSU-Geo-Location: 48.145745, 17.116062
```

HTTP PIISP Response header example:

Header

```
   Content-Type: application/json
   Response-ID: ac30869e-29e2-40f7-83fb-ed1c6bdde216
   Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
   Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
```

7.1.2 PIISP Operation: Balance check

The operation provides the resolution whether the balance of a bank customer's account identified by IBAN is sufficient for asked amount.

Endpoint: POST /api/v1/accounts/balanceCheck

Request

Attributes structure			Optionality	Type	Description
Level 1	**Level 2**	**Level 3**			
instructionIdentification			Mandatory	String	**Technical identification of payment**, generated by the PIISP
creationDateTime			Optional	DateTime	**The date and time** in RFC3339 format at which a particular action has been requested or executed.
iban			Mandatory	String [34]	**International Bank Account Number** (IBAN)
amount	*value*		Mandatory	Number Float [12.2]	**Transaction amount value** in account currency. Numeric value of the amount as a fractional number.
amount	*currency*		Mandatory	String [3]	**Transaction amount currency**. Formated in Alphabetic codes from ISO 4712.
relatedParties	*tradingParty*	*identification*	Optional	String [35]	Unique identification of a third party. For card transaction, this is ID of merchant.
relatedParties	*tradingParty*	*name*	Optional	String [140]	Name of a third party. For card transaction, this is the name of merchant.
relatedParties	*tradingParty*	*address*	Optional	String [70]	**Merchant cummulative address** identification usually containing concatenation of street name, street number, etc.
relatedParties	*tradingParty*	*countryCode*	Optional	String [2]	The two letter **merchant country code** adopted from ISO3166.
relatedParties	*tradingParty*	*merchantCode*	Optional	String [4]	A **Merchant Category Code** (MCC) coordinated by MasterCard and Visa.
references	*chequeNumber*		Optional	String [35]	**For card transactions**, this is the card number in format **** **** **** 1111
references	*holderName*		Optional	String[35]	**Card holder name**

Response (if no error)

Attributes structure	Optionality	Type	Description
Level 1			
response	Mandatory	Enum	**response** is enumeration: - APPR (sufficient funds on the account) - DECL (insufficient funds in the account)
dateTime	Mandatory	DateTime	**The date and time** in RFC3339 format at which a particular action has been requested or executed.

Error codes

Recommended set of HTTP Status codes and corresponding custom error codes:

HTTP Status	Error code	Description
400	**parameter_missing**	Mandatory parameter is missing
400	**parameter_invalid**	Value of input parameter is not valid
500, 503	**server_error**	Authorization server error.
Rest of HTTP Status codes and error codes are defined according to RFC 6749, Section 5.2		

7.2 Alternative flow implementation

To confirm the availability of funds under Article 65, of the Directive the TPP will use the generated **access_token** according to RFC 6749, Section 4.4. Client credentials grant.

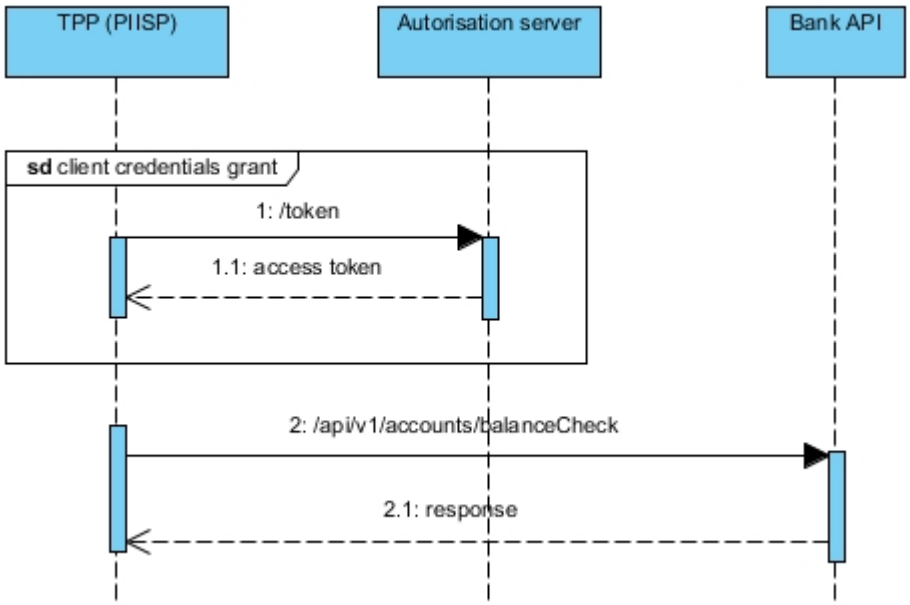

Figure 7: Implementation of PIISP Services

7.2.1 Token for PIISP services

To initialize the payment, or the one-time **access_token** according to RFC 6749, section 4.4, (Client Credentials Grant) is used with valid **client_id** and **client_secret** in authorization header.

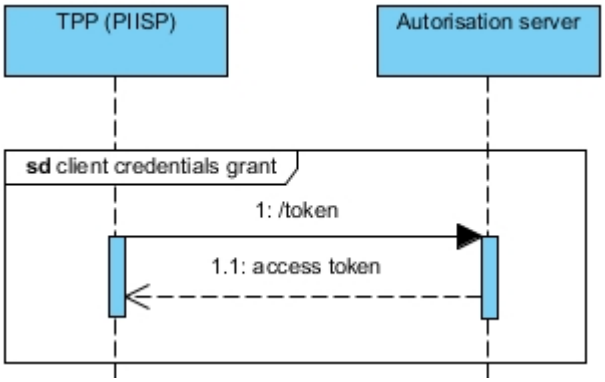

Figure 8: Token for PIISP Services

Request

Attribute	Optionality	Type	Description
grant_type	Mandatory		**client_credentials** exclusively to assign one-time **access_token**
scope	Mandatory		Required scope: "PIISP"

Response

Attribute	Optionality	Type	Description
access_token			Short-term (one-time) token. This token is used to authorize the API request.
expires_in			The remaining time to expiration of **access_token** - in seconds.
token_type			Type of token „Bearer“
scope			"PIISP"

Error codes

Error codes are defined according to RFC 6749, Section 5.2

7.2.2 Usage Example of PIISP Operation: Balance check

Process flow is visible in Figure 7: Implementation of PIISP Services

1: HTTP Request example: POST /token

```
POST /token HTTP/1.1
Host: ib.banka.sk
Content-Type: application/x-www-form-urlencoded
Authorization: Basic BASE64(CLIENT_ID + ":" + CLIENT_SECRET)

grant_type=client_credentials&scope=PIISP
```

1.1: HTTP Response example: POST /token

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
"access_token":"ACCESS_TOKEN_0",
"expires_in":3600,
"token_type":"bearer",
"scope":"PIISP"
}
```

2: HTTP Request example: POST /api/v1/accounts/balanceCheck

Header

```
Content-Type: application/json
Authorization: Bearer IDWJJBCHQ5DZJWEMO7ZWM4DLYWOFWKXX
Request-ID: c2c48fc8-1f79-4934-a47b-56d61a28f351
Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
PSU-IP-Address: 192.168.0.100
PSU-Device-OS: iOS 11
PSU-User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2228.0 Safari/537.36
PSU-Geo-Location: 48.145745, 17.116062
```

Body

```
{
  "instructionIdentification": "9b766084-57de-48b2-be53-1bd2804ae0b7",
  "creationDateTime": "2017-07-31T14:54:32+01:00",
  "iban": "SK6807200002891987426353",
  "amount": {
    "value": 1234.56,
    "currency": "EUR"
  },
  "relatedParties": {
    "tradingParty": {
      "identification": "AAA-GG-SSSS",
      "name": "Jane Doe Company",
      " adress": "My street 123, MyLand",
      "countryCode": "SK",
      "merchantCode": "3370"
    },
  },
  "references": {
    "chequeNumber": "123456******3456",
    "holderName": "Jane Doe"
  }
}
```

2.1: HTTP Response example: POST /api/v1/accounts/balanceCheck

Headers

```
HTTP/1.1 200 OK
Content-Type: application/json
Response-ID: 7deb90a9-9900-4c90-a91c-3ecc888c2c88
Correlation-ID: 292163f5-4eee-4447-9292-5672fdf0013b
Process-ID: 4b88bf95-e129-42b8-a17d-1d2379810fbe
```

Body

```
{
  "result": "APPR",
  "creationDateTime": "2017-07-31T14:54:32+01:00"
}
```

8 Bibliography

1. *RFC 6749 - The OAuth 2.0 Authorization Framework*, [online]. The Internet Engineering Task Force, October 2012. Available from WWW: https://tools.ietf.org/html/rfc6749

2. *RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage,* [online]. The Internet Engineering Task Force, October 2012. Available from WWW: https://tools.ietf.org/html/rfc6750

3. *RFC 7636 - Proof Key for Code Exchange by OAuth Public Clients,* [online]. The Internet Engineering Task Force, September 2015. Available from WWW: https://tools.ietf.org/html/rfc7636

4. *OpenID - OpenID Connect Core 1.0 incorporating errata set 1*, [online]. N. Sakimura et al., November 2014. Available from WWW: http://openid.net/specs/openid-connect-core-1_0.html

5. *RFC 7519 - JSON Web Token (JWT)*, [online]. The Internet Engineering Task Force, May 2015. Available from WWW: https://tools.ietf.org/html/rfc7519

6. *RFC 7515 - JSON Web Signature (JWS)*, [online]. The Internet Engineering Task Force, May 2015. Available from WWW: https://tools.ietf.org/html/rfc7515

7. *EVCG - CA Browser Forum: "Guidelines for The Issuance and Management of Extended Validation,* [online].

8. *EV SSL Certificate Guidelines*, [online]. Available from WWW: https://cabforum.org/extended-validation/

9. *OpenAPI Specification, version 2.0.0*, [online]. Available from WWW: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md

10. *Slovak Banking API Standard, SBA et al.,* [online]. Available from WWW: http://docs.sbaonline.apiary.io/#

11. *The JavaScript Object Notation (JSON) Data Interchange Format*, [online]. The Internet Engineering Task Force, March 2014. Available from WWW: https://tools.ietf.org/html/rfc7159

12. *YAML Ain't Markup Language (YAML™) Version 1.2*, [online]. Available from WWW: http://www.yaml.org/spec/1.2/spec.html

13. *Semantic Versioning 2.0.0*, [online]. Tom Preston-Werner. Available from WWW: http://semver.org/

14. *Hypertext Transfer Protocol -- HTTP/1.1***,** [online]. The Internet Engineering Task Force, June 1999. Available from WWW: http://tools.ietf.org/html/rfc2616

15. *ISO 20022 Financial Services - Universal financial industry message scheme,* [online]. International Organization for Standardization. Available from WWW: https://www.iso20022.org/

9 Apendix A

1. Organization of standard documents
   a) The standard is described by one standalone OpenAPI document (formerly known as "swagger") utilizing the version 2.0.0 of the OpenAPI specification [9] available in those formats:
      - JSON [11]
      - YAML [12]
   b) The standard declares all service operations, mandatory as well as optional ones that form the API alongside with embedded data model in form of JSON schema.
   c) The standard is published at publicly accessible at internet location [10].
   d) The standard document should not be split by optionality or roles of TTPs into more documents.
      - No splitting is required for service operations. The entire web service with all standardized service operations provided by an ASPSP should be described just in one OpenAPI document.
      - The data model of the standard may be externalized into a separate document publicly accessible.
      - For grouping service operations of the standard by optionality and/or roles of TPPs should be the tags described in the OpenAPI specification [9] employed.
2. Schema externalization for sharing
   a) Despite the fact that the standard document should not be split by service operations, the data model of the standardized API may be externalized into a separate document.
      - The situation is similar to service oriented architecture, where an entire interface of a web service describes usually one Web Service Description Language document, which imports or includes one or more Extensible Markup Language schema definition document describing a data model of that interface.
   b) Data model of the standard can be externalized either in JSON [11] or YAML [12] format, preferably in both of them (hereinafter referred as "standard schema").
   c) If an ASPSP provides extended API according the Section Recommended form of ASPSP web services extension, its extended API should reference and use standard schema as much as reasonable.
      - An ASPSP should not create new data elements in its schema with the same semantics as some element defined in the standard schema already has in order not to break principles from Section Design principles for APIs, especially semantic messaging prohibition.
      - An ASPSP should extend the data model of its extended API by extending the standard schema.
   d) In order to enable public availability and access to the standard schema, eventually to an extended schema, those should be hosted on a publicly accessible HTTP server with the feature Cross Origin Resource Sharing enabled.