VÚB BANKA
Intesa Sanpaolo Group

Berlin Group NextGenPSD2

Implementation guidelines for Third party providers

Version	1.5
Status	Approved
Author(s)	developers@vub.sk

The following Terms and Abbreviations have the meaning ascribed to them below, for the purposes of this document:

Abbreviation	Description
API	Application Programming Interface
AIS	Account Information Service
AISP	Account Information Service Provider
ASPSP	Account Servicing Payment Service Provider
Authentication	Identity confirmation
Authorization	Verification of access to ASPSP resources
BGS	Berlin Group NextGenPSD2 standard
Certificate	A qualified certificate in the sense of e-IDAS. After the SCA RTS has been applied, it will be the only acceptable certificate for authentication of TPP
Directive / PSD2	PSD2 Directive. Directive of the European Parliament and of the Council (EU) No. 2015/2366 In Slovak republic, Directive was implemented into the Slovak Act No. 492/2009 Coll. on payment services as amended
EV	Extended Validation certificate
IBAN	International Bank Account Number
JOSE	JSON Object Signing and Encryption.
JSON	JavaScript Object Notation
OAuth	Open Authorization protocol
OIDC	OpenID Connect
Optional input parameter	TPP can ignore this parameter
Optional output parameter	The ASPSP may fill the parameter value
PIIS	Payment Instrument Issuer Service
PIISP	Payment Instrument Issuer Service Provider
PIS	Payment Initiation Service
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 European Banking Authority
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)

Abbreviation	Description
	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.
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.
VUB	Všeobecná úverová banka, a.s., Mlynské nivy 1, 829 90 Bratislava 25, registered with Companies Registry of the District Court Bratislava I, Section: Sa, Insertion No.: 341/B, BIC: SUBASKBX, Identification No.: 31 320 155

LIST OF CONTENTS

LIST OF TABLES

1 DISCLAIMER

The presented documentation describes the Application Programming Interface (API) provided by VUB to PSD2 licensed Third Parties Providers (TPPs) that provides the Payment Initiation Services, Account Information Services and issue the card-based payment instruments. The documentation is provided for purposes of integration of TPPs applications with VUB´s API and describes the prerequisites needed for connection to VUB API, data flows, endpoints and attributes of API.

The documentation may be used only for the purposes described above. Any breach of this obligation may be subject to damage recovery and penalties under Civil Code, Commercial Code, Criminal Law or any other applicable laws and regulations of the Slovak Republic.

2 GOAL OF THE DOCUMENT

Main aim of the document is to clarify and explain VUB's implementation of BGS, its provided API and flows to PSD2 licensed TPPs.

VUB provides bank interfaces to other licensed TPPs. Developers of TPPs can integrate their applications with VUB's API and gain access to the payment accounts held by VUB, that are available online, upon PSU's consent. These services do not replace existing bank services.

This document describes prerequisites needed for connection to VUB's API, data flows among all participated parties, endpoints and attributes of API. Each API is described by OpenAPI specification in the form of Swagger file.

This document does not define provided API in detail. Each API is described in Berlin Group documentation "NextGenPSD2 Access to Account Interoperability Framework - Implementation Guidelines".

3 BEFORE YOU START

APIs provided by the VUB are intended to be used by TPPs, who are subjects of the license for:

- *Payment Initiation Service Provider* (PISP) - subject has a license for providing payment initiation services to *Payment Service Users* (PSU)
- *Account Information Service Provider* (AISP) - subject has a license for providing account information services to *Payment Service Users* (PSU)
- *Payment Instrument Issuer Service Provider* (PIISP) - subject has a license for providing PIIS services to *Payment Service Users* (PSU)

License can be obtained by relevant national regulator (Slovak National Bank for companies with its seat in Slovak Republic).

Mutual Transport Layer Security (MTLS) is used for secure communication between *Third Party Provider* and *Account Services Payment Service Providers* (ASPSP) – Bank/ VUB. Each TPP, who wants to use VUB API has to provide eIDAS, or EV certificate until RTS SCA has been applied, signed by trusted *Certification Authority* (CA).

After successful TPP PSD2 registration by National regulator, the TPP obtains license number and PKI certificate, which contains the license number (obtained by trusted Certification Authority). Since this point, the TPP is allowed to perform communication with an ASPSP.

VUB will manage communication with TPP using the IETF RFC 6749 - The OAuth 2.0 Authorization Framework („OAuth“, hereafter only). Therefore, in order to access VUB API, TPP must obtain 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.

3.1 Enrollment process

Each TPP, who wants to use VUB API, has to register using OAuth2.0 Open ID registration via API https://api.vub.sk/psd2/register. Registration swagger is available for download from Open Banking Portal (https://isbd.openbanking.intesasanpaolo.com/). Client (TPP) side SSL verification is required in order to avoid MITM attacks and also to validate provided eIDAS certificate and license status of TPP.

Enrollment of licensed TPP is fully automatic and therefore does not require additional processing or approval. Each registration request if check for following:

- Check of Client Certificate date validity
- Check of Client Certificate Revocation Status
- Check of Client Certificate for mandatory PSD2 eIDAS fields (organizationIdentifier and QCStatement)
- Check of Client Certificate Issuer
- Check of TPP license (status, country, scopes)

This API also offers the possibility for TPP to automatically change existing registration data, such as redirect URI update or eIDAS client certificate update.

After successful registration, TPP will use its License number as Client ID and eIDAS Certificate's serial number as Client Secret. Under Directive, TPP shall identify itself towards the ASPSP of the PSU for each communication session / every time a payment is initiated, and communicate with the ASPSP in a secure way.

Registration in Open Banking Portal is not a prerequisite for TPP in order to call VUB's API, but it is strongly recommended, as it provide additional documentation, swagger files, ticketing functionality for TPP's inquiries and reports of issues and ability to generate testing certificates for Sandbox environment.

For unlicensed TPPs or TPPs with pending licensing process, there is possibility to access VUB Sandbox environment for testing API. These TPPs need to register in Open Banking Portal and generate testing certificate in their management page. This certificate allows TPP to access Sandbox environment only.

All TPP registration requests to Open Banking Portal shall be resolved in 3 business days.

4 COMMON SECURE 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:

Table 1 – Supported Cipher suites

Cipher suite
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_256_CBC_SHA256

TLS_RSA_WITH_AES_256_CBC_SHA
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA

4.1 TPP and VUB authentication

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

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.

VUB uses Authorization code grant flow according to RFC 6749, Section 4.1 of OAuth for APIs, where PSU authentication is necessary, Client credentials grant flow according to RFC 6749, Section 4.4. of OAuth, where PSU participation is not needed, and Refresh token grant flow according to RFC 6749, Section 6. of OAuth, where PSU participation is not needed as well.

OAuth2.0 APIS provided by VUB are implemented in compliance with RFC 6749 specification (https://datatracker.ietf.org/doc/html/rfc6749).

4.2 Signatures

When TPP sends a digital signature, the signature must obey the following requirements.

Digest header

When a TPP includes a signature header, he also must include a “Digest” header. The “Digest" header must contain the hash of the message body, if the message does not contain a body, the “Digest” header must contain the hash of an empty bytelist.

The only hash algorithms that may be used to calculate the Digest within the context of this specification are SHA-256 and SHA-512.

Signature Header

The keyId field is a string that the server can use to look up the component they need to validate the signature. Serial Number of the TPP's certificate included in the "TPP-Signature-Certificate" header of this request.

It shall be formatted as follows: keyId="SN=XXX,CA=YYYYYY YYYYYYYYYY" where “XXX" is the serial number of the certificate in hexadecimal coding given in the TPP-Signature-Certificate Header and "YYYYYYYYYYYYYYYY" is the full Distinguished Name of the Certification Authority having produced this certificate.

The Algorithm parameter is used to specify the digital signature algorithm to use when generating the signature. The algorithm must identify the same algorithm for signature as described for the public key in the certificate element “TPP-Signature-Certificate” of the Request. It must identify SHA-256 or SHA-512 as Hash algorithm.

The Headers parameter is used to specify the list of HTTP headers included when generating the signature for the message.
Must include:

- “digest”
- “x-request-id”
- “date”

Must conditionally include (if and only if header is included as a header of the HTTP Request):

- “psu-id”
- “psu-corporate-id”
- “tpp-redirect-uri”

No other entries may be included

The signature parameter is a base 64 encoded digital signature. The client uses the `algorithm` and `headers` signature parameters to form a canonicalized `signing string`. This `signing string` is then signed with the key associated with `keyId` and the algorithm corresponding to `algorithm`. The `signature` parameter is then set to the base 64 encoding of the signature.

Example BASH

```
httpHost="https://api.vub.sk"
httpMethod="post"
reqPath="/v1/payments/sepa-credit-transfers"
reqId="78706f23-da34-4ffc-b4df-08d69bb28a2c"
ip="192.168.0.1"
redirect=https://api.vub.sk/callbacktest


# Serial number of the certificate in hexadecimal code
keyId="SN=051dc3bb36b1fe5da192b30000000000000000ff,CA=CN = Certification Authority TEST,C = SK"

payload='{
  "instructedAmount" : {
    "amount" : "0.2",
    "currency" : "EUR"
  },
  "currencyOfTransfer" : "EUR",
  "creditorName" : "creditor Revolut",
  "creditorAccount" : {
    "iban" : "${CIBAN}",
    "currency" : "EUR"
  },
  "requestedExecutionDate" : "2024-08-22"
}'
```

```
payloadDigest=`echo -n ${payload} | openssl dgst -binary -sha256 | openssl base64`
digest="SHA-256=${payloadDigest}"

reqDate=$(LC_TIME=en_US.UTF-8 date -u "+%a, %d %b %Y %H:%M:%S GMT")

# signingString must be declared exactly as shown below
signingString=`printf "digest: ${digest}\nx-request-id: ${reqId}\ndate: ${reqDate}\ntpp-redirect-uri: ${redirect}"`
signature=`printf %s "${signingString}" | openssl dgst -sha256 -sign "${signingCert}.pem" -passin "pass:${certPass}" | openssl base64 -A`

cert="${signingCert}.cer"

curl -v -i -k -X POST ${httpHost}${reqPath} \
-d "${payload}" \
-H "Accept:application/json" \
-H "Content-Type:application/json" \
-H "X-Request-id: ${reqId}" \
-H "PSU-IP-Address: ${ip}" \
-H "TPP-Signature-Certificate: ${cert}" \
-H "Date: ${reqDate}" \
-H "TPP-Redirect-URI: ${redirect}" \
-H "Digest: ${digest}" \
-H "Signature: keyId=\"${keyId}\",algorithm=\"rsa-sha256\",headers=\"digest x-request-id date tpp-redirect-uri\",signature=\"${signature}\"" \
--cert ${signingCert}.pem:${certPass}
```

5 PROVIDED APIS

Live API base URL: https://api.vub.sk/

Sandbox base URL: https://api.vub.sk/ + http request header "X-PSD2Sandbox" set to "yes"

API list of services implemented by VUB is provided in following table.

Table 2 - List of provided APIs

	Service	HTTP Method	Endpoint
AIS	Read Account List	GET	/v1/accounts
	Read Account Details	GET	/v1/accounts/{accountId}
	Read Balance	GET	/v1/accounts/{accountId}/balances
	Read Transaction List	GET	/v1/accounts/{accountId}/transactions
	Read Transaction Details	GET	/v1/accounts/{accountId}/transactions/{resourceId}
	Create an Account Information Consent	POST	/v1/consents
	Get Consent	GET	/v1/consents/{consentId}
	Delete an Account Information Consent	DELETE	/v1/consents/{consentId}
	Get Consent Status	GET	/v1/consents/{consentId}/status
PIS	Single Payment initiation	POST	/v1/payments/{paymentProduct}
	Bulk Payment initiation	POST	/v1/bulk-payments/{paymentProduct}
	Periodic Payment (Standing order) initiation	POST	/v1/periodic-payments/{paymentProduct}
	Get Single Payment Details	GET	/v1/payments/{paymentProduct}/{paymentId}
	Get Bulk Payment Details	GET	/v1/bulk-payments/{paymentProduct}/{paymentId}
	Get Periodic Payment Details	GET	/v1/periodic-payments/{paymentProduct}/{paymentId}
	Single Payment cancellation	DELETE	/v1/payments/{paymentProduct}/{paymentId}
	Bulk Payment cancellation	DELETE	/v1/bulk-payments/{paymentProduct}/{paymentId}
	Periodic payment cancellation	DELETE	/v1/periodic-payments/{paymentProduct}/{paymentId}
	Get Single Payment status	GET	/v1/payments/{paymentProduct}/{paymentId}/status
	Get Bulk Payment status	GET	/v1/bulk-payments/{paymentProduct}/{paymentId}/status

	Get Periodic Payment status	GET	/v1/periodic-payments/{paymentProduct}/{paymentId}/status
PIIS	Get Confirmation of Funds	POST	/v1/funds-confirmations
OAuth 2.0	Start Authorization	GET	/oauth2/authorize
	Issue Token	POST	/oauth2/token
Enrollment	TPP Register	POST	/psd2/register

6 ACCOUNT INFORMATION SERVICES

Consent request supports OAuth2.0 redirection flow and following representation:

- Dedicated Accounts
- Bank Offered Consent
- Global Consent - Account List of Available Accounts
- Global Consent - Account List and Balances of Available Accounts
- Global Consent - Access to all Accounts for all PSD2 defined AIS

As there is always single authorization object – Consent – authorized by single PSU, there is no need for explicit call of Consent Authorization request (POST /v1/consents/{consentId}/authorisations), therefore this step has been removed from Live API flow in compliance with BGS recommendation.

After Consent creation - TPP starts SCA & authorization process by redirecting PSU to OAuth2.0 /authorization call (https://api.vub.sk/oauth2/authorize) with scope = “AIS:{consentId}”

After successful SCA, OAuth2.0 authorization code is issued, which can be exchanged for access and refresh tokens, that are issued within AIS:{consentId} scope. In all AIS services calls, bearer token needs to be present inside “Authorization” http request header, set as “Authorization: Bearer {access_token}”.

6.1 Supported Bank Transaction Codes

In *Read Transaction List* and *Read Transaction Detail* APIs, values of *bankTransactionCode* matches BTC Codification ISO20022 standard, supporting following codes:

Table 3 – Supported Bank Transaction Codes

bankTransactionCode	Domain	Family	Sub-Family
PMNT-CNTR-BCDP	Payments	Counter Transactions	Branch Deposit
PMNT-CNTR-BCWD	Payments	Counter Transactions	Branch Withdrawal
PMNT-CCRD-CDPT	Payments	Customer Card Transactions	Cash Deposit
PMNT-CCRD-CWDL	Payments	Customer Card Transactions	Cash Withdrawal
PMNT-CCRD-POSD	Payments	Customer Card Transactions	Point-of-Sale (POS) Payment - Debit Card
PMNT-ICHQ-CCHQ	Payments	Issued Cheques	Cheque
PMNT-ICDT-XBCT	Payments	Issued Credit Transfers	Cross-Border Credit Transfer
PMNT-ICDT-INTR	Payments	Issued Credit Transfers	Interests (Generic)
PMNT-ICDT-ESCT	Payments	Issued Credit Transfers	SEPA Credit Transfer
PMNT-IRCT-XBCT	Payments	Issued Real-Time Credit Transfers	Cross-Border Credit Transfer
PMNT-IRCT-ESCT	Payments	Issued Real-Time Credit Transfers	SEPA Credit Transfer
PMNT-OTHR-OTHR	Payments	Other	Other
PMNT-RCDT-XBCT	Payments	Received Credit Transfers	Cross-Border Credit Transfer
PMNT-RCDT-INTR	Payments	Received Credit Transfers	Interests (Generic)
PMNT-RCDT-ESCT	Payments	Received Credit Transfers	SEPA Credit Transfer
PMNT-RDDT-PMDD	Payments	Received Direct Debits	Direct Debit
PMNT-RRCT-XBCT	Payments	Received Real-Time Credit Transfers	Cross-Border Credit Transfer
PMNT-RRCT-ESCT	Payments	Received Real-Time Credit Transfers	SEPA Credit Transfer

6.2 AIS Sandbox

For Sandbox, calling Consent Authorization request is necessary due to technical limitations of environment and proper Consent object authorization. Sandbox environment supports Dedicated Accounts representation only, with usage of following Accounts:

- SK1602000000004022453180
- SK4002000000007666767677
- SK3502000000006678666888

7 PAYMENT INITIATION SERVICES

After Payment initiation - TPP starts SCA & authorization process by redirecting PSU to OAuth2.0 /authorization call (https://api.vub.sk/oauth2/authorize) with scope = “PIS:{paymentId}”.

After successful SCA and payment authorization, the payment order is processed and OAuth2.0 authorization code is issued, which can be exchanged for access and refresh tokens, that are issued within PIS:{paymentId} scope. In following optional PIS services calls (Get Payment Status, Get Payment Detail), bearer token needs to be present inside “Authorization” http request header, set as “Authorization: Bearer {access_token}”.

7.1 PIS Sandbox

For Sandbox, initiating Payment order is possible only for integer values. Decimal values are not supported due to technical limitations of environment. Sandbox environment supports payment orders with usage of following Accounts:

- SK1602000000004022453180
- SK4002000000007666767677
- SK3502000000006678666888