Technical Manual for Implementing the “Account Direct Access” API

1 June 2020

Version 1 (pilot version)

TABLE OF CONTENTS

BACKGROUND

- In connection with the relationship with KB, the Partner shall comply with the rules described in the Code of Prudent Conduct.
- Before reading this Manual, we advise you to read the general description of the service. It also specifies the requirements and actions that must be met or taken prior to the technical connection to the Account Direct Access API.

REGISTRATION WITHIN THE API PORTAL

In order to actually connect to the service, it is necessary to register within the production API portal, register an application (third party to consume the issued API) and generate an API key. For more details, see the Production API Portal manual.

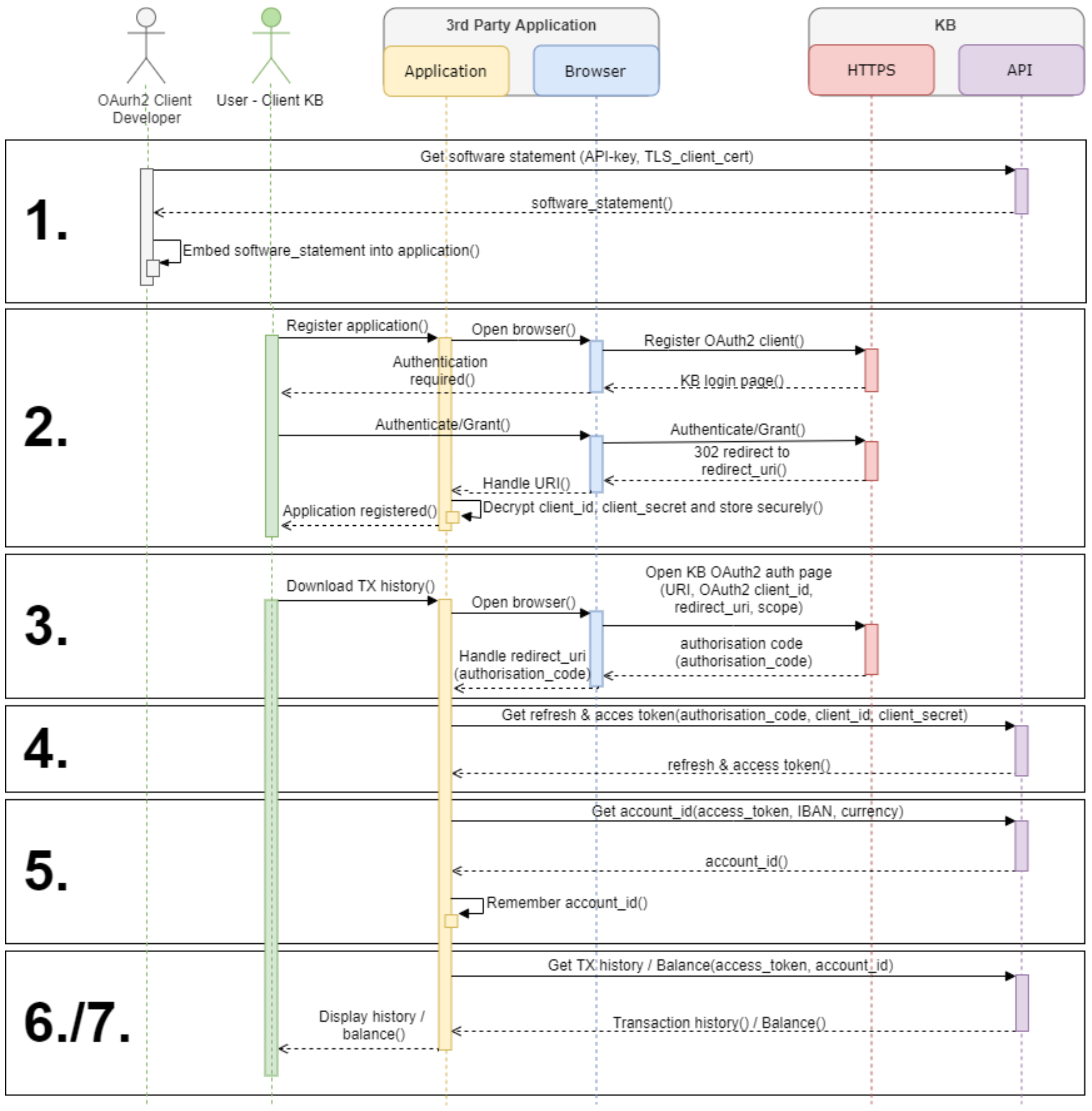
It is then necessary to subscribe the application to API:

- Account Direct Access
- OAUTH2
- Client registration

The swagger for the aforementioned service contains detailed description of all attributes and should be sufficient to understand the functionality. In general, the swagger documentation is written in English.

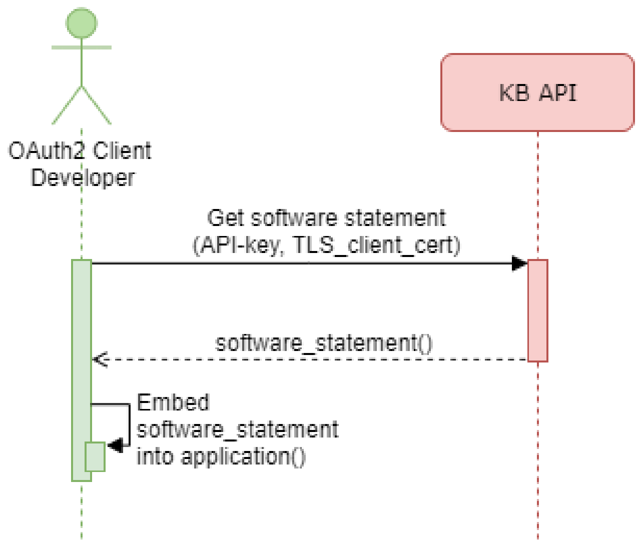
BANK'S RECOMMENDATIONS:

- Keep the API key secure;
- Access your API key remotely from your application;
- Generate new API key for each new version of your application.


OAurh2 Client Developer
User - Client KB
3rd Party Application
Application
Browser
KB
HTTPS
API
1.
Get software statement (API-key, TLS_client_cert)
software_statement()
Embed software_statement into application()
2.
Register application()
Open browser()
Register OAuth2 client()
Authentication required()
KB login page()
Authenticate/Grant()
Authenticate/Grant()
302 redirect to redirect_uri()
Handle URI()
Decrypt client_id, client_secret and store securely()
Application registered()
3.
Download TX history()
Open browser()
Open KB OAuth2 auth page (URI, OAuth2 client_id, redirect_uri, scope)
authorisation code (authorisation_code)
Handle redirect_uri (authorisation_code)
4.
Get refresh & acces token(authorisation_code, client_id, client_secret)
refresh & access token()
5.
Get account_id(access_token, IBAN, currency)
account_id()
Remember account_id()
6./7.
Get TX history / Balance(access_token, account_id)
Transaction history() / Balance()
Display history / balance()

1. ISSUANCE OF A SOFTWARE STATEMENT SIGNED BY THE BANK

Objective	**Get SW statement signed by Komerční banka**
Type	REST API
Called API	[POST] https://api.kb.cz/open/api/client-registration/v1/software-statements
Security	API key
Requirements	Valid qualified certificate issued by a certification authority
Request data	Certificate
Response data	Signed statement in the form of a JWT token


OAuth2 Client Developer
KB API
Get software statement (API-key, TLS_client_cert)
software_statement()
Embed software_statement into application()

Software statement is based on the OAUTH2 standard in line with RFC 7591 for dynamic registration of OAUTH2 agents. The statement is necessary for automatic registration of the application to ensure the registration credibility and security. Once API is successfully called, you will receive a statement – signed by the bank – in the form of a JWT token, by which a client's application will register in the bank. The statement is valid for 6 months. In case of new statement (due to expiration or lost) we highly recommend remove old one (it is possible to have more valid token for one application, but this is not desirable). JWT token must be securely implemented / saved in your application (or all installations thereof). For detailed API description, see the swagger documentation.

SW STATEMENT PARAMETERS

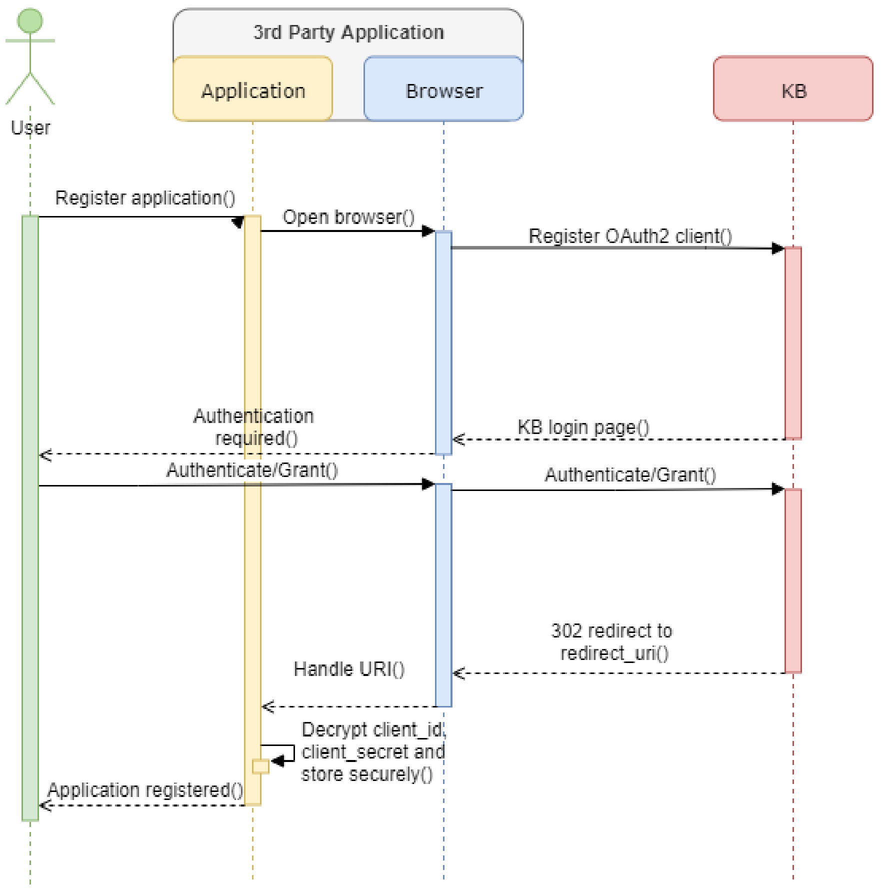
Parameter name	Mandatory	Data type	Validation	Description
softwareName	Y	String	5 - 50 characters	Software name - CZ
softwareNameEn	N	String	5 - 50 characters	Software name - EN
softwareId	Y	String	max 64 characters	Software identification code, any string (structure or value itself is not validated)
softwareVersion	Y	String	1 - 30 characters	Software version
softwareUri	N	String	-	Software URL
redirectUris	Y	List<String>	-	List of adresses, where the auth. code can be sent (specific adress is defined in redirect_uri, see Client's consent to data downloads via third-party application and obtaining authorisation code)
tokenEndpointAuthMethod	N	String	-	Fill „client_secret_post"
grantTypes	N	List<String>	-	Fill „authorization_code"
responseTypes	N	List<String>	-	Fill „code"
registrationBackUri	Y	String	-	URL for sending registration parameters (client_id, client_secret), see Registration of OAUTH2 agent – third party application
contacts	N	List<String>	max 50 characters	List of contact addresses (to contact people responsible for this client, typically email addresses).
logoUri	N	String	-	URL that references a logo for the client.
tosUri	N	String	-	URL that points to a human-readable terms of service document for the client.
policyUri	N	String	-	URL that points to a human-readable privacy policy document.

BANK'S RECOMMENDATIONS:

- Keep the software statement secure;
- Access the software statement remotely from your application.
- Don't use localhost as a redirectUri (in production version of service), due to the possibility of theft data by third party.

2. REGISTRATION OF OAUTH2 AGENT – THIRD PARTY APPLICATION

Objective	**Register with the bank a third-party application, as an OAUTH2 agent to allow communication between the application and the bank**
Type	Call (SSL) HTTP POST (SAML)
Called URI	https://api.kb.cz/client-registration/saml/register
Security	HTTPS, SSL, Software statement
Requirements	Signed statement in the form of JWT tokens, RSA-256 encryption support, embedded browser support is a plus
Request data	Statement in the form of a JWT token, redirect URI to receive the authorisation code. Encryption key, encryption algorithm (see the definition of parameters)
Response data	Encrypted registration data (clientId, clientSecret) as (encryptedData, salt)



The process of registering application with the bank is subject to client's consent to the operation. Web service calls are used for this purpose (following the consent and application registration). Your application should support an embedded browser.

Registration URI must be opened upon client's initiative, through which the client logs in with the bank, assigns a name to the application instance, and signs the operation. Registration parameters are sent to the banking URI via HTTP POST in a parameter (registrationRequest) as a JSON form in the BASE64URL format; see an example below.

Following the operation completion and successful registration, registration data (client_id, client_secret) – encrypted with an encryption key (encryptionKey) and encryption algorithm (encryptionAlg) – will be returned to the browser at the selected address (as specified in your software statement in step 1 - registrationBackUri). The data will be returned in parameters (encryptedData, salt).

DEFINITION OF PARAMETERS OF THE FORM FOR CALLED URI

Parameter	Mandatory	Data type	Validation	Description
applicationType	Y	String	„web" | „native"	Application type: • web • native
clientName	Y	String	4 - 255 characters	Name of registered OAUTH2 agent
clientNameEn	N	String	4 - 255 characters	English version of the name
redirectUris	Y	List<String>	Not empty	List of adresses, where the auth. code can be sent (specific adress is defined in redirect_uri, see Client's consent to data downloads via third-party application and obtaining authorisation code.
scope	Y	List<String>	Not empty	Scope of authorisation – "**adaa**" value is used.
encryptionKey	Y	String	-	Encryption key used to encrypt client_id, client_secret.
encryptionAlg	Y	String	„AES-256"	Encryption algorithm; **AES-256** is currently the only supported algorithm.
softwareStatement	Y	String	-	Software statement signed by the bank – in the form of a JWT token.

EXAMPLE OF A COMPLETED JSON FORM

```
{
   "clientName":"Nejlepsi Produkt",
   "clientNameEn":"Best Product",
   "applicationType":"web",
   "redirectUris":[
      "https://client.example.org/callback",
      "https://client.example.org/callback2"
   ],
   "scope":[
      "adaa"
   ],

    "encryptionKey": "djh5L0I/RShHK0tiUGVTaFZtWXEzdDZ3OXokQyZGKUo=",
    "encryptionAlg": "AES-256",
    "softwareStatement":"eyJhbGciOiJSUzI1NiJ9"
}
```

FINAL REQUEST FOR REGISTRATION OF THE OAUTH2 AGENT

The completed form is converted to the BASE64URL format and sent as a request in the **registrationRequest** parameter.

https://api.kb.cz/client-registration/saml/register?**registrationRequest**=eyAKICAgImNsaWVudE5hbWUiOiJOZWpsZXBzaSBrbGllbnQiLAogICAiY2xpZW50TmFtZUVuIjoiQmVzdCBjbGllbnQiLAogICAiYXBwbGljYXRpb25UeXBlIjoid2ViIiwKICAgInJlZGlyZWN0VXJpcyI6WyAKICAgICAgImh0dHBzOi8vY2xpZW50LmV4YW1wbGUub3JnL2NhbGxiYWNrIiwKICAgICAgImh0dHBzOi8vY2xpZW50LmV4YW1wbGUub3JnL2NhbGxiYWNrMiIKICAgXSwKICAgInNjb3BlIjpbIAogICAgICAiYWRhYSIKICAgXSwKICAgInNvZnR3YXJlU3RhdGVtZW50IjoiZXlKaGJHY2lPaUpTVXpJMU5pSjkiCn0

RESULT OF SUCCESSFUL REGISTRATION

In case the OAUTH2 agent is successfully registered, HTTP 302 redirect is returned to the registration redirectUri with **salt** and **encryptedData** parameters. The parameters are in the BASE64URL format and it is necessary to decode them from BASE64URL prior to decryption.

```
https://client.example.org/callback?salt=yMzQ1NiIsImVtYWlsOiBleGFtcGxlQGdvb2Rzb2Z0Lm&
encryptedData=VlciIsInNvZnR3YXJlTmFtZSI6Ik5lamxlcHNpIFByb2R1a3QiLCJzb2Z0d2FyZU5hbWVFb
iI6IkJlc3QgUHJvZHVjdCIsInNvZnR3YXJlSWQiOiJmNjRiZjJlNDQ3ZTU0NTIyO
```

DECRYPTED DATA STRUCTURE FOR SUCCESSFUL REGISTRATION

It results in JSON data registered in the bank for the given OAUTH2 agent. In order to complete further steps, it is necessary to securely save the **client_id** and **client_secret** values used for OAUTH2 agent's identification.

```
{
   "client_id":"NejlepsiProdukt-7427",
   "client_secret":"6dBtcLp27Q0UXrvfoXOSug",
   "api_key":"NOT_PROVIDED",
   "application_type":"web",
   "redirect_uris":[
      "https://client.example.org/callback",
      "https://client.example.org/callback2"
   ],
   "client_name":"Nejlepsi Produkt",
   "client_name#en-US":"Best Product",
   "logo_uri":"https://client.example.org/logo.png",
   "contact":"example@goodsoft.com",
   "bin":"45317054"
}
```

3. CLIENT'S CONSENT TO DATA DOWNLOADS VIA THIRD-PARTY APPLICATION AND OBTAINING AUTHORISATION CODE

Objective	**Grant consent to download transaction history from the bank in the application and obtain an authorisationCode**
Type	Call (SSL) HTTP POST (OAUTH2)
Called URI	https://login.kb.cz/autfe/ssologin
Security	HTTPS, SSL, OAUTH2
Requirements	Registered application with the bank as an OAUTH2 agent
Request data	client_id, redirect_uri, scope, response_type, state
Response data	code, state

The approval process of transaction history downloads by a third-party application is subject to client's consent to the operation. Bank web service calls are used for this purpose, through which the client gives its consent to access to his accounts. URI of the consent process (see the definition of parameters) must be opened upon client's initiative, through which the client logs in with the bank and signs the operation. Following the operation completion, an authorization code is returned to the specified redirect_uri for the purpose of obtaining refresh and access tokens that are used to download client data. State (optional parameter) serves as Cross-Site Request Forgery protection (see https://tools.ietf.org/html/rfc6749#section-4.1.1).

DEFINITION OF PARAMETERS OF THE CALLED URI

URI parameter	**Description**
redirect_uri	URI address to redirect authorisation_code specified upon registration of the OAUTH2 agent (validation, whether they are identical).
client_id	OAUTH2 agent's ID; assigned by the bank following the registration during the previous step.
scope	Scope of validity – "**adaa**" value is used.
response_type	Response value type; "**code**" value is used.

FINAL REQUEST FOR CONSENT

```
https://login.kb.cz/autfe/ssologin?scope=adaa&client_id= NejlepsiProdukt-7427&redirect_uri= https://client.example.org/callback&response_type=code
```

RESULT OF SUCCESSFUL CONSENT PROCESS

In case the consent for TH downloads by an OAUTH2 agent is successfully granted, HTTP 302 redirect is returned to the registration redirectUri with **code** parameter that contains an authorisation code.

```
https://client.example.org/callback?code=ZnR3YXJlTmFtZSI6Ik5lamxlcHNp
```

4. RECEIVE TOKEN

Objective	Receive authorisation code to exchange for an access and refresh token and allow transaction history downloads
Type	REST API
Called API	[POST] https://api.kb.cz/open/api/oauth2/v1/access_token
Requirements	Consent given by a client, authorisation_code
Request data	client_id, client_secret, authorisation_code
Response data	refresh_token, access_token

After the receipt of an authorisation code issued on the basis of client's consent to transaction history downloads, it must be exchanged for refresh and access tokens. Endpoint is used for this purpose. Endpoint is also used to obtain an access token with a refresh token. Using an access taken (access_token), it is possible to call upon all ADAA endpoints. For detailed API description, see the swagger documentation.

DEFINITION OF TOKENS:

Token	Description	Expiration
access_token	It is used as an authorisation parameter to call the ADAA. Note: The token may be used repeatedly prior to its expiration.	3 minutes
refresh_token	It is used to issue a new access token; it is used as a parameter to call the OAUTH2 API service. Note: The token may be used repeatedly prior to its expiration.	12 months

5. REPLACE IBAN WITH AN ACCOUNT IDENTIFICATION NUMBER

Objective	Obtain accountID for the given IBAN
Type	REST API
Called API	[POST] https://api.kb.cz/open/api/adaa/v1/account-ids
Requirements	Known an IBAN number and account currency
Request data	IBAN, currency, access_token
Response data	account_id

For the purpose of enhanced security, it is not possible to send IBAN in the header in its open form; it is necessary to apply for the so-called identification number (i.e. encrypted IBAN). Transaction history or account information downloads are called using the identification number - accountId. For detailed API description, see the ADAA swagger documentation.

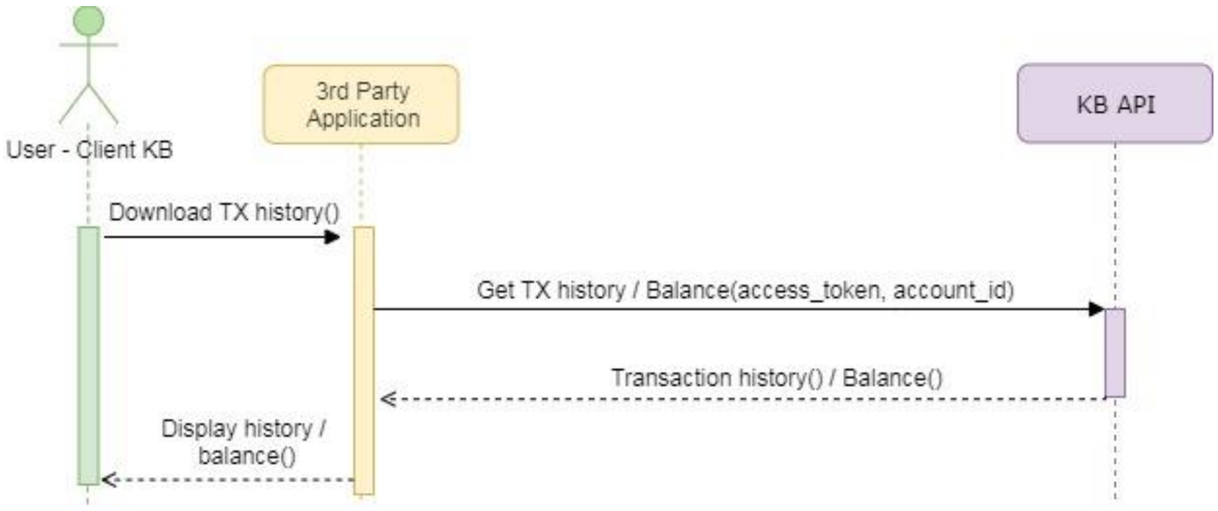
BANK'S RECOMMENDATIONS:

- Keep the account_id in the application for later use. It is not necessary to call the endpoint again.

6. TRANSACTION HISTORY DOWNLOADS

Objective	**Download transaction history**
Type	REST API
Called API	[GET] https://api.kb.cz/open/api/adaa/v1/transactions
Requirements	Valid access_token, account_id
Request data	account_id, access_token
Response data	Transaction history

The endpoint is used to download transaction history for the account in question. For detailed API description, see the ADAA swagger documentation.


User - Client KB
3rd Party Application
KB API
Download TX history()
Get TX history / Balance(access_token, account_id)
Transaction history() / Balance()
Display history / balance()

7. DOWNLOAD ACCOUNT BALANCE

Objective	**Download account balance**
Type	REST API
Called API	[GET] https://api.kb.cz/open/api/adaa/v1/balances
Requirements	Valid access_token, account_id
Request data	account_id, access_token
Response data	Account balance

The endpoint is used to download account balance for the account in question. For detailed API description, see the swagger documentation.

8. DOWNLOAD ACCOUNT STATEMENT

Cíl	**Obtain statement metadata (include statementId)**
Typ	REST API
Volané API	[GET] https://api.kb.cz/open/api/adaa/v1/statements
Požadavky	Valid access_token, account_id
Request data	account_id, access_token, dateFrom
Response data	Statement metadata (include statemen_iId – id of specific PDF file)

The endpoint is used to download information about account statements (metadata). Metadata contains important parameter – statement_id – id of specific PDF file (for downloading specific PDF file; see endpoint below). For detailed API description, see the ADAA swagger documentation.

Cíl	**Download account statement in PDF format**
Typ	REST API
Volané API	[GET] https://api.kb.cz/open/api/adaa/v1/statements
Požadavky	Valid access_token, account_id, statement_id
Request data	account_id, access_token, statement_id
Response data	Statement in PDF format

The endpoint is used to download specific statement / file in PDF format. For detailed API description, see the ADAA swagger documentation.