PSD2 – Integrační příručka pro poskytovatele třetích stran (TPP)

1 Úvod

Tento dokument popisuje základní způsob integrace třetí strany ke službám PSD2 poskytovaným J&T Bankou. Je určen pro poskytovatele třetích stran (TPP – Third Party Providers), kteří implementují přístup k bankovním službám dle standardu vycházejícího z Czech Open Banking Standard (COBS) verze 7:

Služba	Scope	Popis
AISP	AISP	Přístup k informacím o účtech (Account Information Service)
PISP	PISP	Inicializace plateb (Payment Initiation Service)
CISP	CISP	Ověření dostupnosti finančních prostředků (Card Issuer Service)

Implementace API PSD2 je v souladu s následujícími standardy:

- **OAuth 2.0** (RFC 6749)
- **PKCE** : **Proof Key for Code Exchange by OAuth Public Clients** viz RFC 7636
- **PAR: Pushed Authorization Requests** viz RFC 9126 (pouze PISP)
- **RAR: Rich Authorization Requests** viz RFC 9396 (pouze PISP)
- **OAuth 2.0 Dynamic Client Registration** — RFC 7591
- **Czech Open Banking Standardu** verze 7

2 Předpoklady (Prerequisites)

Pro úspěšnou integraci je třeba si zajistit

- kvalifikovaný eIDAS certifikát (QWAC/QSealC), předepsaný regulací (RTS on SCA & CSC – EU 2018/389) certifikát pro mTLS komunikaci se systémy J&T poskytující PSD2 služby (registrationNumber, registrator a roles)
- Schopnost provádět OAuth2 klientské přihlášení pomocí grantů *client credentials* a *authorization code*

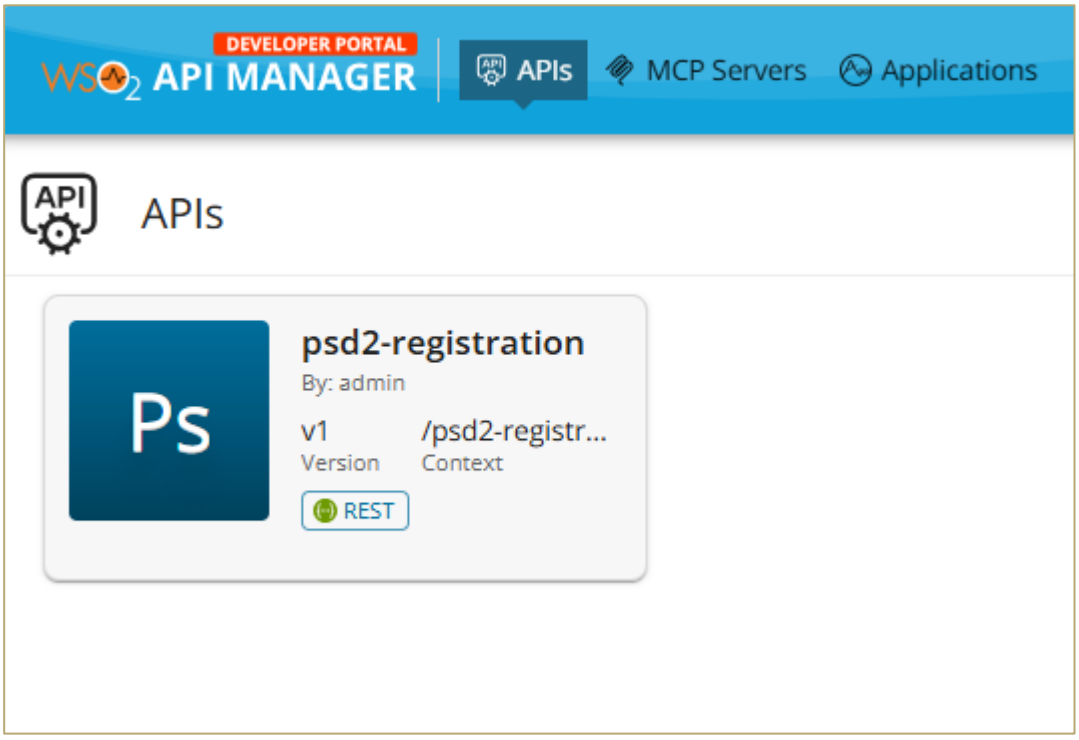
3 Seznam zástupných hodnot parametrů

Placeholder	Význam	Hodnota
apigw-hostname	PSD2 API gateway	https://apigw.jtbank.cz
portal-hostname	Developer portal	https://apiportal.jtbank.cz
authserver-url	Autorizační server	https://bezpecnost.jtbank.cz/sezamas/v1
clientId	Unikátní identifikátor TPP aplikace (UUID)	n/a
clientSecret	Tajný klíč — neměnný po celou dobu platnosti klienta	n/a
redirectUri	Registrovaná callback URL — nelze měnit po registraci	n/a
scope	Rozsah oprávnění přidělených access tokenu	n/a
pathToCertificate	Cesta ke kvalifikovanému certifikátu	n/a
authorization	Aurtizační řetzec např. Bearer ……	n/a
tppName	Jméno poskytovatele třetí strany	n/a
accessCode	Autorizační kód pro získání access tokenu	n/a
refreshToken	Token pro obnovení access tokenu	n/a
state	Ochranný parametr proti CSRF útoku	n/a
bearerToken	Přístupový token typu Bearer pro API volání	n/a
pkceCodeVerifier	Ochranný parametr proti CSRF útoku	n/a
authorizationDetails	Rozšířená autorizace specifikující detail oprávnění (platební instrukce)	n/a

4 Registrace do developer portálu (Registering into developer portal)

Prvním krokem pro správnou integraci je vytvoření účtu v Developer portalu, kde je následně možné se zaregistrovat k odběru služeb PSD2.

Na stránkách **https://*{{portal-hostname}}*/devportal/apis** naleznete příslušné API PSD2Registration, které vám umožní registraci vaší společnosti jako PSD2 poskytovatele třetí strany (Third Party Provider).



Registrace se provádí pomocí REST volání **POST https://*{{apigw-hostname}}*/psd2-registration/v1/tpps** která zajistí přístup do developer portálu J&T Banky a následně zaregistrovat oAuth2 klienty, kterými jste oprávněni volat PSD2 API.

> ⚠ Důležité: všechna volání musíte provolat s kvalifikovaným eIDAS certifikáty (QWAC/QSealC), které jsou předepsané regulací (RTS on SCA & CSC – EU 2018/389)

Příklad provolání:

```
curl -X POST "https://{{apigw-hostname}}/psd2-registration/v1/tpps" \
 --cert {{pathToCertificate}} --key {{pathToKey}} \
 -H "Content-Type: application/json" \
 -d '{
   "tppId": "{{tppId}}",
   "tppName": "{{tppName}}",
   "ico": "{{ico}}",
   "address": "{{address}}",
   "password": "{{password}}",
   "firstName": "{{firstName}}",
   "lastName": "{{lastName}}",
   "phoneNumber": "{{phoneNumber}}",
   "email": "{{email}}"
 }'
```

kde :

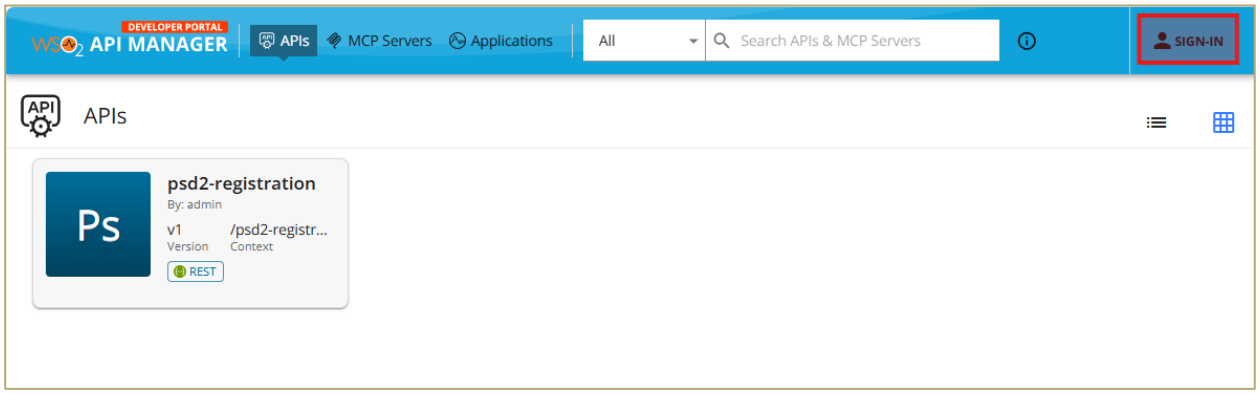
- **tppId** je uživatelské jméno pro přístup do developer portálu. Nesmí obsahovat mezery, podtržítko ani diakritiku.
- **password** heslo pro identity server splňující požadavky bezpečnostní politiky (policy pattern): ^((?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$%&*])).{0,100}$
- **email** je povinným údajem pro účely potvrzení registrace a zasílání notifikací. Po úspěšné registraci bude na uvedenou adresu odeslán e-mail s potvrzovacím odkazem.
- **address** a **phoneNumber** jsou rovněž povinné údaje. Administrátoři J&T Banky vás mohou z bezpečnostních důvodů kontaktovat. Uvedení nepravdivých údajů může být nahlášeno příslušným orgánům.

4.1 2.2 Errors

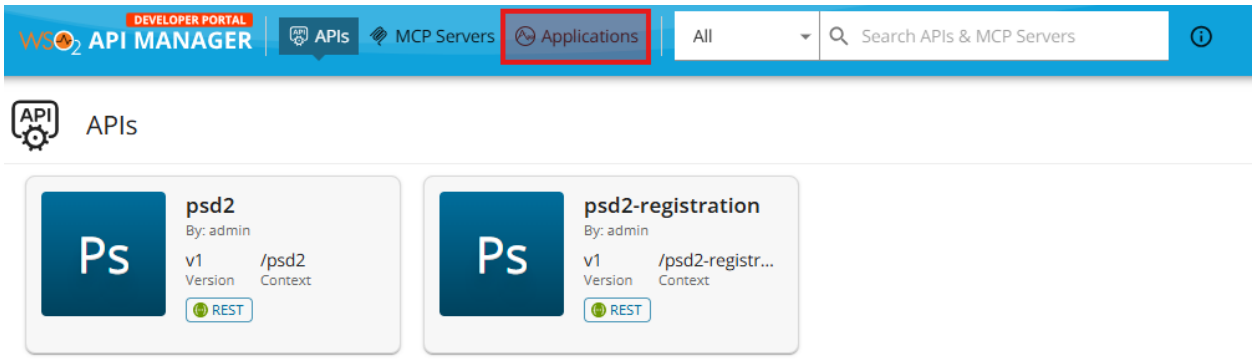
http error	Error message	Popis
403	User xxxx already exists in the system. Please use a different username.	Zadané tppId je již registrováno. Použijte prosím jiné.
400	The certificate is missing valid metadata	Certifikát nesplňuje požadované náležitosti (viz předpoklady).
400	Some of following parameters is missing in JSON request: tppId or password or tppName or email or phoneNumber	Tělo požadavku není platný nebo kompletní JSON.
400	The name should contain character '_'	
403	Unable to process registration. The TPP has been already registered under a different tppId: xxx	Jiný uživatel s poskytnutým certifikátem je již registrován.
200	User has been already registered. No changes has been done.	
403	The TPP has been already registered	
500	Unable to register user – internal error - contact support please	Interní chyba.
500	Unable to set roles	Interní chyba.

5 Vytvoření oAuth2 klienta (registrace WSO2 aplikace)

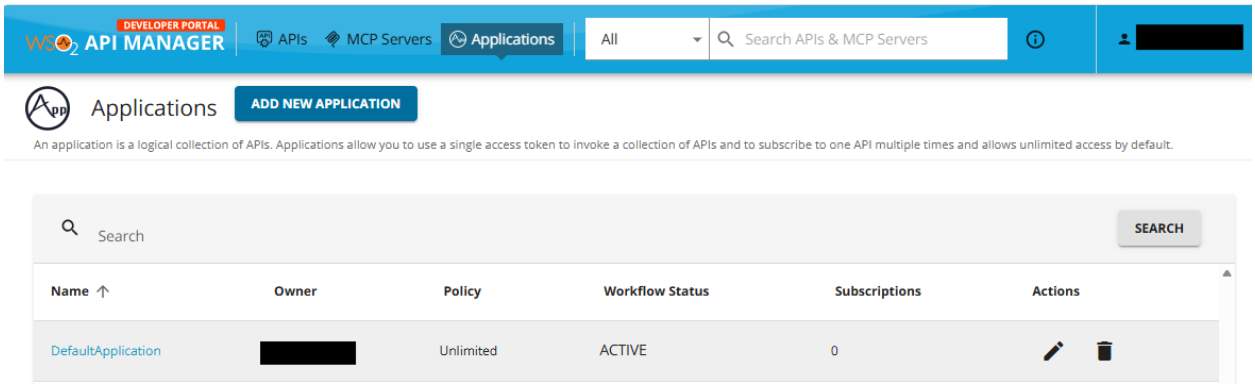
Po úspěšné registraci je nutné vytvoření oAuth2 klienta a se přihlásit k odběru PSD2 služeb. Za tímto účelem je třeba se přihlásit k Developer portálu



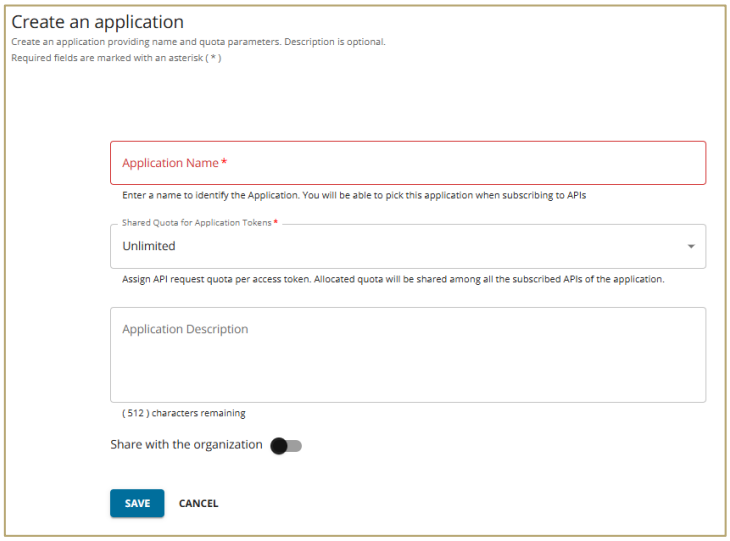
Po úspěšném přihlášení přejděte do sekce Applications a vytvořte novou aplikaci.



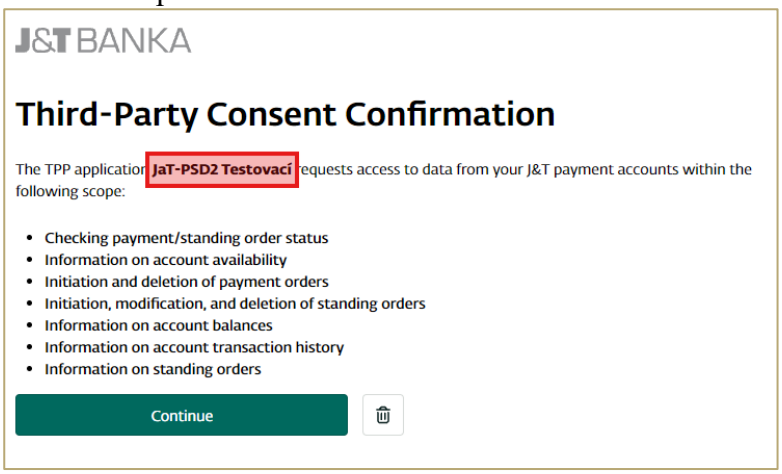
Můžete použít výchozí (default) aplikaci, ale mějte na paměti, že její název a **popis (application description)** se bude zobrazovat koncovým uživatelům na obrazovce při udělování souhlasu (Third-Party Consent Confirmation).



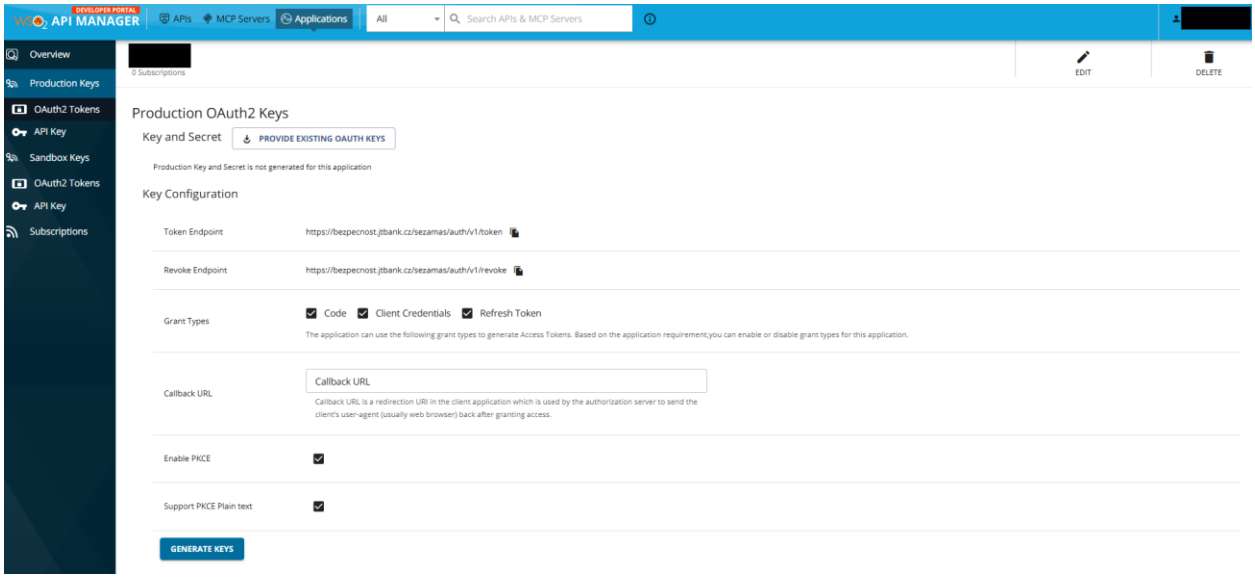
Při založení nové wso2 aplikace je třeba vyplnit



- Název aplikace
  - Nesmí obsahovat znak podtržítka (_).
- Per Token Quota
  - Toto pole slouží k nastavení limitu API požadavků na jeden access token.
  - Nastavený limit je sdílen mezi všemi API, ke kterým je aplikace přihlášena.
  - Můžete ponechat hodnotu Unlimited, pokud nechcete limit omezovat.
- Description
  - Popis aplikace je povinný a zobrazuje se při přihlášení uživatele v rámci autorizace operace



Následně je možné provést konfiguraci oAuth2 klienta v levém menu „Production Keys >> oAuth2 tokens“

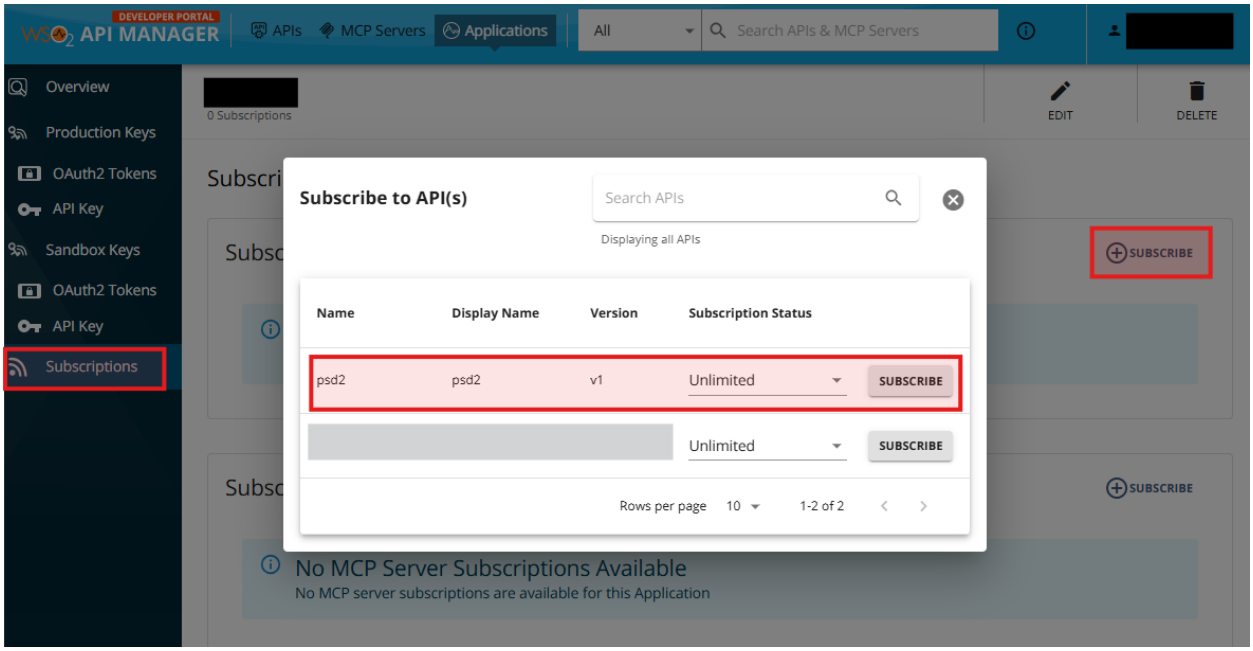


Zadejte **Callback URL**, která bude využita pro OAuth2 toky **Authorization Code**, přičemž tato URL musí odpovídat endpointu aplikace, na který bude uživatel přesměrován po úspěšné autentizaci nebo autorizaci; **client credentials** lze využít pouze ve specifických scénářích, například pro iniciaci plateb.

Poté klikněte na tlačítko **Generate keys**, čímž získáte hodnoty **clientId (consumerKey)** a **clientSecret (consumerSecret)**

6 API Subscription

Aplikace musí být přihlášena (subscribed) k odběru daného API, aby bylo možné toto API volat pomocí oAuth2 tokenů získaných na základě clientId a clientSecret. To se provede v rámci wso2 aplikace v levém menu „Subcription“ >> ⊕ SUBSCRIBE v sekci : Subscribed APIs



7 Klientský a aplikační token (autentizace a autorizace)

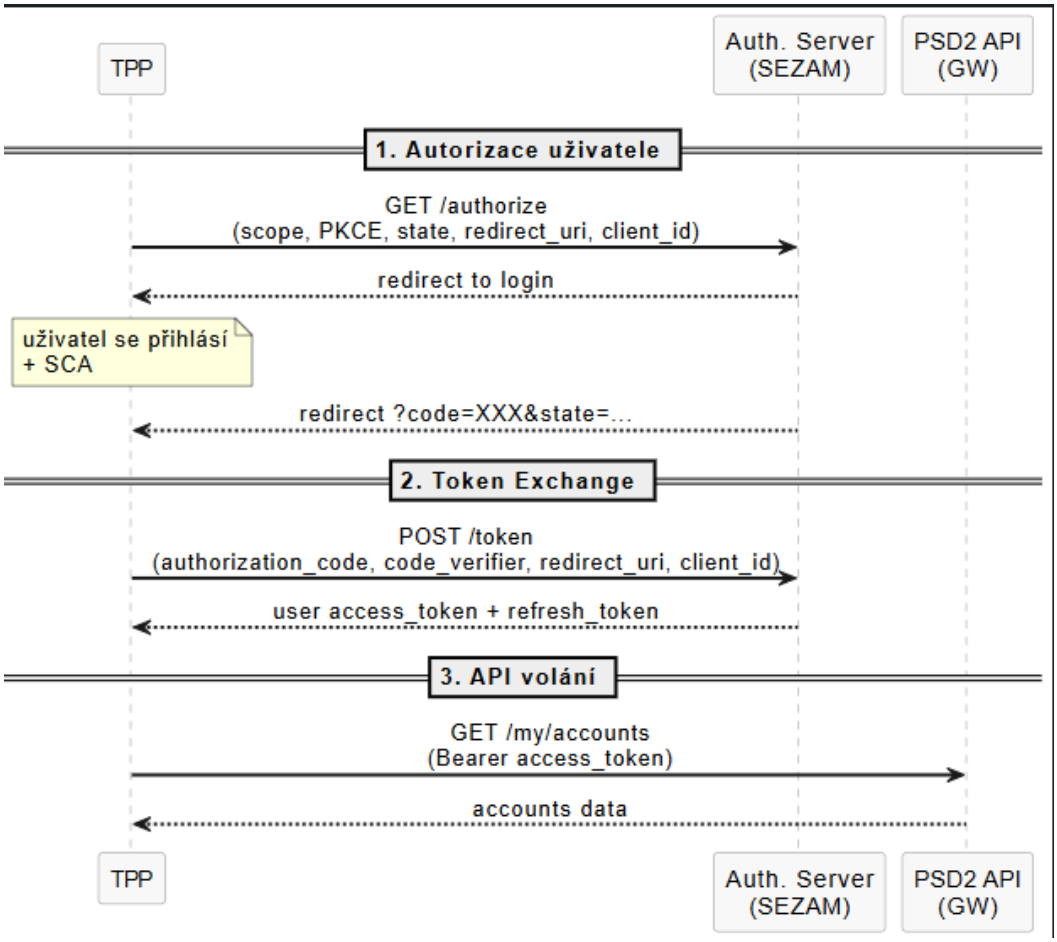
V rámci jednotlivých služeb API PSD2 je uveden způsob autorizace, který lze pro danou službu využít.

- **Klientský (uživatelský) token**: OAuth 2.0 access token vydaný v rámci autorizačního toku (*Authorization Code Flow*), který je získán na základě silného ověření klienta (SCA) a je vázán na konkrétního koncového uživatele

- **Aplikační token TPP**: OAuth 2.0 access token vydaný v rámci toku *Client Credentials*, kdy dochází k ověření výhradně na základě OAuth 2.0 přihlašovacích údajů (client_id / client_secret) přidělených třetí straně viz výše. Token není vázán na konkrétního uživatele, ale na aplikaci jako takovou.

7.1 Autentizace PSD2 služeb

Tímto způsobem se autentizují/autorizují služby mimo zadání plateb a trvalých příkazů.
Získání klientského/uživatelského tokenu (autentizace/autorizace klienta)




```
curl -X GET "https://{{authserver-url}}/authorize\
?response_type=code\
&scope={{scope}}\
&redirect_uri={{redirectUri}}\
&client_id={{clientId}}\
&state={{state}}\
&code_challenge={{pkceCodeChallenge}}\
&code_challenge_method=S256"
```

```
curl -X POST 'https://{{authorizationServerUrl}}/token' \
-H 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'redirect_uri={{redirectUri}}' \
--data-urlencode 'grant_type=authorization_code'\
--data-urlencode 'code={{accessCode}}' \
```

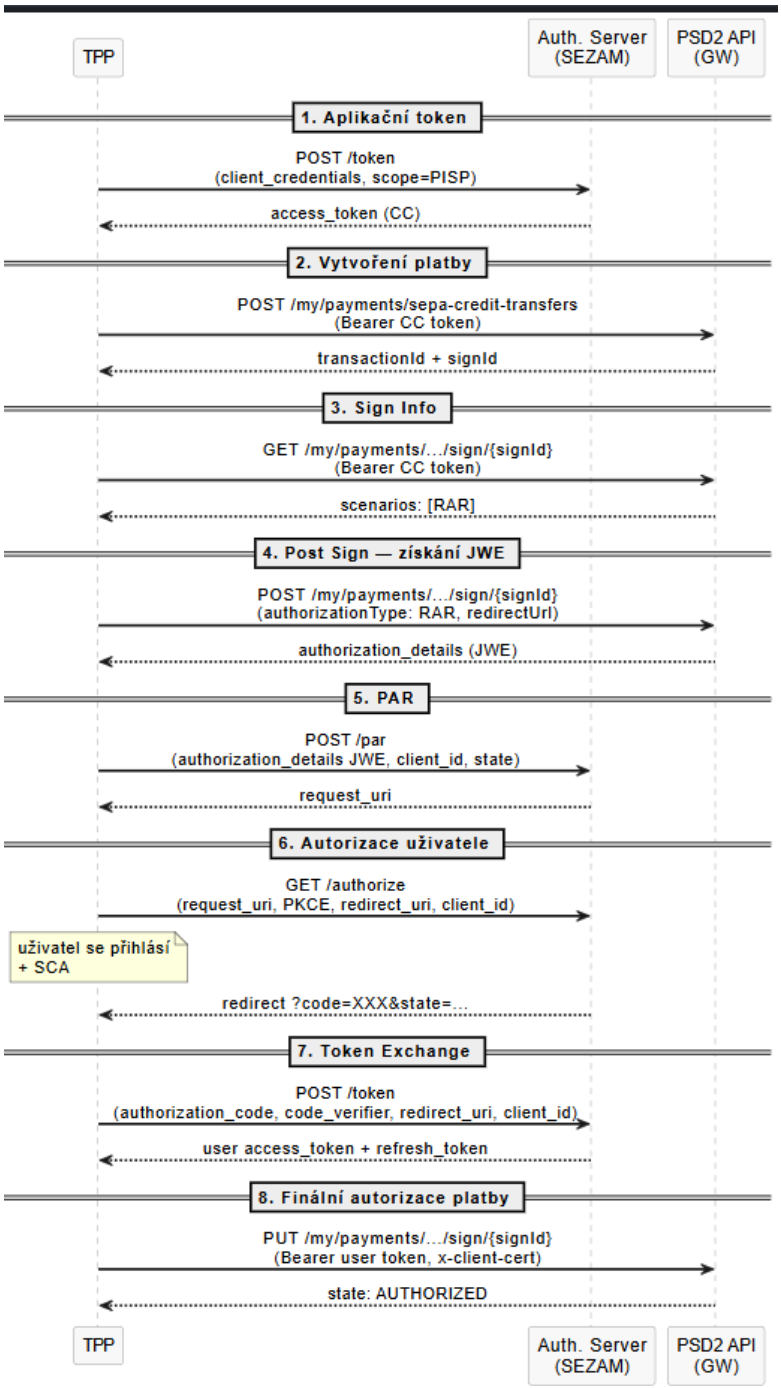
```
--data-urlencode 'scope={{scope}}' \
--data-urlencode 'client_id={{clientId}}' \
--data-urlencode 'state={{state}}' \
--data-urlencode 'code_verifier={{pkceCodeVerifier}}' \
```

Získání aplikačního tokenu (autentizace/autorizace TPP)

```
curl -X POST "https://{{authserver-url}}/token" \
  -H "Authorization: Basic {{authorization}}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=client_credentials" \
  --data-urlencode "scope={{scope}}"
```

7.2 Autorizace platebních instrukcí a trvalých příkazů

V případě autorizace trvalých plateb a platebních instrukcí je třeba předat autorizačnímu serveru autorizační objekt. Za tímto účelem je v rámci autentizace využit Rich Authorization Request a Push Athorization Request



```
curl -X POST "https://{{authserver-url}}/par" \
  --cert client.crt --key client.key \
  -H "accept: application/json" \
  -H "Traceparent: true" \
  -H "Date: Wed, 21 Oct 2015 07:28:00 GMT" \
  -H "User-Involved: 1" \
  -H "TPP-Name: {{tppName}}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -H "Authorization: Bearer {{bearerToken}}" \
  -H "x-client-cert: {{pathToCertificate}}" \
  --data-urlencode "client_id={{clientId}}" \
  --data-urlencode "scope={{scope}}" \
  --data-urlencode "authorization_details={{authorizationDetails}}" \
  --data-urlencode "response_type=code" \
  --data-urlencode "state={{state}}" \
  --data-urlencode "redirect_uri={{redirectUri}}"
```

```
curl -X POST "https://{{authorizationServerUrl}}/authorize\
?client_id={{clientId}}\
&request_uri={{requestUri}}\
&code_challenge={{pkceCodeChallange}}\
&code_challenge_method={{pkceMethod}}\
&ui_locales=cs"
```

```
curl -X POST "https://{{authorizationServerUrl}}/token" \
  --cert client.crt --key client.key \
  -H "Authorization: Basic {{authorization}}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "redirect_uri={{redirectUri}}" \
  --data-urlencode "grant_type=authorization_code" \
  --data-urlencode "code={{accessCode}}" \
  --data-urlencode "scope={{scope}}" \
  --data-urlencode "client_id={{clientId}}" \
  --data-urlencode "state={{state}}" \
  --data-urlencode "code_verifier={{pkceCodeVerifier}}"
```

7.3 Časové limity

Parametr	Hodnota	Popis
access_token platnost	3600s (60 min)	Konfigurovatelné na straně serveru
refresh_token platnost	180 dní	Platí po celou dobu souhlasu uživatele
request_uri (PAR) platnost	300s (5 min)	Po expiraci nelze použít v authorize
authorization_code platnost	300s (5 min)	Použít ihned po získání

8 Refresh token

Access token je platný pouze 3600 sekund, proto Authorization Code grant poskytuje kromě access tokenu také refresh token, který má platnost 180 dní a lze jej použít k získání nového access tokenu prostřednictvím OAuth refresh grant toku.

```
curl -X POST "https://{{authserver-url}}/token" \
  -H "Authorization: Basic {{authorization}}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=refresh_token" \
  --data-urlencode "refresh_token={{refreshToken}}"
```

⚠ **Refresh Token Rotation**: server vrací při každém refresh grantu nový refresh token a původní token zneplatňuje; implementace musí nový refresh token uložit a používat pro další obnovu access tokenu

9 Integrační scénáře

9.1 AISP Scénář — Získání seznamu účtů

AISP nevyžaduje PAR — autorizace probíhá přímo přes /authorize (RFC 6749, Section 4.1).

Krok 1: Client Credentials Token (volitelný)

Poznámka: CC token není potřeba pro user-authorization flow (kroky 2–4). Získejte ho tehdy, kdy TPP potřebuje volat API na pozadí bez přítomnosti uživatele (background sync, batch zpracování). Pokud chcete pouze přihlásit uživatele a načíst jeho účty, přejděte přímo na krok 2.

```
curl -X POST "https://{{authserver-url}}/token" \
  -H "Authorization: {{authorization}}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=client_credentials" \
  --data-urlencode "scope={{scope}}"
```

Response (200):

```
{
  "access_token": "{{accessToken}}",
  "refresh_token": "{{refreshToken}}",
  "scope": "{{scope}}",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

Krok 2: Authorize — přesměrování uživatele

Přesměrovat prohlížeč uživatele na tuto URL:

```
curl -X GET "https://{{authserver-url}}/authorize\
?scope={{scope}}\
&response_type=code\
&redirect_uri={{redirectUri}}\
&client_id={{clientId}}\
&code_challenge={{pkceCodeChallenge}}\
&code_challenge_method=S256\
&state={{state}}"
```

Uživatel:

1. přihlásí se (username + password)
2. provede SCA (Strong Customer Authentication — SMS OTP nebo mobilní aplikace)
3. Potvrdí souhlas (consent)

Po úspěšném přihlášení je uživatel přesměrován na:

```
https://{{redirectUri}}/callback?code={{callbackCode}}
&state={{state}}
```

Krok 3: Token Exchange
Výměna authorization code za uživatelský access token.

```
curl -X POST "https://{{authserver-url}}/token" \
  -H "Authorization: {{authorization}}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "redirect_uri={{redirectUri}}" \
  --data-urlencode "grant_type=authorization_code" \
  --data-urlencode "code={{accessCode}}" \
  --data-urlencode "scope={{scope}}" \
  --data-urlencode "client_id={{clientId}}" \
  --data-urlencode "state={{state}}" \
  --data-urlencode "code_verifier={{pkceCodeVerifier}}"
```

Response (200):

```
{
  “access_token": “{{accessToken}}”,
  "token_type": "bearer",
  "expires_in": "3600",
  "refresh_token": "{{refreshToken}}",
  "scope": "{{scope}}"
}
```

⚠ Pozor: **expires_in** může být string i number — implementace musí zvládnout oba formáty

Krok 4: GET Accounts (seznam účtů)

```
curl -X GET "https://{{apigw-hostname}}/psd2/v1/my/accounts?page=0&size=10" \
  --cert {{pathToCertificate}} --key {{pathToKey}} \
  -H "Authorization: {{authorization}}" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -H "Traceparent: {{traceParent}}" \
  -H "User-Involved: true" \
  -H "TPP-Name: {{tppName}}" \
  -H "Action-ID: {{actionId}}" \
  -H "TPP-Identification: {{tppId}}" \
  -H "X-Request-ID: {{xRequestId}}" \
  -H "Date: {{date}}"
```

Response (200):


```
{
  "pageNumber": 0,
  "pageCount": 1,
  "pageSize": 10,
  "totalCount": 3,
  "accounts": [
    {
      "accountId": "CZ7158000000000000000123",
      "identification": {
        "iban": "CZ7158000000000000000123",
        "other": "123"
      },
      "currency": "CZK",
      "servicer": {
        "bankCode": "5800",
        "countryCode": "CZ",
        "bic": "JTBPCZPP"
      },
      "nameI18N": "Běžný účet",
      "productI18N": "Běžný účet"
    }
  ]
}
```

Krok 5: Token Refresh (po expiraci)

Po expiraci access tokenu použijte refresh token pro získání nového.

```
curl -X POST "https://{{authserver-url}}/token" \
  -H "Authorization: {{authorization}}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=refresh_token" \
  --data-urlencode "refresh_token={{refreshToken}}" \
  --data-urlencode "client_id={{clientId}}"
```

Response (200):

```
{
  "access_token": "{{accessToken}}",
  "token_type": "bearer",
  "expires_in": "3600",
  "refresh_token": "{{refreshToken}}",
  "scope": "{{scope}}"
}
```

9.2 PISP Scénáře — Iniciace platby

PISP flow vyžaduje PAR (RFC 9126) pro předání platebního autorizačního objektu (JWE). Autorizace platby používá RAR (RFC 9396). Debetní účty jsou podporovány pouze ve formátu IBAN.

Banka podporuje všechny typy plateb (Tuzemské – pouze v rámci ČR, SEPA, Zahraniční), typ platby (v souladu s definicí dle ČOBS) se vrací v response služby jako ServiceLevel / Code a určuje ho platební systém JnT dle charakteristiky zadaných informací při iniciaci platby.

9.3 PISP Scénář 1 — Platba bez debetního účtu se získáním uživatelského tokenu

Kdy použít: TPP nezná IBAN debetního účtu předem. Uživatel je identifikován přes uživatelský token získaný před iniciací platby Po autentizaci získá uživatel možnost vybrat debetní účet ze seznamu dle svého disponentského modelu.

Klíčový rozdíl oproti scénáři 2: Pole debtorAccount není uvedeno v těle platby → server vyžaduje uživatelský (ne aplikační) token na /payments.

Krok 1: Client Credentials Token (PISP)

```
curl -X POST "https://{{authserver-url}}/token" \
  -H "Authorization: Basic {{authorization}}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=client_credentials" \
  --data-urlencode "scope={{scope}}"
```

Response (200):

```
{ "access_token": "ACCESS_TOKEN_PISP" }
```

Krok 2: Authorize — přesměrování uživatele (PISP scope)

```
curl -X GET "https://{{authserver-url}}/authorize\
?scope={{scope}}\
&response_type=code\
&redirect_uri={{redirectUri}}\
&client_id={{clientId}}\
&code_challenge={{pkceCodeChallenge}}\
&code_challenge_method=S256\
&state={{state}}"
```

Callback vrátí: https://moje-tpp.example.com/callback?code=UserCodePISP&state=pisp-state-001

Krok 3: Token Exchange → uživatelský PISP token

```
curl -X POST "https://{{authserver-url}}/token" \
  -H "Authorization: Basic {{authorization}}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "redirect_uri={{redirectUri}}" \
  --data-urlencode "grant_type=authorization_code" \
  --data-urlencode "code={{accessCode}}" \
  --data-urlencode "scope={{scope}}" \
  --data-urlencode "client_id={{clientId}}" \
  --data-urlencode "state={{state}}" \
  --data-urlencode "code_verifier={{pkceCodeVerifier}}"
```

Response (200):

```
{
  "access_token": "{{accessToken}}",
  "refresh_token": "{{refreshToken}}",
  "scope": "{{scope}}",
  "token_type": "bearer",
  "expires_in": "3600"
}
```

Krok 4: Create Payment — bez debtorAccount (s uživatelským tokenem)

```
curl -X POST "https://{{apigw-hostname}}/psd2/v1/payments" \
  --cert {{pathToCertificate}} --key {{pathToKey}} \
  -H "Authorization: Bearer {{bearerToken}}" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -H "Traceparent: {{traceParent}}" \
  -H "User-Involved: true" \
  -H "TPP-Name: {{tppName}}" \
  -H "Action-ID: {{actionId}}" \
  -H "TPP-Identification: {{tppId}}" \
  -H "X-Request-ID: {{xRequestId}}" \
  -H "Date: {{date}}" \
  -d '{
    "paymentIdentification": {
```

```
    "instructionIdentification": "PLATBA-S1-001"
   },
   "amount": {
    "instructedAmount": {
     "value": 100.00,
     "currency": "CZK"
    }
   },
   "creditorAccount": {
    "identification": {
     "iban": "CZ5508000000001234567899"
    }
   }
 }'
```

debtorAccount není uvedeno — server identifikuje plátce z uživatelského tokenu.

Response (200):

```
{
  "paymentIdentification": {
    "transactionIdentification": "{{paymentId}}"
  },
  "signInfo": {
    "state": "OPEN",
    "signId": "{{signId}}"
  },
  "paymentTypeInformation": {
    "serviceLevel": { "code": "DMCT" }
  }
}
```

Uložit: transactionIdentification = {{transactionId}} a signId = {{signId}}

Krok 5: GET Sign Info

```
curl -X GET "https://{{apigw-hostname}}/psd2/v1/my/payments/paymentId/{{paymentId}}/sign/{{signId}}" \
  --cert {{pathToCertificate}} --key {{pathToKey}} \
  -H "Authorization: Bearer {{bearerToken}}" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -H "Traceparent: {{traceParent}}" \
  -H "User-Involved: true" \
  -H "TPP-Name: {{tppName}}" \
  -H "Date: {{date}}"
```

Response (200):

```
{
  "scenarios": ["RAR"],
  "signInfo": {
    "state": "OPEN",
    "signId": {{signId}}
  }
}
```

Krok 6: POST Sign — získání autorizačního objektu (JWE)

```
curl -X POST "https://{{apigw-hostname}}/psd2/v1/my/payments /paymentId/{{paymentId}}/sign/{{signId}}" \
  --cert {{pathToCertificate}} --key {{pathToKey}} \
  -H "Authorization: Bearer {{bearerToken}}" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -H "Traceparent: {{traceParent}}" \
  -H "User-Involved: true" \
  -H "TPP-Name: {{tppName}}" \
  -H "Date: {{date}}" \
  -H "x-client-cert: {{pathToCertificate}}" \
  -d '{
    "authorizationType": "RAR",
    "redirectUrl": "{{redirectUri}}"
  }'
```

Response (200):

```
{
  "authorizationType": "RAR",
  "signInfo": {
    "state": "OPEN",
    "signId": "{{signId}}"
  },
  "rar": {
    "par_uri": "https://{{authserver-url}}/par",
    "scope": "signId{{signId}}",
    "authorization_details": "{{authorizationDetails}}"
  }
}
```

Krok 7: PAR (RFC 9126)

```
curl -X POST "https://{{authserver-url}}/par" \
  --cert {{pathToCertificate}} --key {{pathToKey}} \
  -H "Authorization: Bearer {{bearerToken}}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -H "x-client-cert: {{pathToCertificate}}" \
  --data-urlencode "authorization_details={{authorizationDetails}}" \
  --data-urlencode "client_id={{clientId}}" \
  --data-urlencode "state={{state}}"
```

Response (201):

```
{
  "request_uri": "{{requestUri}}",
  "expires_in": 300
}
```

Krok 8: Authorize s request_uri (SCA + consent)

```
curl -X GET "https://{{authserver-url}}/authorize\
?client_id={{clientId}}\
&request_uri={{requestUri}}\
&redirect_uri={{redirectUri}}\
&code_challenge={{pkceCodeChallenge}}\
&code_challenge_method=S256"
```

PKCE je povinné — code_challenge (S256) musí být vždy v /authorize, i když se používá PAR. Auth server váže PKCE na /authorize URL.

Klíčový rozdíl PISP vs AISP: response_type, scope a state jsou v rámci PISP součástí PAR objektu (JWE) — nesmí se duplikovat v authorize URL.

Callback vrátí: https://moje-tpp.example.com/callback?code=PispSignCode&state=pisp-sign-state-002

Krok 9: Token Exchange → PISP token pro podpis

```
curl -X POST "https://{{authserver-url}}/token" \
  -H "Authorization: Basic {{authorization}}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "redirect_uri={{redirectUri}}" \
  --data-urlencode "grant_type=authorization_code" \
  --data-urlencode "code={{accessCode}}" \
  --data-urlencode "scope={{scope}}" \
  --data-urlencode "client_id={{clientId}}" \
  --data-urlencode "state={{state}}" \
  --data-urlencode "code_verifier={{pkceCodeVerifier}}"
```

Response (200):

```
{ "access_token": "{{accessToken}}",...}
```

Krok 10: PUT Sign — finální autorizace platby

```
curl -X PUT "https://{{apigw-hostname}}/psd2/v1/my/payments/paymentId/{{paymentId}}/sign/{{signId}}" \
  --cert {{pathToCertificate}} --key {{pathToKey}} \
  -H "Authorization: Bearer {{bearerToken}}" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -H "Traceparent: {{traceParent}}" \
  -H "User-Involved: true" \
  -H "TPP-Name: {{tppName}}" \
  -H "Date: {{date}}" \
  -H "x-client-cert: {{pathToCertificate}}"
```

Response (200):

```
{
  "state": "AUTHORIZED",
  "instructionStatus": "ACTC"
}
```

state: "AUTHORIZED" = platba byla úspěšně autorizována.

9.4 PISP Scénář 2 — Platba s debetním účtem, pouze aplikační token

Kdy použít: TPP zná IBAN debetního účtu předem (např. z předchozí AISP session). Platba se inicializuje s aplikačním (CC) tokenem — implementace podporuje One-SCA pokud jsou známé kompletní údaje platby. Není tedy nutné mít uživatelský token před vytvořením platby.

Klíčový rozdíl oproti scénáři 1: debtorAccount je uveden v těle platby → CC token postačuje pro POST /payments a kroky sign. Uživatelský token se získá až v kroku autorizace (PAR → authorize → token exchange).

Krok 1: Client Credentials Token (PISP)

```
curl -X POST "https://{{authserver-url}}/token" \
  -H "Authorization: Basic {{authorization}}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=client_credentials" \
  --data-urlencode "scope={{scope}}"
```

Response (200):

```
{ "access_token": "{{accessToken}}", ... }
```

Krok 2: Create Payment — s debetním účtem (aplikační token)

```
curl -X POST "https://{{apigw-hostname}}/psd2/v1/my/payments \
  --cert {{pathToCertificate}} --key {{pathToKey}} \
  -H "Authorization: Bearer {{bearerToken}}" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -H "Traceparent: {{traceParent}}" \
  -H "User-Involved: true" \
  -H "TPP-Name: {{tppName}}" \
  -H "Action-ID: {{actionId}}" \
  -H "TPP-Identification: {{tppId}}" \
  -H "X-Request-ID: {{xRequestId}}" \
  -H "Date: {{date}}" \
  -d '{
    "paymentIdentification": {
      "instructionIdentification": "PLATBA-S2-001"
    },
    "amount": {
      "instructedAmount": {
        "value": 250.00,
        "currency": "CZK"
      }
    },
    "debtorAccount": {
      "identification": {
        "iban": "CZ3158000000000002351250"
      }
    },
    "creditorAccount": {
```

```
    "identification": {
      "iban": "CZ5508000000001234567899"
    }
  }
}'
```

debtorAccount je explicitně uveden → CC token postačuje.
Response (200):

```
{
 "paymentIdentification": {
   "transactionIdentification": "{{paymentId}}"
 },
 "signInfo": {
   "state": "OPEN",
   "signId": "{{signId}}"
 }
}
```

Kroky 3–6: GET Sign / POST Sign / PAR / Authorize
Shodné se Scénářem 1, kroky 5–8 — s použitím cc-pisp-token-ABC a nových hodnot transactionIdentification / signId.

Krok 7: Token Exchange → uživatelský token (po authorize)

```
curl -X POST "https://{{authserver-url}}/token" \
 -H "Authorization: Basic {{authorization}}" \
 -H "Content-Type: application/x-www-form-urlencoded" \
 --data-urlencode "redirect_uri={{redirectUri}}" \
 --data-urlencode "grant_type=authorization_code" \
 --data-urlencode "code={{accessCode}}" \
 --data-urlencode "scope={{scope}}" \
 --data-urlencode "client_id={{clientId}}" \
 --data-urlencode "state={{state}}" \
 --data-urlencode "code_verifier={{pkceCodeVerifier}}"
```

Response (200):

```
{ "access_token": "{{accessToken}}", "scope": "{{scope}}", ... }
```

Server automaticky vrátí consent pro PISP po autorizaci platby.

Krok 8: PUT Sign — finální autorizace

```
curl -X PUT "https://{{apigw-hostname}}/psd2/v1/my/payments/paymentId/{{paymentid}}/sign/{{signId}}" \
 --cert {{pathToCertificate}} --key {{pathToKey}} \
 -H "Authorization: Bearer {{bearerToken}}" \
 -H "Content-Type: application/json" \
 -H "Accept: application/json" \
 -H "Traceparent: {{traceParent}}" \
 -H "User-Involved: true" \
 -H "TPP-Name: {{tppName}}" \
 -H "Date: {{date}}" \
 -H "x-client-cert: {{pathToCertificate}}"
```

Response (200):

```
{
"state": "AUTHORIZED",
"instructionStatus": "ACTC"
}
```

9.5 PISP Scénář 3 — Platba s použitím existujícího a platného PISP tokenu

Kdy použít: TPP má platný PISP uživatelský token z předchozí session (uložený access_token + refresh_token se scope PISP nebo AISP PISP). Kroky CC token a authorize se přeskočí — platba se inicializuje přímo s existujícím tokenem.

Pokud je existující access_token expirovaný, nejprve ho obnovte přes refresh grant (viz níže).
(Volitelné) Obnovení existujícího tokenu přes refresh

```
curl -X POST "https://{{authserver-url}}/token" \
  -H "Authorization: Basic {{authorization}}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=refresh_token" \
  --data-urlencode "refresh_token={{refreshToken}}" \
  --data-urlencode "client_id={{clientId}}"
```

Response (200):

```
{
  "access_token": "{{accessToken}}",
  "refresh_token": "{{refreshToken}}",
  "scope": "{{scope}}",
  "expires_in": "3600"
}
```

Refresh Token Rotation: uložit nový refresh_token — původní je zneplatněn.

Krok 1: Create Payment (s existujícím tokenem)

```
curl -X POST "https://{{apigw-hostname}}/psd2/v1/my/payments" \
  --cert {{pathToCertificate}} --key {{pathToKey}} \
  -H "Authorization: Bearer {{bearerToken}}" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -H "Traceparent: {{traceParent}}" \
  -H "User-Involved: true" \
  -H "TPP-Name: {{tppName}}" \
  -H "Action-ID: {{actionId}}" \
  -H "TPP-Identification: {{tppId}}" \
  -H "X-Request-ID: {{xRequestId}}" \
  -H "Date: {{date}}" \
  -d '{
    "paymentIdentification": {
      "instructionIdentification": "PLATBA-S3-001"
    },
```

```
  "amount": {
   "instructedAmount": {
    "value": 50.00,
    "currency": "CZK"
   }
  },
  "debtorAccount": {
   "identification": {
    "iban": "CZ3158000000000002351250"
   }
  },
  "creditorAccount": {
   "identification": {
    "iban": "CZ5508000000001234567899"
   }
  }
 }'
```

Response (200):

```
{
 "paymentIdentification": {
  "transactionIdentification": "{{paymentId}}"
 },
 "signInfo": {
  "state": "OPEN",
  "signId": "{{signId}}"
 }
}
```

Kroky 2–5: GET Sign / POST Sign / PAR / Authorize

Shodné se Scénářem 1, kroky 5–8 — s existujícím refreshed-pisp-token-999 a novými hodnotami transactionIdentification / signId.

Krok 6: Token Exchange → nový PISP token pro podpis a krok 7: PUT Sign jsou opět shodné se získáním finálního tokenu a exekucí platby výše.

Krok 7: PUT Sign

9.6 CISP Flow (Balance Check)

Jednoduchý flow — nevyžaduje autorizaci uživatele.

```
curl -X POST "https://{{apigw-hostname}}/psd2/v1/my/accounts/balanceCheck" \
 --cert {{pathToCertificate}} --key {{pathToKey}} \
 -H "Authorization: {{authorization}}" \
 -H "Content-Type: application/json" \
 -H "Accept: application/json" \
 -H "Traceparent: {{traceParent}}" \
 -H "User-Involved: true" \
 -H "TPP-Name: {{tppName}}" \
 -H "Action-ID: {{actionId}}" \
 -H "TPP-Identification: {{tppId}}" \
```

```
-H "X-Request-ID: {{xRequestId}}" \
-H "Date: {{date}}" \
-H "x-client-cert: {{pathToCertificate}}" \
-d '{
  "exchangeIdentification": "858576010faf0a2309",
  "debtorAccount": {
    "identification": { "iban": "CZ3158000000000002351250" },
    "currency": "CZK"
  },
  "authenticationMethod": "NPIN",
  "transactionDetails": {
    "currency": "CZK",
    "totalAmount": 10050.15
  }
}'
```

10 Důležitá pravidla

10.1 Parametry Authorize a PAR

AISP nepoužívá PAR — všechny parametry jdou přímo do authorize:

Parametr	AISP Authorize	Popis
scope	ANO (AISP)	Požadovaný scope
response_type	ANO (code)	Typ odpovědi
client_id	ANO	Identifikátor klienta
redirect_uri	ANO	Callback URL
code_challenge	ANO	PKCE challenge
code_challenge_method	ANO (S256)	PKCE algoritmus
state	ANO	Náhodný stav pro CSRF ochranu

PISP používá PAR pro předání **platebního objektu**. Parametry z PAR se **NESMÍ** opakovat v authorize:

Parametr	PISP PAR	PISP Authorize
authorization_details	ANO (JWE, singular)	NE
client_id	ANO	ANO
state	ANO	NE
response_type	NE	NE (je v JWE)
scope	NE (je v JWE)	NE
redirect_uri	NE	ANO
code_challenge	NE	ANO
code_challenge_method	NE	ANO (S256)
request_uri	—	ANO (z PAR response)

10.2 Přehled tokenů dle scénáře

Scénář	POST /payments	GET/POST sign	PAR	PUT sign

PISP-1 (bez debtorAccount)	Uživatelský token	Uživatelský token	Uživatelský token	Nový uživatelský token (po authorize)
PISP-2 (s debtorAccount)	CC token	CC token	CC token	Uživatelský token (po authorize)
PISP-3 (existující token)	Existující PISP token	Existující PISP token	Existující PISP token	Nový uživatelský token (po authorize)

10.3 Refresh Token Rotation

- každý refresh grant vrací nový access token + nový refresh token
- Původní refresh token se zneplatní
- Implementace MUSÍ uložit nový refresh token

10.4 Encoding

- Všechny HTTP response body číst s UTF-8 kódováním
- expires_in v token response může být string nebo number
- authorization_details v PAR je URL-encoded (form body)

10.5 Chybové kódy

Jsou definované v rámci Open API definici, která je dostupná v Developer portálu.

11 Přílohy

11.1 Konfigurační checklist

Před zahájením integrace ověřit:

- [ ] kvalifikovaný eIDAS certifikát (QWAC/QSealC), předepsaný regulací (RTS on SCA & CSC – EU 2018/389)
- [ ] Registrace přístupu do Developer portálu
- [ ] Registrace OAuth2 klienta (client_id, client_secret, redirect_uri) v developer portálu
- [ ] Zajištěn síťový prostup k API Gateway (https://apigw.jtbank.cz) — síťová konektivita, proxy
- [ ] Zajištěn síťový prostup k autorizačnímu serveru (https://bezpecnost.jtbank.cz) — síťová konektivita
- [ ] Podpora implementace PKCE (S256)

- [ ] UTF-8 encoding pro všechny HTTP odpovědi
- [ ] Zpracování expires_in jako string i number
- [ ] Refresh token rotation — ukládání nového RT