Corporate API

Szolgáltatás leírás

OTP BANK Nyrt.

Verzió: 3.5
Budapest, 2024.02.06

Verzió történet

Verzió változtatások összefoglalója 3.5

Változások	Érintett szolgáltatás	Fázis
V2: Melléklet hozzáadása	Regisztrációs folyamat	1.
V3: Melléklet hozzáadása	Megbízás beküldése mintafolyamat	1.
V3.1: Folyamat leírás kiegészítés	Üzleti authentikáció, Electra felhasználónév és jelszó használata. Jelszó titkosítás technikai leírása.	1.
V3.2: Folyamat kiegészítés	• Electra kliens telepítése, Electra kezdeti jelszó megváltoztatása, Aláírói jelszó megadása. • Új API szolgáltatás: Adatformátumú és hitelesített PDF számlakivonat lekérdezés • OTP Bank CorporateAPI swagger configmap CorporateToBank Rest API leíró 1.4 verziójában történt változtatások	1.
V3.3: Folyamat kiegészítés	• A szolgáltatás folyamatainak pontosítása, kiegészítése	1.
V3.4: Folyamatleírás kiegészítése, folyamatábrák frissítése	• Regisztrációs folyamat • Regisztrációhoz tartozó folyamatábra	1.
V3.5: Meglévő szolgáltatás leírások módosítása, Új szolgáltatás leírások hozzáadása	• Általános rendelkezések módosítása • Ügyfél-Bank irány technikai szolgáltatás leírás módosítása • Bank-Ügyfél irány technikai szolgáltatás leírás módosítása • Üzleti szolgáltatás feltételek leírás módosítása • Új Ügyfél-Bank API szolgáltatás: Csoportos megbízások kezdeményezés • Új Ügyfél-Bank API szolgáltatás: Csoportos megbízások listázása • Új Ügyfél-Bank API szolgáltatás: Csoportos megbízások lekérdezése • Új Bank-Ügyfél szolgáltatás: Tranzakció státusz értesítés (kötegelt állományok esetében) • Új Bank-Ügyfél szolgáltatás: Kártya foglalási értesítés	2.

Tartalomjegyzék

RÖVIDÍTÉSEK / szakkifejezések

Megnevezés	Leírás
ISO20022	Egyetemes pénzügy ágazati üzenetrendszer.
API	**Application Programing Interface. Alkalmazásprogramozási interfész, amelyen keresztül az adott szolgáltatás elérhető.**
API Gateway	Az OTP API Gateway rendszere biztosítja és támogatja a szabványosított kommunikációt a vállalat alkalmazása és a Bank háttérrendszerei között.
Sandbox környezet	**Tesztelési célra használható környezet, amely a beérkező kérésekre statikus mock válaszokat ad.**
Éles / Production környezet	Valós számlákra vonatkozó lekérdezések és pénzügyi műveletek elvégzését biztosító környezet.
PSU	**Payment Services User. Az Ügyfél azaz a Fizetési Szolgáltatások felhasználója. A PSD2 ökoszisztémához hasonlóan itt is azt az alanyt jelenti, aki a szolgáltató által meghatározott és nyújtott pénzforgalmi szolgáltatást igénybe veszi.**
Electra azonosító és jelszó	Ügyfél authentikálására szolgál, az API szolgáltatás igénybevételéhez szükséges egyedi azonosító és jelszó; valamint az aláírásokhoz szükséges azonosító/jelszó páros is. **Fontos!** Az OTP Bank Electra szolgáltatása és az API szolgáltatása kettő külön önálló szolgáltatás, nincs összefüggés vagy átjárhatóság a szolgáltatások között.

1 ELŐSZÓ

Az OTP Bank Nyrt. – továbbiakban Bank – ebben a dokumentumban definiálja a leendő vállalati ügyfelei számára, hogy milyen üzleti feltételek megléte mellett nyújtja a ***Corporate API szolgáltatását***, tovább részletesen leírja annak technikai követelményeit és megvalósítási folyamatát.

A dokumentum célja, hogy Vállalati Ügyfeleinket támogassa fejlesztési feladataikban minden szükséges információval a Sandbox és az Éles környezeti szolgáltatásaink igényléséhez, valamint azokhoz történő technikai csatlakozáshoz.

2 ÜZLETI FELTÉTELEK A SZOLGÁLTATÁS IGÉNYBEVÉTELÉHEZ

2.1 Általános rendelkezések

A Bank a Corporate API keretében nyújtott szolgáltatások igénybevétele esetén az Ügyfélnek saját rendszeréből közvetlenül nyílik lehetősége megbízások küldésére és bankszámla információk lekérdezésére, fogadására.

Az API szolgáltatás igénybevételének előfeltétele, hogy a nem lakossági Ügyfél számlatulajdonos a Banknál fizetési számlával, valamint OTPdirekt Electra szerződéssel rendelkezzen. Továbbá a Bankkal a Corporate API szolgáltatás igénybevételére kiegészítő megállapodást kössön, és a szükséges nyomtatványokat kitöltve a Bankhoz benyújtsa. A szolgáltatás igénybevételének jogosultsága az OTPdirekt szerződésben és az ahhoz mellékelt OTPdirekt Electra terminál igénybejelentő adatlapokon rögzített feltételekhez igazodik.

A szolgáltatásra vonatkozó szerződés aláírásával a számlatulajdonos saját költségén biztosítja a szolgáltatás igénybevételéhez szükséges, működőképes állapotban lévő Ügyféloldali eszközöket, technikai, műszaki feltételeket, beleértve a hardver és szoftver eszközöket is.

A szolgáltatás az év minden napján, a nap 24 órájában rendelkezésre áll. A Bank fenntartja a jogot, hogy alkalomszerűen (pl.: rendszerfrissítés miatt) a szolgáltatás rendelkezésre állását rövid időre szüneteltesse. A Bank vállalja, hogy az előrelátható szüneteltetés időpontjáról Ügyfeleit a kiegészítő szerződésben megadott elérhetőségeiken értesítse.

Az Ügyfél által kezdeményezett tranzakciók esetében, az OTPdirekt Electra szerződésben meghatározott számlakörre vonatkozó jogosultságok érvényesek.

A Bank által küldött értesítő üzenetek (push) esetében az Ügyfélnek kell meghatároznia, hogy mely számláit szeretné bevonni a szolgáltatásba. A számlaszámok megjelölését követően pedig **meg kell küldenie az OTP Bank számára, hogy a Bank által küldendő üzeneteket pontosan mely végpontokon várja**. Ennek hiányába a szolgáltatás nem képes a rendeltetésének megfelelően működni. A felhasználó köteles a jelszót és az azonosítót titokban tartani, valamint biztosítani, hogy a szolgáltatás igénybevételéhez szükséges eszközökhöz illetéktelen harmadik személy ne férjen hozzá. **A Bank fenntartja a jogot – az üzenetek érzékeny adattartalmai miatt - a megadott Ügyfél végpontok periodikus ellenőrzésére.**

2.2 Biztonság

A Corporate API használatával elérhető szolgáltatások igénybevételét a Bank által átadott Éles API kulcs biztosítja. Egy API kulcs egy szerződéssel és egy Ügyfél által használható fel. Amennyiben ugyanazt az API kulcsot több Ügyfél is fel kívánja használni, úgy minden Ügyfélnek külön-külön szerződést kell kötnie, és a díjazást minden szerződés után külön-külön meg kell fizetni.

A Bank fenntartja a jogot, hogy a Corporate API keretében összekapcsolódásra használt API-t bármikor módosítsa. Ez esetben a további működéshez esetlegesen szükséges fejlesztéseket az Ügyfél saját maga végzi el saját költségére. A várható API módosításról a Bank az ügyfelet megfelelő felkészülési idő biztosításával előzetesen tájékoztatja.

2.3 Megbízások

Az Ügyfél a rendelkezésre jogosult API-n keresztül adhat fizetési megbízást a Banknak, a Bankhoz történő érkezési időpontnak a Bank számítógépes rendszere által megállapított időpont tekinthető. A Bankhoz beérkezett több tételt tartalmazó állományon belüli sorrend nem azonos a tételek feldolgozási sorrendjével.

A Bank a Corporate API-n benyújtható megbízások körére és azok összegére a Kondíciós Listában és az Üzleti Feltételekben limitet határozhat meg.

2.4 Kamatok, díjak, jutalékok, költségek

A szolgáltatás használatáért a számlatulajdonos a Banknak díjat köteles fizetni. A számlatulajdonos által igényelt szolgáltatások mindenkori díjai a számlatulajdonossal kötendő szerződésben, illetve a Kondíciós Listában kerülnek meghatározásra.

3 TECHNIKAI FELTÉTELEK ÉS KÖVETELMÉNYEK A SZOLGÁLTATÁS IGÉNYBEVÉTELÉHEZ

3.1 Ügyfél – Bank irány

Vállalati ügyfelünknek a Sandbox, illetve Éles Corporate API szolgáltatásunkhoz való kapcsolódása esetén a következőket kell teljesítenie:

3.1.1 Regisztráció – Sandbox környezet

A regisztráció előfeltétele, hogy Ügyfelünk rendelkezzen egy megbízható tanúsítvány kibocsátó által aláírt tanúsítvánnyal, melyet az API hívások során a szolgáltatást igénybe vevő alkalmazás authentikációjához fog felhasználni.

A regisztráció során az alábbi adatokat szükséges megküldeni a corpapisupport@otpbank.hu email címre:

- ***Ügyfél vállalat neve**:* az alkalmazást birtokló vagy támogató szervezet neve.
- ***E-mail cím a kapcsolattartáshoz**:* Az alkalmazás támogató csapatának csoportos e-mail címe. Technikai problémák esetén ezen az e-mail címen vesszük fel a kapcsolatot az ügyfelünkkel.
- ***Tanúsítvány**:* Az Ügyfél hitelesítési folyamathoz használt, megbízható tanúsítvány kibocsátó által aláírt tanúsítványának publikus része PEM formátumban (a teljes tanúsítvány lánc szükséges). Lehetséges küldési megoldások:
    - *zip*-ben az email mellékleteként, vagy
    - *email*-be ágyazva szövegesen.

Amennyiben a megadott adatok helyesek munkatársaink válaszlevélben **8 munkanapon belül** megküldik a szükséges API kulcsot (client_id) a Sandbox környezethez. Ez a kulcs lesz az Ügyfél oldal azonosítója az OAuth 2.0 engedélyezési folyamatban. (További részletekért kérem tekintse meg a 6.1. pontban található ábrát.)

Tanúsítvány lejárat esetén a csere előtt **legalább 8 nappal szükséges** átadni az új tanúsítványt, jelezve a tanúsítvány kicserélésének pontos időpontját.

3.1.2 Regisztráció – Éles környezet

A ***Kiegészítő megállapodás*** aláírását követően biztosított az Éles környezethez való hozzáférés, melyhez Ügyfelünknek a számlavezetési referensével szükséges felvennie a kapcsolatot. (*További részletekért kérem tekintse meg a 6.2. pontban található ábrát.)*

3.1.3 Authentikáció folyamata

1. JWT generálása és aláírása az átadott tanúsítvány segítségével.
2. /oauth2/token végpont meghívása a megadott adatokkal (login). Sikeres hitelesítés estén válasz üzenetben az access token visszaadásra kerül. Kérjük, hogy a kapott access tokent szolgáltatás igénybe vevő oldalon tárolják és azt lejáratig használják. Indokolatlan mennyiségű (például minden kéréshez új) access token haszálata nem támogatott.
3. Az access token használatával, a szolgáltatás igény bevevő HTTPS-kérés(eke)t küld az openAPI leíróban szereplő végpont(ok)ra.
4. A Bank a kéréseket fogadja, feldolgozza, és válaszol azokra.
5. Az access token lejárattal rendelkezik, ezért lejárat előtt szükséges megújítani (/oauth2/refresh_token), vagy újat kérni igényelni (/oauth2/token)
6. Amennyiben lejárati időn belül már nem várható API hívás a token visszavonható a /oauth2/revoke_token végponton keresztül (logout).

3.1.4 Technikai követelmények

3.1.4.1 REST-API hívások kezdeményezése HTTPS-en keresztül

Támogatott csatorna protokoll: HTTP/1.1 – TLS 1.2 titkosítással.

3.1.4.2 JSON formátumú adatok létrehozása és feldolgozása

Kérés – válaszokban történő üzleti adattartalom küldése és fogadása JSON formátumban történik a CorporateAPI yaml leíróban definiált struktúra szerint.

3.1.4.3 PULL szolgáltatásokhoz használt érvényes tanúsítvány biztosítása

Szeretnénk felhívni a figyelmét, hogy az Ügyfél-Bank irányú kommunikációs kapcsolat biztosításához érvényes tanúsítványra van szükség. Ezért a Corporate API szolgáltatás zavartalan használata érdekében gondoskodjon a Sandbox és az ÉLES környezetben használt tanúsítvány(ok) lejárata előtt, legalább 2 héttel az új tanúsítvány(ok) CorpAPISupport@otpbank.hu címre történő eljuttatásáról.

3.1.5 Authentikáció

3.1.5.1 JWToken generálása

Az Ügyfél-hitelesítési folyamat során a Bank egy JWT Assertiont használ az RFC7523 -ban (JWT Bearer token authorization grant type for OAuth 2.0) meghatározott tokencseréhez. A JWT-t általánosságban az RFC7519 határozza meg.

Az API szolgáltatást használó alkalmazás maga generálja a JWT Assertion-t, amelyhez a Regisztrációkor átadott tanúsítványt használja. A tanúsítványt API Gateway-ben úgy konfigurálják, hogy ellenőrizze a JWT-aláírás érvényességét az alkalmazás számára.

Mező szintű korlátozások:

A JWT-t az RSA SHA256 algoritmussal kell aláírni.

Mező Név	Mező elnevezés	Self-Issued JWT értéke
	Header	
alg	Algoritmus	RS256
	Payload	
iss	Kibocsátó (Issuer)	API kliens azonosító
aud	Közönség (Audience)	Címzett host: api-ext.otpbank.hu vagy api-sandbox.otpbank.hu
sub	Tárgy (Subject)	API kliens azonosító (client_id)
exp	Lejárati idő (Expiration time)	A token létrehozásának időpontjától számított 3 percen belüli időbélyegző.
iat	Kibocsátva (Issued At) - *opcionális*	A token generálásának időbélyege.
jti	JWT azonosító (JWT ID) - *opcionális*	A JWT egyedi azonosítója. Ha egy (elosztott) alkalmazás egyszerre több JWT-t használ, akkor ez megakadályozza az alkalmazás több példánya által egyszerre generált JWT-k ütközését. Arra is használható, hogy megakadályozza a JWT újrafelhasználását.

Példa JWT payload:


```
{
 "iss": "apiKey123",
 "aud": " api-sandbox.otpbank.hu  ",
 "sub": "apiKey123",
 "iat": 1598967249,
 "exp": 1598967549,
 "jti": "apigw6098364921"
}
```


3.1.5.2 Access token igénylése (login)

A JWT-t egy opaque OAuth access tokenre szükséges cserélni. Az access tokent minden későbbi üzleti típusú API-hívás esetében kötelezően meg kell adni. Az access token élettartammal rendelkezik és több API-híváshoz újra felhasználható, amíg le nem jár. Figyelni kell arra is, hogy nem szabad minden egyes API-híváshoz új access tokent kérni.

A szolgáltatás igénybe vevőnek meg kell hívnia a token végpontot, hogy megkapja az OAuth2 access tokent a JWT Assertion-val és a client_id-vel (alkalmazási kulcs) együtt. Példa:

```
POST /oauth2/token HTTP/1.1
Host: api-sandbox.otpbank.hu
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-
bearer&client_id=Xdfsdvoc7KX5Gezz9le745UEql5dDmj&assertion=eyJpc3MiOiAiM0
1WRz...[ az átláthatóság kedvéért kihagyva a teljes assertion leíró]...ZT
```

3.1.5.3 Access token megújítása (refresh)

Access token lejártát követően lehetőség van annak megújítására a login során kapott refresh token segítségével annak életciklusán belül. Példa:

```
POST /oauth2/refresh_token HTTP/1.1
Host: api-sandbox.otpbank.hu
Content-Type: application/x-www-form-urlencoded
grant_type=refresh_token&refresh_token&rVSmm3QaNa0xBVFbUISz1NZI15akvgLJ
&client_id=Xdfsdvoc7KX5Gezz9le745UEql5dDmj
```

3.1.5.4 Access token visszavonása (logout)

Amennyiben az access token-re életciklusán belül már nincs szükség, akkor visszavonható. Példa:

```
POST /oauth2/revoke_token HTTP/1.1
Host: api-sandbox.otpbank.hu
Content-Type: application/x-www-form-urlencoded
access_token=AnoHsh2oZ6EFWF4h0KrA0gC5og3a&client_id=Xdfsdvoc7KX5Gezz9l
e745UEql5dDmj
```

3.1.5.5 Az API Gateway OAuth 2 szolgáltatásainak openAPI leírója

A vonatkozó OAuth2 CorporateAPI yaml leírója a szolgáltatás leíró dokumentumot tartalmazó levél csatolmányaként került átadásra.

Vonatkozó yaml fájl:

OTP_Bank_CorporateAPI_swagger_configmap_CorporateToBank_oauth2_0-ext_v1.yaml

3.1.6 Tranzakció azonosító

A tranzakcióazonosítók (Request ID, Correlation ID) a rendszereken keresztülmenő kérések és üzenetek azonosítására és korrelálására szolgálnak. A Request ID, azaz kérési azonosító segítségével kapcsolható össze az adott híváshoz/kéréshez kapcsolódó összes esemény. A Correlation ID segítségével pedig a külső alkalmazás/rendszer kapcsolhatja azon elemi hívásokat, melyek az adott folyamathoz tartoznak. A tranzakció azonosítókat az adott API hívás/kérés HTTP header elemében szükséges küldeni a CorporateAPI yaml leíróban definiáltak szerint.

Ezen tranzakció azonosítók a rest API kérések nyomonkövethetősége, valamint hibakeresések támogatása céljából kerültek definiálásra. Nem összetévesztendőek az üzleti vonatkozású End-to-End azonosítókkal, melyek az adott kérelmek üzleti adat (BnsDta) payload részében utaznak.

3.1.6.1 Részletek

A Bank Corporate API szolgáltatása két HTTP fejléc mezőt használ a tranzakciók azonosítására. Ezek a fejlécek kötelezőek.

HTTP Fejléc Mezők	Leírás	Megjegyzés
X-Request-ID	A kérelem egyedi azonosítója.	A kérelem azonosítója mindig egyedi, ami azt jelenti, hogy nincs olyan tranzakcióazonosító, amely azonos lenne. Ezt a tranzakciókulcsot a kérelem kezdeményezője generálja, azaz Ügyfél felelőssége, hogy az azonosító egyedi legyen.
X-Correlation-ID	Több, egymással összefüggő kérés egyedi azonosítója (üzleti munkamenet).	A korrelációs azonosító mindig egyedi a kérelem azonosítójával együtt. Olyan kérésekhez és üzenetekhez kapcsolódik, amelyek lehetővé teszik az egyedi keresést egy adott tranzakcióra, vagy eseményláncra. Egyedi tranzakció esetén az X-Correlation-ID megegyezhet az X-Request-ID-val.

3.1.6.2 Követelmények

Az ügyfeleknek az előírásoknak megfelelően mind az X-Request-ID-t, mind pedig az X-Correlation-ID-t generálniuk és küldeniük szükséges az összes API kérésük folyamán. Mivel a tranzakció azonosítók kötelező mezők, ezért az OTP Bank Corporate API szolgáltatása minden olyan API kérést, ami nem tartalmazza ezeket a kötelező mezőket "400 - as Bad Request" szabványos HTTP-válaszkóddal utasít vissza.

Példa hibaüzenetet:

```
{
    "fault": {
        "faultstring": "Missing mandatory request header. X-Request-ID and X-
Correlation-ID headers must be present in the request.",
        "detail": {
            "errorcode": "RaiseFault.MissingMandatoryHeader"
        }
    }
}
```

3.1.6.3 Regulációk

A kérelem azonosítójának mindig egyedinek kell lennie. Az X-Request-ID és az X-Correlation-ID minimális karakter hossza 8, a maximális hossza pedig nem haladhatja meg a 48 karaktert. Minden egyes karakternek érvényesnek kell lennie az ISO-8859-1 karakter készlet szabványnak.

Ajánlások

Mezők	Megoldási javaslat	Példa
X-Request-ID	Az UUID (Universally Unique Identifier) egy 128 bites szám, amelyet a számítógépes rendszerekben lévő információk azonosítására használnak.	df098eee-a623-4212-8688-8c4c6de0ec4c
X-Correlation-ID	<Ügyfél alkalmazásának rövid neve>-<uuid>	UGYFALK-df098eee-a623-4212-8688-8c4c6de0ec4c

3.1.6.4 Hiba válaszok

Általánosságban

A Corporate API szolgáltatás által visszaküldött hiba válaszok a következőkből állnak:

- A HTTP válasz üzenet kódja nem 200-as.
- A HTTP válasz header fejlécében X-Error-Source elem a következő értékek egyikével:
  - "Backend": ez azt jelzi, hogy a hibát a háttér rendszer generálta.
  - "Gateway": ez azt jelzi, hogy a hibát az API-átjáró (APIGee) generálta.
- Ha az X-Error-Source header elemeben kapott érték "Gateway", és az eredeti kérés Content-Type-ja application/json, vagy a Content-Type nincs definiálva (pl. GET kérések esetén), akkor ebben az esetben a szolgáltatás igénybe vevő Ügyfél egy JSON választ kap vissza, mely tartalmaz egy faultstringet, valamint egy hibakódot. Példa:

```
{
  "fault":
    {
      "faultstring": "The JWT format is not valid: ...",
      "detail":
        {
          "errorcode": "RaiseFault.InvalidJWTFormat"
        }
    }
}
```

A dokumentáció jelen része a http jellegű hibákkal kapcsolatos információkat definiálja. Az üzleti jellegű hibákat a későbbiek folyamán részletezi.

Hiba típusok OAuth2 hívások esetében

Az OAuth2 hívások a következő HTTP hibakódokat adhatják vissza. A válasz üzenet Body részében található faultstring tartalmazza a hiba okát.

Típus	Hiba esemény	HTTP hiba kód	Hiba kód	Intézkedés
Not found	A kért útvonal nincs definiálva az OAuth2 API proxyban	404	404NotFound	A kérési útvonal javítása (pl.: /oauth2/token)
Invalid Access Token	A megadott JWT nem érvényes (rossz aláírás, stb.) vagy az Ügyfél_id nem létezik. További információért lásd https://docs.apigee.com/api-platform/reference/policies/oauthv2-policy#errorreference	401	InvalidToken	Vegye fel a kapcsolatot az OTP Bank Corporate API üzemeltetői területével.
Bad Request	A kérés formátuma érvénytelen, pl. nem felel meg az RFC7523 szabványnak.	400	InvalidRequestJWT	Javítsa a JWT-t a leírásnak

				megfelelően: JWT Assertion
Token Expired	A kérésben megadott token érvényessége lejárt.	401	TokenExpired	Generálja újra a JWT-t és ismételje meg a kérést.
Spike Arrest Fault	https://docs.apigee.com/api-platform/reference/policies/spike-arrest-policy#runtime-errors	429		
Internal server error	A szolgáltatás server leállt, vagy nem elérhető.	500		Ismételje meg a kérését 3 alkalommal 5 percen belül. Ha a kapott válasz üzenet továbbra is 500-as, akkor kérjük vegye fel a kapcsolatot az OTP Bank Corporate API üzemeltetői területével.

Hiba típusok API hívások esetében

Az OAuth2 hívások a következő HTTP hibakódokat adhatják vissza. A válasz üzenet Body részében található faultstring tartalmazza a hiba okát.

Típus	Hiba esemény	HTTP hiba kód	Hiba kód	Intézkedés
Invalid Access Token	https://docs.apigee.com/api-platform/reference/policies/oauthv2-policy#errorreference	401	keymanagement.service.invalid_access_token	Generáljon új access token-t.
Not found	A kérésben használt útvonal nem definiált az OTP Corporate API szolgáltatásban.	404	RaiseFault.404NotFound	Vegye fel a kapcsolatot az OTP Bank Corporate API üzemeltetői területével.
				Vegye fel a kapcsolatot az OTP Bank Corporate API üzemeltetői területével.
Spike Arrest Fault	https://docs.apigee.com/api-platform/reference/policies/spike-arrest-policy#runtime-errors	429	policies.ratelimit.SpikeArrestViolation	Vegye fel a kapcsolatot az OTP Bank Corporate API üzemeltetői területével.
Quota Fault	https://docs.apigee.com/api-platform/reference/policies/quota-policy#errors	429		Vegye fel a kapcsolatot az OTP Bank Corporate API üzemeltetői területével.
Concurrent Rate Limit Fault	https://docs.apigee.com/api-platform/reference/policies/concurrent-rate-limit-policy#errorcodes	429		Vegye fel a kapcsolatot az OTP Bank Corporate API üzemeltetői területével.
Default	Egyéb hiba	429		Vegye fel a kapcsolatot az OTP Bank Corporate API üzemeltetői területével.
Internal Server Error	A háttér rendszerek nem elérhetőek.	500	messaging.adaptors.http.flow.ErrorWritingRequestToTarget	Vegye fel a kapcsolatot az OTP Bank Corporate API üzemeltetői területével.

Bad Gateway	https://docs.apigee.com/api-platform/troubleshoot/runtime/502-bad-gateway-mp-ssl-backend https://docs.apigee.com/api-platform/troubleshoot/runtime/502-bad-gateway-timeout Megjegyzés: ez HTML hiba választ adhat vissza!	502		Vegye fel a kapcsolatot az OTP Bank Corporate API üzemeltetői területével.
Backend server unavailable	https://docs.apigee.com/api-platform/troubleshoot/runtime/503-service-unavailable	503		Vegye fel a kapcsolatot az OTP Bank Corporate API üzemeltetői területével.
Gateway timout	https://docs.apigee.com/api-platform/troubleshoot/runtime/504-gateway-timeout	504	messaging.adaptors.http.flow.GatewayTimeout	Vegye fel a kapcsolatot az OTP Bank Corporate API üzemeltetői területével.

3.2 Bank – Ügyfél irány

Az itt nyújtott szolgáltatás során a Bank – mint hívó fél – REST API formájában kérést kezdeményez a vele szerződésben álló vállalati Ügyfél rendszerének irányába. Az ehhez szükséges követelményeket hivatott definiálni a dokumentum ezen része.

3.2.1 Korlátozások

- A Bank API kérését fogadó ügyfél alkalmazás csatorna protokolljától elvárt a HTTP/1.1 használat TLS 1.2 titkosítással.
- A Bank API kérést fogadó alkalmazásnak képesnek kell lennie az üzenetek fogadására.
- Az alkalmazásnak képesnek kell lennie a REST API hívások során használatos lapozási funkcionalitással. Ez olyan esetekben fordulhat elő, amikor a Bank által küldeni kívánt teljes üzenet tartalma meghaladja a – banki oldalon engedélyezett - maximális méretet. Ebben az esetben a bank nem egy üzenetben fogja küldeni, hanem "lapokra" bontja. Az egyes üzenetekben pedig jelöli az Ügyfél számára, hogy melyik oldalt küldte, és van-e még érkező oldal.
- Az Ügyfél alkalmazásának törekednie kell arra, hogy beérkezések feldolgozásának és megválaszolásának válaszideje maximum 2 másodperc legyen.
- Meg kell határozni a szolgáltatás ügyfél oldali rendelkezésre állási időablakát (például 7x24, M-F 07:00-18:00 stb.), beleértve a műveletek támogatását is. A döntést követően minden szükséges információt meg kell küldeni a corpapisupport@otpbank.hu email címre (pl.: meghatározott időablak, Ügyfél oldali támogatási osztály elérhetősége, stb.)
- A szolgáltatás minimum elvárt teljesítménye 10 TPS (tranzakciók per másodperc).

3.2.2 Authentikáció

A Corporate API a következő hitelesítési típusokat támogatja (az összes adatforgalmat TLS védi). Az Ügyfél alkalmazása kizárólag a Bank megadott komponensein keresztül fogadhat kérést, melynek a következőket kell biztosítania:

- mTLS kommunikáció: a szolgáltatás igénybevevő (API Gateway) szerverének azonosítása, és
- JWT token kezelés. A kérés header elemei között kerül továbbításra a Bank által generált JWT Bearer séma használatával. A JWT feldolgozásához, aláírásának validálásához szükséges tanúsítványt (***OTP_Bank_Nyrt._Corporate_API_PUSH_notif_trustedcer.zip fájlban található api.otpbank.hu.cer***) a Bank megküldi az Ügyfél számára.
- A JWT-t a Bank alkalmazása írja alá, és a következő állításokat tartalmazza:
    - **sub** mező: hívó rendszer által átadott azonosító
    - **exp** mező: az Assertion lejáratának idejét. Az 1970-01-01T0:0:0Z-től számított másodpercek számában kifejezve, UTC-ben mérve.
    - **iss** mező: jwt kibocsátója, apigee-prod-external-sandbox vagy apigee-prod-external-prod értéket vehet fel
    - **aud** mező: A cél nevét.
    - **client_app_name** mező: A hívó alkalmazás nevét.

Példa JWT tartalom JSON formátumban:


```
{
 "sub": " https://api-sandbox.otpbank.hu",
 "aud": " jwt_audience",
 "client_app_name": "OTPTestApplication",
 "iss": "apigee-prod-external-sandbox",
 "exp": 1575821438,
 "iat": 1575792638,
 "jti": "6df26b4f-f6ed-4225-ba34-2d79cc46bd47"
}
```


Az API-hoz való hozzáférés engedélyezésnek a JWT aláírásának érvényességén kell alapulnia.

Teljes, kódolt JWT-minta:

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJuTWlWM2EzczVOcGpvQnNYTm83WGx2R1dCT0wyeEF2cCIsImF1ZCI6InRlc3RfYmFja2VuZCIsImNsaWVudF9hcHBfbmFtZSI6Ik9UUFRlc3RBcHBsaWNhdGlvbiIsImlzcyI6ImFwaWdlZS1kZXYtaW50ZXJuYWwtZGV2IiwiZXhwIjoxNTk0MjIwMTc2LCJpYXQiOjE1OTQxOTEzNzYsImp0aSI6IjgzMjRjOWJjLTE3NjEtNDYwZi1hYmU5LTlmYmZlYWQ3NWVmMSJ9.BV2v-442kO91dYBDDbEuJBfy1_hqg6gBAWLARFOn9tZuG27mCh2BIxZeMEKymjig1L0ibd_Y-SVcctEAZeSpiJledH8-olvr79WRBuy7R2z1ZSMeSNb66dYvoYpxF1ON_C4QC33JUizLIJfuQ8cHhepCZVHbcpmOWgRgmumZxGvZIzOv4rYz4H7uoBKUYr68qgdt50_kKgXO1NtvvNMjPGSNmhg3os7hbWaHEEg54eyAFKHN7qyecEcjzZHgsb5gTCocWW3ZbdhQ_xfrtvsZsRMQmMiYMur5iIX2KrashtcVcqWwaFlnEXP7fbw19E2T2fEP2KN4KxAKEWN0GChYYw

A Banktól az Ügyfél rendszerei felé irányuló összes HTTP-kérés az alábbi header elemket tartalmazza:

- X-Request-ID: Az adott kérés teljesen egyedi azonosítója.
- X-Correlation-ID: A tranzakció egyedi azonosítója.

A fenti fejlécek értékeit az Ügyfél oldalon naplózni kell az alkalmazás nyomon követhetőségének, a hibák észlelési és elemzési folyamatának javítása érdekében.

4 CORPORATE API ÜZLETI SZOLGÁLTATÁSOK

4.1 Ügyfél – Bank irány

A vonatkozó Corporate API üzleti szolgáltatások openAPI leírója a szolgáltatás leíró dokumentumot tartalmazó levél csatolmányaként került átadásra.

Vonatkozó yaml fájl: (a verziószám jelölés eltérő lehet):

OTP_Bank_CorporateAPI_swagger_configmap_CorporateToBank_v1.6.0.yaml

4.1.1 Általános HTTP fejléc elemek

Ahhoz, hogy a Corporate API üzleti szolgáltatásait a Bank ügyfelei meg tudják hívni, ahhoz szükséges a fentebb már említett OAuth2 hívás során megszerzett access token, melyet az üzleti hívások fejlécében szükséges küldeni az erre létrehozott specifikus mezőben (Authorization). A mezőbe a "Bearer" + " " fix értékeket követően szükséges megadni a korábban megkért access tokent.

4.1.2 Általános Payload elemek

Az Ügyfél által a Bank irányába kezdeményezett REST API hívások payload tartalmát API kérés típusoktól függően 2-3 részt különböztethetünk meg:

- Kontextus (*Cntxt*)
- Ügyfél authentikáció (*CrptAuthNfmtn*)
- Üzleti tartalom (*BnsDta*)

4.1.2.1 Kontextus

A Kontextus (Cntxt) rész négy alcsoportban tartalmazza a tranzakcióval kapcsolatos környezeti információkat:

- Az Ügyfél (PSU) és a Bank közötti, a szolgáltatási kérelem létrehozásához használt http-csatorna leírása. (psu-http-Hdr). Ez egy kulcs-érték tömb.
- Eszközzel kapcsolatos információ, amely leírja a PSU által a tranzakció megbízásának létrehozásakor használt tényleges eszközt. (Devc). Kulcs-érték tömb.
- Az Ügyfél rendszerre vonatkozó adatok. (Extnl)
- PSU-val kapcsolatos információk. (PSU)

4.1.2.1.1 psu-http-Hdr:

kulcs lista elemei:

- Accept
- AcceptCharset
- AcceptEncoding
- AcceptLanguage
- User-Agent
- AgentType
- AgentName
- AgentVersion
- ClientLanguage

- DeviceName
- DeviceVersion
- OSName
- OSVersion
- HttpHeader-SecurityLevel
- UserAgentToken
- X-Forwarded-For
- Proxy1IP
- Proxy2IP
- RemoteIP
- CustHeader

4.1.2.1.2 Devc:

kulcs lista elemei:

- DeviceHash
- ClientAppHash
- HashAlgorithmVersion
- Device-ID
- BrowserPlugin
- Cookie
- CookieEnbl
- CPUTyp
- Font
- Pdf
- Flash
- Java
- Silverlight
- Dotnet
- Mem
- Stor
- Disp
- SuperCookie
- IMEI
- MSI
- MSISDN
- SIM

4.1.2.1.3 Extnl:

TxId:

Kötelező mező. A tranzakció egyedi azonosítója a külső, hívó rendszerben (pl. Vállalati Ügyfél) - az ugyanahhoz a tranzakcióhoz tartozó üzeneteknek ugyanaz a tranzakció azonosítójuk lehet. Formátum követelmény: Az első nyolc karakter egyértelműen azonosítja a vállalati ügyfelet. (pl. a cégnév rövid rövidítése: OTPBANK, OTPEP).
Példa: COMPNAME-85c744acb6bb8a281f088c79

RqstId:

Kötelező mező. A külső rendszer által kezdeményezett kérelem azonosítója. Globálisan egyedi, de meg kell egyeznie a Cntxt/Extnl/TxId mező értékével.
Példa: COMPNAME-85c744acb6bb8a281f088c79

SsnId:

Kötelező mező. Munkamenet azonosító a külső rendszerben (pl. Vállalati Ügyfél).
Példa: 1234567

4.1.2.2 Ügyfél authentikáció

Az Ügyfél authentikálására szolgáló rész (CrptAuthNfmtn), ahol a szolgáltatás igénybevételéhez szükséges egyedi Electra azonosító és jelszó, valamint az aláírásokhoz szükséges azonosító jelszó párosokat kell megadni. Két részre bontható:

- Vállalati Információ (CrpNfmtn)
- Vállalati Aláíró Részletek (CrpSgnDtls)

FONTOS! Abban az esetben, ha a Corporate API szolgáltatását igénybe vevő Ügyfél korábban nem rendelkezett OTPdirekt Electra hozzáféréssel, vagy új OTPdirekt Electra szerződés keretében kívánja igénybe venni azt, akkor a következő telepítéseket és beállításokat szükséges elvégeznie a Corporate API szolgáltatás igénybevételéhez:

1. Electra kliens telepítése a csatolmányként küldött Electra_telepitesi_utm_7_00_02_verziotol.pdf leíró dokumentáció alapján kell elvégezni.
2. Electra kezdeti jelszó megváltoztatása. Melyet az Electra kliens első bejelentkezésekor automatikusan kikényszerít.
3. *(Opcionális)* Electra aláíró jelszó megadása, melynek folyamatát szintén a telepítési útmutató tartalmazza (4.8-as pont).

4.1.2.2.1 Vállalati Információ

Vállalati információk rész alatt szükséges a szolgáltatást igénybe vevő vállalati ügyfeleknek az Electra rendszerhez létrehozott azonosítóját és jelszavát megadnia.

- **CrpIdntfctnData:** Kötelező mező. Az Ügyfél Electra rendszerhez tartozó azonosító kódja.
- **CrpPsswrd:** Kötelező mező. Az Ügyfél jelszava az Electra rendszerhez. Az értéket RSA4096-tal kell titkosítani. A titkosításhoz szükséges publikus kulcsot (***OTP_Bank_Nyrt._Corporate_API_Electra_encryption.zip fájlban található OTP_Bank_Nyrt._Corporate_API.cer***) az Ügyfél - a Corporate API szolgáltatás szerződés létrejöttét követően – számára a Corporate API üzemeltetői területe küldi meg.

4.1.2.2.2 Vállalati Aláíró Részletek

A vállalati aláíró részletek rész a Bank Corporate API kiajánlott szolgáltatásainál a Fizetési megbízásokat érinti jelenleg. Az Ügyfeleknek a Fizetési igények feladásához szükséges az aláírók megadása, melyet a vonatkozó azonosító/jelszó párossal képesek megtenni.

- **SgnrIdntfctnData:** Az aláíró Ügyfél felhasználóneve.
- **SgnrIdntfctnPsswrd:** Az aláíró felhasználóhoz tartó jelszó, mely RSA4096-tal kell, hogy titkosítva legyen. A titkosításhoz hasonlóan az Ügyfél itt is azt a publikus kulcsot (***OTP_Bank_Nyrt._Corporate_API_Electra_encryption.zip fájlban található OTP_Bank_Nyrt._Corporate_API.cer)*** kell, hogy használja, amelyet aBank Corporate API üzemeltető területe küldött meg a számára.

4.1.2.3 Üzleti tartalom

A Bank által nyújtott minden egyes API szolgáltatásnak van egy "Üzleti adatok" (BnsDta) része, amely a szolgáltatásspecifikus adatokat tartalmazza. A vonatkozó API szolgáltatás leíró swagger dokumentáció részletesen tartalmazza az üzleti tartalmi rész követelményeit, mező szintű információit.

4.1.3 Válasz kódok

A Bank válaszüzeneteiben a következő http hibakódok szerepelnek:

HTTP válasz kódok	Leírás	Megjegyzés
200	Sikeres kérés	A kérést sikeresen feldolgozták (a feldolgozás eredménye a válasz payload-jában található)
400	Rossz kérés, érvénytelen bemenet, érvénytelen objektum.	
405	A metódus nem engedélyezett	
406	Nem elfogadható	A kiszolgáló nem tud olyan választ adni, amely megfelel a kérés proaktív tartalomtárgyalási fejlécében meghatározott elfogadható értékek listájának
500	Belső Szerver Hiba	
503	Szolgáltatás Nem elérhető	

A többi HTTP státuszkód és hibakód az RFC 6749 5.2. szakasza szerint került definiálásra. https://tools.ietf.org/html/rfc6749#section-5.2

4.1.4 Alkalmazás authentikáció

Az API szolgáltatások igénybevételéhez az Ügyfeleknek elengedhetetlen alkalmazás szinten authentikálni magukat. Az alkalmazás szintű authentikációra az OAuth2 Rest API szolgál, melynek sikeres eredményeként a Rest API hívás válaszában szerezhetik meg az üzleti hívásokhoz szükséges access tokent. A megszerzett access token felhasználásának segítségével ezt követően már hívhatók az üzleti típusú Rest API szolgáltatások. Részletes leírás jelen dokumentáció 3.1.5 pontjában.

4.1.5 Üzleti szolgáltatások authentikáció

A Corporate API szolgáltatások igénybevételéhez a Bank Ügyfeleinek rendelkeznie kell OTPdirekt Electra szolgáltatás szerződéssel, valamint ehhez tartozófelhasználó névvel és jelszóval. A 4.1.4-es pontban leírt első körös sikeres authentikációt követően az Ügyfeleknek a Corporate API szolgáltatások esetében felhasználói szinten is authentikálni kell magukat, melyet az OTPdirekt Electra felhasználó névvel és a titkosított, majd base64-el elkódolt jelszóval tehetnek meg.

A titkosítás RSA algoritmussal kell, hogy történjen, melyhez a Bank által megküldött publikus tanúsítványból (***OTP_Bank_Nyrt._Corporate_API.cer***) kell az Ügyfeleknek a Publikus kulcsot kigenerálva felhasználnia. A publikus kulcs kigenerálásához használható a következő parancssor: `openssl x509 -pubkey -noout -in OTP_Bank_Nyrt._Corporate_API.cer > pubkey.pem`

Java programozási nyelvben pedig a teljes titkosítás a következőképpen valósulhat meg:

```
import java.io.IOException;
import java.security.Key;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.PublicKey;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;
import javax.crypto.Cipher;

public class ElectraPwdEncryptForOTP {

	/**
	 * Public key recieved from OTP which will be used for enryption
	 */
	static String publicKeyString =
"MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAzxVn9DiFQB3tvoxIIGt/"
			+ "AB9309rJknYv17n0sowMMX3k2frWiM0I9SCayLWWqf25b7hd+JIXg3801aqN4MPw"
			+ "AlIMx1VWIXJ6jJgeXdvOl5gqaP5yuwiV4YjREbwqg/cyrBZOgsXJc9veGSeE+QqV"
			+ "Y5LFfNDjRb7mluL8YvWH8PZk9jDiDh7Ao5l1USi21NLheceTJgAmE7az3UaQgPjF"
			+ "EsL9/JiahT4xmvczxeq69IGQpwJnuyt1MWBwqBqumRv2Ks2YrIe4TdjadXBFTjnX"
			+ "zB1ndAvFMpfu0Zf/u8xRXK57p8tl/94Y+O9IJYKLL9Cavbk4r9eVdThNVatshZxF"
			+ "LyQZWn0Au4rBMc5n7twp5oUY8ETsa2GFqpZ+xOsXADVEBAb3ytFlachJVPOsVDcK"
			+ "bTzYPvsFd8x/HyFTFStZqVud+OlUKoqn1XOyWxp5y3nvlmJQgqfnEKpo2NVqMcmz"
			+ "mKulWjAWTMN0xfVuIFXtCObubSKBxg3RyKhI1U7x6ff8fRDtnGZw4YgYQVKjjtoO"
			+ "hMwPoYaT/0JA2PTWwxAN6Xmh8KqOkCEnmqsDKVZEOY77qYZZRSN7UqDmgyUs2GBc"
			+ "zmhzD1SPleQ1wzMiaFhm9Mvo+oS6hc7p4jVMsZfNi/mIqzo5zGioidMecZEIzw0c"
			+ "Vk8ZLoBMcejY21/H6r8AaMkCAwEAAQ==";

	public static void main(String[] args) throws IOException, InvalidKeySpecException,
NoSuchProviderException {
		PublicKey publicKey = null;
		byte[] encryptedTextBytes = null;
```

```
		/**
		 * The original password to encrypt
		 */
		String originalPasswordToEncrypt = "yourPassword";

		try {

			/**
			 * Creates a new X509EncodedKeySpec with the given encoded key.
			 */
			X509EncodedKeySpec keySpec = new
X509EncodedKeySpec(Base64.getDecoder().decode(publicKeyString));

			/**
			 * Returns a KeyFactory object that converts public/private keys of
			 * the specified algorithm. This method traverses the list of
			 * registered security Providers, starting with the most preferred
			 * Provider. A new KeyFactory object encapsulating the KeyFactorySpi
			 * implementation from the first Provider that supports the
			 * specified algorithm is returned.
			 */
			KeyFactory keyFactory = KeyFactory.getInstance("RSA");

			/**
			 * Generates a public key object from the provided key specification
			 */
			publicKey = keyFactory.generatePublic(keySpec);

		} catch (InvalidKeySpecException e) {
			e.printStackTrace();
		} catch (NoSuchAlgorithmException e) {
			e.printStackTrace();
		}

		/**
		 * Calling the encrypt(...) method to get the encrypted text in byte
		 * format
		 */
		encryptedTextBytes = encrypt(publicKey, originalPasswordToEncrypt);

		/**
		 * To see the original text
		 */
		System.out.println("input:\n" + originalPasswordToEncrypt);

		/**
		 * This will be the final form of the encrypted text and it must be used
		 * for the inputs
		 */
		System.out.println(
				"encrypted text:\n" + new
String(Base64.getEncoder().encode(encryptedTextBytes), "UTF-8"));
	}

	/**
	 * @param PublicKey
	 *          generated with Keyfactory.
	 * @param TextToEnrypt
	 *          password text for encryption.
	 * @return byte[] The encrypted password in bytes
```

```
     */
    private static byte[] encrypt(Key publicKey, String textToEnrypt) {
        try {
            /**
             * Ciphers, also called encryption algorithms, are systems for
             * encrypting and decrypting data. A cipher converts the original
             * message, called plaintext, into ciphertext using a key to
             * determine how it is done. In order to create a Cipher object, the
             * application calls the Cipher's getInstance method, and passes the
             * name of the requested transformation to it The default RSA key
             * size is 1024 bit
             */
            Cipher rsa;

            /**
             * Returns a Cipher object that implements the specified
             * transformation.
             */
            rsa = Cipher.getInstance("RSA");

            /**
             * Initializes this cipher with a key
             */
            rsa.init(Cipher.ENCRYPT_MODE, publicKey);

            /**
             * Encrypts data in a single-part operation
             */
            return rsa.doFinal(textToEnrypt.getBytes());
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

Az így előállított, eltitkosított OTPdirekt Electra jelszó és felhasználó név páros kell, minden esetben az adott Corporate API hívás payloadjának **CrptAuthNfmtn** objektumába megadni. Pl.:

```
  "CrptAuthNfmtn": {
    "CrpNfmtn": {
      "CrpIdntfctnData": "ELECTRA_USER_ID",
      "CrpPsswrd": "moVDRNJNFUA93LCTkVStv99Ed2HuCeTvM569V74gc+Y6jmcz9ZfCuav6bSyLxkzXgkE/g0agncZo+NzdgA7A69PcX84C
oTsbi7bWV/dL6St9E8xspImHOnzvcr+OUNIPKR7QpLeMfE5/NUhm1ENnMCG1pHNXH2/3RFiIKDMNgC87ge6fHiSo791MoeJCEbg9I7RVULuhM6qR
bGRLp+0VpP8U4/4WQyK61vOTelbiqrbIe+WNAUimW85iwS9TD3Dx+I13UwlEeJ0nyVtJi4cANFWJnmxlIQuqIDf4ne3MWMZwiCx/KI88L0pmiyfl
Ma+cBEKWVtwOiCdhqYRAYNb1mQ=="
    }
  },
```

Azon Corporate API szolgáltatások esetében, ahol szükséges aláírói jóváhagyás (ezek alapvetően a nem lekérdező jellegű szolgáltatások, pl. megbízások), ott szintén hasonlóan kell eljárni. Tehát az aláírói jogosultsággal rendelkező OTPdirekt Electra felhasználó(k) azonosítóját, illetve a hozzájuk tartozó eltitkosított jelszót is küldeni szükséges az API payload-jának **CrpSgnDtls** listájában. Pl.:

```
  "CrptAuthNfmtn":{
     "CrpNfmtn":{
        "CrpIdntfctnData":"ELECTRA_USER_ID",
```

```
        "CrpPsswrd":"BZW3u3D8z8yEf0rO52uG+IUNQp+rvgtmBYt0lE6paSiqzlDUpTi+SV4QJAM92aOt68l9LTnScseZRvG4ox6cAWUKwL
QQO0N5NxctCZI5/WDiAWUxZ8qNs/8+ceossVFKKgvYH0gQW6+sShKftgHhVLxQAbZSEqLWbfXxk+TqyOPNB9agqATttHr4OhAv6RzMvZrdAN0RH7
yvoSi/jOlLGYOSk6H+02TVIEibw5IGISiydodL2Zap7PP7yBtEXdkpYivNH3l1xyjzxfKyzTCbexbEX6FSPa6qQaqQtSAMtFzEuleUqorIz1K5KQ
9DFwy9QD1amWraT/g2Z12a3zjXuQ=="
      },
      "CrpSgnDtls":[
         {
            "SgnrIdntfctnData":"NEBEATA",
            "SgnrIdntfctnPsswrd":"BZW3u3D8z8yEf0rO52uG+IUNQp+rvgtmBYt0lE6paSiqzlDUpTi+SV4QJAM92aOt68l9LTnScseZRv
G4ox6cAWUKwLQQO0N5NxctCZI5/WDiAWUxZ8qNs/8+ceossVFKKgvYH0gQW6+sShKftgHhVLxQAbZSEqLWbfXxk+TqyOPNB9agqATttHr4OhAv6R
zMvZrdAN0RH7yvoSi/jOlLGYOSk6H+02TVIEibw5IGISiydodL2Zap7PP7yBtEXdkpYivNH3l1xyjzxfKyzTCbexbEX6FSPa6qQaqQtSAMtFzEul
eUqorIz1K5KQ9DFwy9QD1amWraT/g2Z12a3zjXuQ=="
         }
      ]
   },
```


A **CrpSgnDtls** listában a már ismert aláíró pontszám validáció is érvényesül, azaz, ha az adott aláíró nem rendelkezik önmagában elegendő pontszámmal a jóváhagyáshoz, akkor lehetőség van arra, hogy több aláíró is küldésre kerüljön egy azon hívás során. Ezt a Corporate API Gateway ellenőrzi és a feltételek teljesülése esetében végrehajtja az adott üzleti kérést.

4.1.6 Számlainformáció (egyenleg) lekérdezés

Az API szolgáltatás meghívásával az Ügyfél megkapja a kérelemben elküldött számla információit, számla elnevezését, típusát, beleértve az egyenlegeket is.

4.1.6.1 Speciális követelmény

Az API kérelemben a Bank az alábbi mezőkben küldött értékek tartalmával kapcsolatban a következő speciális követelményeket fogalmazza meg az Ügyféllel szemben:

- **BnsDta/AcctRptReq/GrpHdr/MsgId:** Az MsgId mezőben érkező értéknek meg kell egyeznie a Cntxt/Extnl/TxId és Cntxt/Extnl/RqstId mezők értékeivel.

4.1.7 Számlatörténet lekérdezés

Az API szolgáltatás meghívásával az Ügyfél egy adott bankszámláján a kérelemben megadott időszakon belül végrehajtott pénzügyi tranzakciók listáját kapja vissza. A tranzakciók időrendbeli sorrend szerint kerülnek átadásra. A kérelemben érkező időszakot a rendszer ellenőrzi, abban az esetben, ha annak intervalluma meghaladja az 1 hónapot, akkor automatikusan elutasítja a lekérdezést. Ezek ismeretében az engedélyezett maximálisan lekérdezhető időszak 1 hónap (30 naptári nap).

4.1.7.1 Speciális követelmény

Az API kérelemben az alábbi mezőkben küldött értékek tartalmával kapcsolatban a következő speciális követelményeket fogalmazza meg az Ügyféllel szemben:

- **BnsDta/AcctRptgReq/GrpHdr/MsgId:** Az MsgId mezőben érkező értéknek meg kell egyeznie a Cntxt/Extnl/TxId és Cntxt/Extnl/RqstId mezők értékeivel.

4.1.8 Fizetési megbízások (egyedi, kötegelt)

Az API szolgáltatás meghívásával az Ügyfélnek lehetősége van egyedi és kötegelt fizetési megbízások kezdeményezésére. A szolgáltatással az Ügyfél kezdeményezhet egyedi vagy kötegelt megbízásokat:

- eseti és értéknapos belföldi forint átutalás (bankon belül és kívül),
- deviza átutalás (bankon belül és kívül),
- SEPA megbízásokat,
- VIBER megbízásokat.

A szolgáltatás az ISO20022 pain.001 alapú szabvány struktúrát követi, amely meghatározza a kereskedelmi tranzakcióhoz kapcsolódó fizetési aktiválási kérelmet.

FONTOS! Az OTP Bank Corporate API Fizetési Megbízások szolgáltatása kizárólag aszinkron módon történik. A Fizetési megbízások sikeres fogadását követően a Bank a válasz üzenetében értesíti az Ügyfelét, hogy a kérelem befogadásra került. Kérjük Ügyfeleinket, hogy ilyen esetben SOHA ne ismételjék meg Fizetési megbízásukat közvetlenül a megelőző feladást követően. Először ellenőrizzék a feladott megbízásuk állapotát a célra kiajánlott Tranzakció státusz lekérdezéssel. Abban az esetben, ha belátható időn belül a státusz lekérdezésre kapott válaszban semmilyen üzletileg elfogadható állapot nem érkezik, akkor kérjük, vegye fel a kapcsolatot Bankunkkal a CorpAPISupport@otpbank.hu email címen keresztül.

Továbbá felhívjuk Corporate Api szolgáltatást igénybe vevő Ügyfeleink figyelmét, hogy a banki háttérrendszer az, amely a tranzakció részletei és a banki szabályok alapján eldönti, hogy az adott átutalási megbízás megfelel-e annak a megbízási típusnak (AFIR, FCY, IG2, SEPA, VIBER) mellyel az a Bankhoz beérkezett, majd a kritériumok alapján dolgozza fel a tranzakciót.

4.1.8.1 Speciális követelmény

Az API kérelemben az alábbi mezőkben küldött értékek tartalmával kapcsolatban a következő speciális követelményeket fogalmazza meg az Ügyféllel szemben:

- **BnsDta/CstmrCdtTrfInitn/GrpHdr/MsgId:** Az MsgId mezőben érkező értéknek meg kell egyeznie a Cntxt/Extnl/TxId és Cntxt/Extnl/RqstId mezők értékeivel
- A fizetési megbízások megkövetelik az Aláírói azonosítást. Ezek értelmében ügyelni kell arra, hogy a beküldött kérelem a szabályoknak megfelelően legalább tartalmazzon egy önálló aláírót, vagy több együttes aláírót. (**CrpSgnDtls**).

4.1.9 Tranzakció státusz lekérdezés

Az API szolgáltatás meghívásával az Ügyfél képes az eredeti tranzakció azonosítóval a korábban feladott egyedi vagy kötegelt fizetési megbízásainak állapotát lekérdezni. A kérelem beküldésekor az Ügyfélnek ügyelnie kell arra, hogy vagy a tranzakció eredeti azonosítóját, vagy a tranzakció befogadását visszaigazoló üzenet egyedi banki azonosítóját adja meg. A szolgáltatás válaszában az Ügyfél a tranzakció állapotától függetlenül információt kap a tranzakció tétel(ek) összegéről, devizaneméről, terhelendő és jóváírandó számlaszámokról.

4.1.9.1 Speciális követelmény

Az API kérelemben az alábbi mezőkben küldött értékek tartalmával kapcsolatban a következő speciális követelményeket fogalmazza meg az Ügyféllel szemben:

- **BnsDta/CstmrPmtStsReq/GrpHdr/MsgId:** Az MsgId mezőben érkező értéknek meg kell egyeznie a Cntxt/Extnl/TxId és Cntxt/Extnl/RqstId mezők értékeivel.

4.1.9.2 Egyéb Speciális Információk

A Bank Corporate API Tranzakció státusz lekérdezési szolgáltatásának válaszában a tételekre vonatkozó státusz kódok mellett, indoklási kódot (Reason – Rsn) is visszaad Ügyfelei számára. A Bank az indoklási kódok magyarázatára egyedi forgató táblát készített, mely külön dokumentumban (OTPBank_CorpAPI_RsnCd_Mapping_v1.2.xlsx) került definiálásra.

4.1.10 Adatformátumú, vagy hitelesített PDF számlakivonat lekérdezése

A Corporate API szolgáltatás meghívásával az Ügyfél képes akár adatformátumú (ISO20022 szabvány szerinti), akár hitelesített PDF számlakivonatok lekérdezésére.

4.1.10.1 Speciális követelmény

Az API kérelemben az alábbi mezőkben küldött értékek tartalmával kapcsolatban a következő speciális követelményeket fogalmazza meg az Ügyféllel szemben:

- **BnsDta/AcctRptgReq/GrpHdr/MsgId:** A MsgId mezőben érkező értéknek meg kell egyeznie a Cntxt/Extnl/TxId és Cntxt/Extnl/RqstId mezők értékeivel.

4.1.10.2 Egyéb Speciális Információk

A PDF számlakivonat lekérdezések esetében a Bank a jelenlegi folyamatok szerint a vonatkozó hitelesített PDF dokumentumot a API válasz üzenet payload részben base64 encode-olással adja meg az Ügyfél számára. A base64-el elkódolt string értéknek a hitelesített PDF formátumra történő decode-olását Ügyfél oldalon szükséges kezelni.

4.1.11 Csoportos fizetési megbízások

Az API szolgáltatás meghívásával az Ügyfélnek lehetősége van a UGIRO csoportos üzenetszabványnak megfelelő csoportos átutalási-, csoportos beszedési megbízások, valamint csoportos beszedési felhatalmazások visszaigazolásának kezdeményezésére.

A szolgáltatás válaszában az Ügyfél az adott tranzakció állapotára, valamint a bankon belüli és kívüli tételek összesített értékére vonatkozó információt kap.

A szolgáltatás keretrendszere a GIRO utasítása alapján a magyarországi csoportos tranzakciókra vonatkozó egyedi UGIRO üzenetszabványokat követi.

4.1.11.1 Speciális követelmény

Az API kérelemben az alábbi mezőkben küldött értékek tartalmával kapcsolatban a következő speciális követelményeket fogalmazza meg az Ügyféllel szemben:

- **BnsDta/GrpTrnsctnInitReq/GrpHdr/MsgId:** Az MsgId mezőben érkező értéknek meg kell egyeznie a Cntxt/Extnl/TxId és Cntxt/Extnl/RqstId mezők értékeivel.

4.1.11.2 Egyéb Speciális Információk

Az Ügyfélnek az API hívás üzleti részében található **GrpTrnsctnDtls** mezőben kell átadnia a ***tömörített*** és ***Base64***-el elkódolt – a választott megbízási típus UGIRO szabvány formai követelményeknek megfelelő – csoportos állományt.

Az állomány megfelelő tömörítésre és annak Base64-el történő elkódolásra az alábbi Java programozási nyelveben (Java build verzió: 1.8) megadott példa szolgálhat alapul:

```
package CompressOrDecompress;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintStream;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Base64;
import java.util.zip.DeflaterInputStream;
import java.util.zip.DeflaterOutputStream;
import java.util.zip.InflaterInputStream;

public class CompressOrDecompress {

    public static void main(String[] args) throws IOException {

        /**
         * The given arguments must be the following:
         * 1st is the file name
         * 2nd can be D if decode OR E Encode
         * Note that if D is selected ISO_8859_1 must be the 3. arg
         * 3rd is the encoding
         * For example: java -jar Compress_Decompress_Ugiro_data.jar encode.txt E ISO_8859_1
         */
        if (args.length < 3) {
            System.out.println("Invalid input data");
        }

        String inputFile = args[0];
        String operation = args[1];
        String encoding = args[2];

        FileInputStream fis = new FileInputStream(inputFile);
        InputStreamReader ins = new InputStreamReader(fis,encoding);
        BufferedReader inReader = new BufferedReader(ins);
        String line;
        StringBuilder inputData = new StringBuilder();
        ArrayList<String> inputList = new ArrayList<>();

        /**
         * Reading in the input file
         */
        while ((line = inReader.readLine()) != null) {
            inputList.add(line);
        }
        for (int i=0; i<inputList.size(); ++i)
            if (i < inputList.size() - 1) {
                inputData.append(inputList.get(i)).append("\r\n");
```

```
            } else {
                inputData.append(inputList.get(i));
            }

        /**
         * Encoding the data and making generate the output file
         */
        if ("E".equals(operation)) {
            String codeddata = codeByteArrayToString(inputData.toString().getBytes());
            try {
                codeddata = zipString(inputData.toString(),encoding);
            } catch (Exception e1) {

                e1.printStackTrace();
            }

            File outansfile = new File(inputFile + "_output.txt");
            FileOutputStream fosans = null;
            try {
                fosans = new FileOutputStream(outansfile);
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }
            PrintStream out3ans = new PrintStream(fosans != null ? fosans : null,true);
            out3ans.print(codeddata);
            out3ans.close();

            /**
             * Decoding the data and making generate the output file
             */
        } else if ("D".equals(operation)) {
            byte[] toDecoode = Base64.getDecoder().decode(inputData.toString());
            String decodedData = decodeByteArrayToString(toDecoode,encoding);
            File outansfile = new File(inputFile + "_output.txt");

            BufferedWriter writer = null;

            try {
                writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(outansfile, true),
StandardCharsets.ISO_8859_1));
            } catch (FileNotFoundException ex) {
                ex.printStackTrace();
            }
            if (writer != null) {
                writer.write(decodedData);
                writer.newLine();
                writer.close();
            }

        }else {
            System.out.println("Wrong argument. Second argument must be D or E");
        }
        fis.close();
    }

    /**
     * DeflaterOutputStream implements an output stream filter for compressing data in the "deflate" compression
format
     */
    private static String zipString(String sb, String encoding) throws Exception {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
```

```
        DeflaterOutputStream dos = new DeflaterOutputStream(baos);
        String res = null;

        if (sb.length() > 0) {

            /**
             * Writes an array of bytes to the compressed output stream
             * Then closes the underlying stream
             */
            try {
                dos.write(sb.getBytes(encoding));
                dos.close();
                /**
                 * Encodes all bytes from the specified byte array into a newly-allocated byte array using the Base64 encoding scheme
                 */
                res = new String(Base64.getEncoder().encode(baos.toByteArray()));
            } finally {
                try {
                    dos.close();
                    baos.close();
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
        }
        return res;
    }

    /**
     * InflaterInputStream implements a stream filter for uncompressing data in the "deflate" compression format
     */
    private static String decodeByteArrayToString(final byte[] bytes, String encodeing) {
        final ByteArrayInputStream bais = new ByteArrayInputStream(bytes);
        final ByteArrayOutputStream baos = new ByteArrayOutputStream();
        final byte[] buf = new byte[bytes.length];
        try (InflaterInputStream iis = new InflaterInputStream(bais)) {
            int count = iis.read(buf);
            /**
             * Writes an array of bytes to the compressed output stream where the count is the number of bytes to write
             */
            while (count != -1) {
                baos.write(buf, 0, count);
                count = iis.read(buf);
            }
            return baos.toString(encodeing);
        } catch (final Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    /**
     * DeflaterOutputStream implements an output stream filter for compressing data in the "deflate" compression format
     */
    private static String codeByteArrayToString(final byte[] bytes) {
        final ByteArrayInputStream bais = new ByteArrayInputStream(bytes);
        final ByteArrayOutputStream baos = new ByteArrayOutputStream();
        final byte[] buf = new byte[bytes.length];
```

```
        try (DeflaterInputStream iis = new DeflaterInputStream(bais)) {
            int count = iis.read(buf);
            /**
             * Writes an array of bytes to the compressed output stream where the count is the number of bytes to
write
             */
            while (count != -1) {
                baos.write(buf, 0, count);
                count = iis.read(buf);
            }
            return new String(baos.toByteArray(), StandardCharsets.ISO_8859_1);
        } catch (final Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

4.1.12 Értesítő lista lekérdezése

Az API szolgáltatás meghívásával az Ügyfél képes az általa beküldött csoportos megbízásokhoz tartozó értesítő állományok, megadott időintervallumon belüli listázására.

A szolgáltatás válaszában keletkezett lista tételesen tartalmazza a csoportos megbízásokhoz tartozó állományokat, azok azonosítóival együtt.

A hívással az alábbi típusú értesítők listája érhető el:

- CS-DETSTA-N,
- RECEIPT,
- CS-STATUS,
- FELHNA,
- FELHKI,
- CS-DETSTA-V

4.1.12.1 Speciális követelmény

Az API hívásban az alábbi mezőkben küldött értékek tartalmával kapcsolatos speciális követelmények:

- **BnsDta/GrpTrnsctnListReq/GrpHdr/MsgId:** Az MsgId mezőben érkező értéknek meg kell egyeznie a Cntxt/Extnl/TxId és Cntxt/Extnl/RqstId mezők értékeivel.

4.1.13 Csoportos tranzakciós analitika lekérdezés

Az API szolgáltatás meghívásával az Ügyfél képes a Bank által generált kommunikációs azonosítóval a korábban feladott csoportos megbízásaihoz (átutalás, beszedés, csoportos beszedési felhatalmazások visszaigazolása) kapcsolódó állományok részletes adatait lekérdezni.

FONTOS, hogy kérelem beküldésekor az Ügyfél a Bank által generált kommunikációs azonosítót adja meg.

4.1.12.1 Speciális követelmény

Az API hívásban az alábbi mezőkben küldött értékek tartalmával kapcsolatos speciális követelmények:

- **BnsDta/GrpTrnsctnQryReq/GrpHdr/MsgId:** A MsgId mezőben érkező értéknek meg kell egyeznie a Cntxt/Extnl/TxId és Cntxt/Extnl/RqstId mezők értékeivel.

4.1.13.1 Egyéb Speciális Információk

A Bank a válaszban az alábbi UGIRO üzenetszabvány szerinti állományokat biztosítja:

- CS-DETSTA-N,
- CS-STATUS,
- FELHNA,
- FELHKI,
- CS-DETSTA-V

Ezek ismeretében, ha a lekérdezett állomány, CS-DETSTA-N, CS-DETSTA-V, FELHKI, FELHNA, vagy CS-STATUS, akkor a Body válaszában található **BkToGrpTrnsctnRpt/OrgnlGrpTrnsctnInf/GrpTrnsctnMsg** mezőben tömörítve és base64 kódolva küldi vissza a lekérdezett típus állományát. Ezen mező érték visszakódolása és kitömörítése a 4.1.11.2 pontban

leírtak szerint az Ügyfél alkalmazás rétegében kell megtörténnie. Abban az esetben, amikor a lekérdezett állomány RECEIPT, azaz átadás-átvételi bizonylat, akkor a Bank a válaszában nem a tömörített és base64-el elkódolt xml fájlt adja vissza, hanem az xml fájl teljes struktúráját JSON formátumban. Tovább információkat ezzel kapcsolatban a vonatkozó ***OTP_Bank_CorporateAPI_swagger_configmap_CorporateToBank_v1.7.0.yaml*** fájl tartalmaz.

Az átadás-átvételi bizonylat (RECEIPT) a csoportos megbízások Bank általi befogadásával kapcsolatos státuszüzenet. Ez az üzenet a csoportos megbízás állományban lévő esetleges hibák jelzésére szolgál, a megbízások tételei bankon belüli- kívüli bontásban szétválogatásra kerülnek, valamint a csoportos megbízások esetében a szükséges fedezetbiztosításhoz tartalmazza a technikai számlaszámokat és az azokra átvezetni szükséges összegeket.

4.2 Bank – Ügyfél irány

A szolgáltatás értesítők, csak azon bankszámlákhoz tartozó eseményekről küld értesítőt, melyet az ügyfél a kiegészítő megállapodásban megjelölt.

Vonatkozó yaml fájl:

OTP_Bank_CorporateAPI_swagger_configmap_BankToCorporate_v2.0.2.yaml

4.2.1 Könyvelési státusz értesítés

A Bank Corporate API kiajánlott Fizetési státusz értesítő szolgáltatása során automatikus értesítési üzeneteket küld az Ügyfél számláin történt előkönyvelési (Pending - PDNG) és könyvelési (Booking – BOOK) eseményekről.

Az értesítők esetében az **előkönyvelési esemény** érdemben teljesen megegyezik a Könyvelési eseménnyel. Ezen típusú üzenetek csak és kizárólag az Azonnali Fizetési Rendszeren keresztül érkezett, vagy küldött tranzakciók esetében kerülnek továbbításra a Bank részéről.

Fizetési státusz értesítő szolgáltatás igénybevételéhez elengedhetetlen az Ügyfél oldali alkalmazás fejlesztése, mely képes arra, hogy ezeket az értesítő üzeneteket fogadja és feldolgozza. A biztonságos kapcsolat kiépítésre vonatkozó technikai és üzleti követelményeket jelen dokumentum 3.2 pontja alatt került részletezésre.

4.2.2 Tranzakció státusz értesítés (kötegelt állományok esetében)

A Bank a Corporate API-n kiajánlott Tranzakció státusz értesítő szolgáltatása során automatikus értesítési üzeneteket küld az Ügyfél irányába az Ügyfél azon 2-20 tételből álló kötegelt fizetési megbízásaira, melyeket a Bank Corporate API szolgáltatáson keresztül kezdeményezett. Az automatikus státusz értesítő szolgáltatás köztes (pl.: feldolgozásra vár), valamint végstátuszok (pl.: sikeresen feldolgozott, elutasított) esetében küld értesítő üzenetet az Ügyfelek számára.

A tranzakció státusz értesítő üzenet olyan tartalommal rendelkezik, amely az Ügyfél által könnyen összeegyeztethető az automatikus könyvelési státusz értesítővel (például az AcctSvcrRef mezőben érkező érték által). Az Ügyfél irányába küldött adatok köre megegyezik a tranzakció státusz lekérdezés adatkörével.

4.2.3 Kártya foglalási értesítés

A Bank a Corporate API szolgáltatásán keresztül automatikus kártya foglalási esemény értesítőket küld az Ügyfél számára a bankszámláihoz kapcsolódó, bankkártyájával végzett műveletek nyomán.

A kártya foglalási értesítő üzenet olyan tartalommal rendelkezik, amely az Ügyfél által könnyen össze egyeztethető a kártya tranzakció elszámolását követően kiküldött automatikus könyvelési státusz értesítővel.

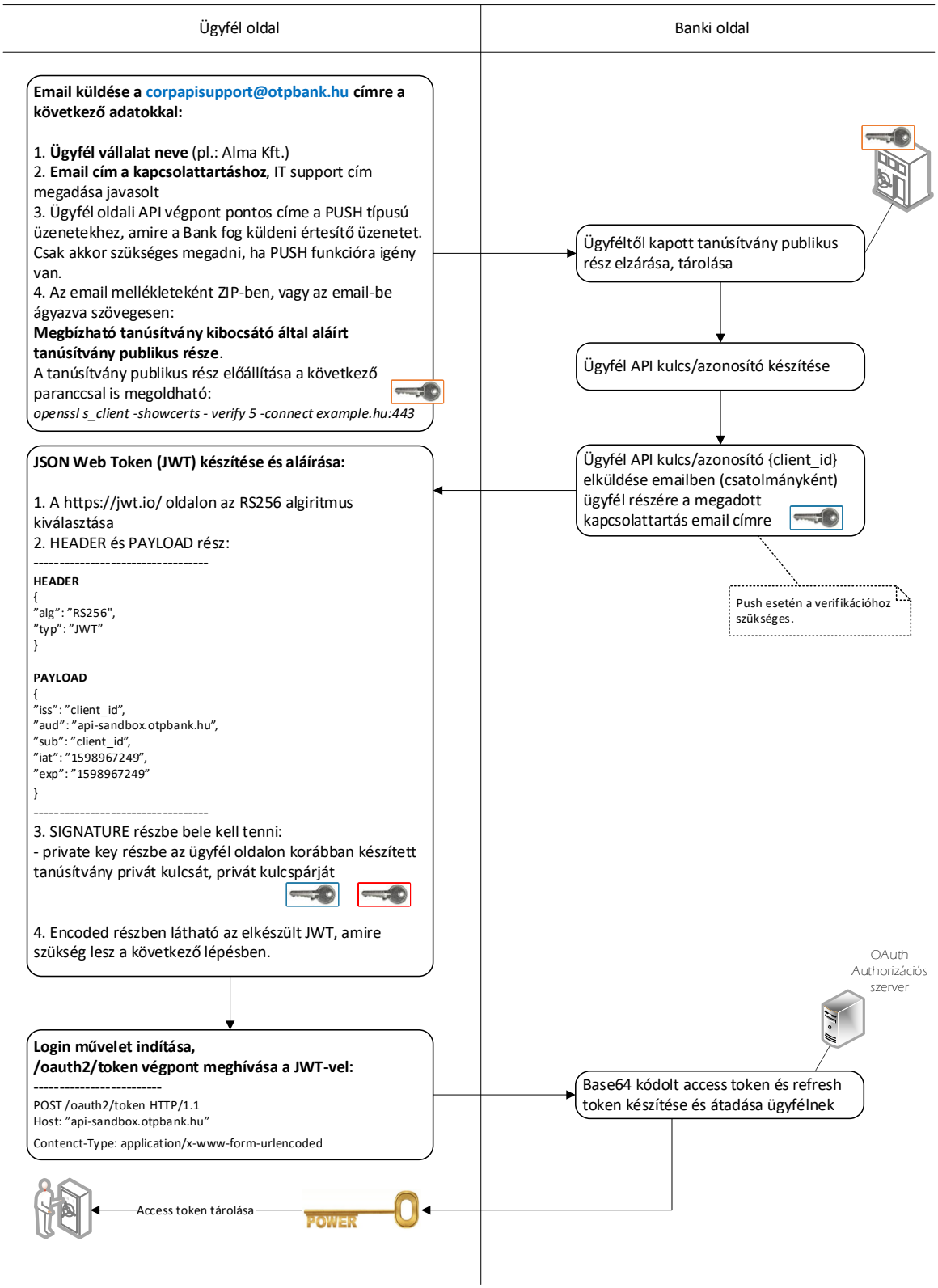
5 Corporate API swagger leíróban bekövetkezett változtatások

Sorszám	Érintett szolgáltatás	Típus	YAML-ben érintett sor(ok)	Leírás
1.	Új szolgáltatás: Group Payments	Új Tag	29	Az Új csoportos szolgáltatás tag azonosítója
2.	Account Details, Transaction Status, Account History, Account Statement Response	Módosítás	261, 1474, 2299, 2973	Response leírások pontosítása
3.	Group Payments Order service	Új API szolgáltatás	3082-3357	Csoportos Megbízások feladása Rest API szolgáltatás leíró hozzáadása.
4.	Group Payments List service	Új API szolgáltatás	3359-3623	Csoportos Megbízások listázása Rest API szolgáltatás leíró hozzáadása.
11.	Új szolgáltatás: Group Payments	Új Tag	29	Az Új csoportos szolgáltatás tag azonosítója

6 MELLÉKLETEK

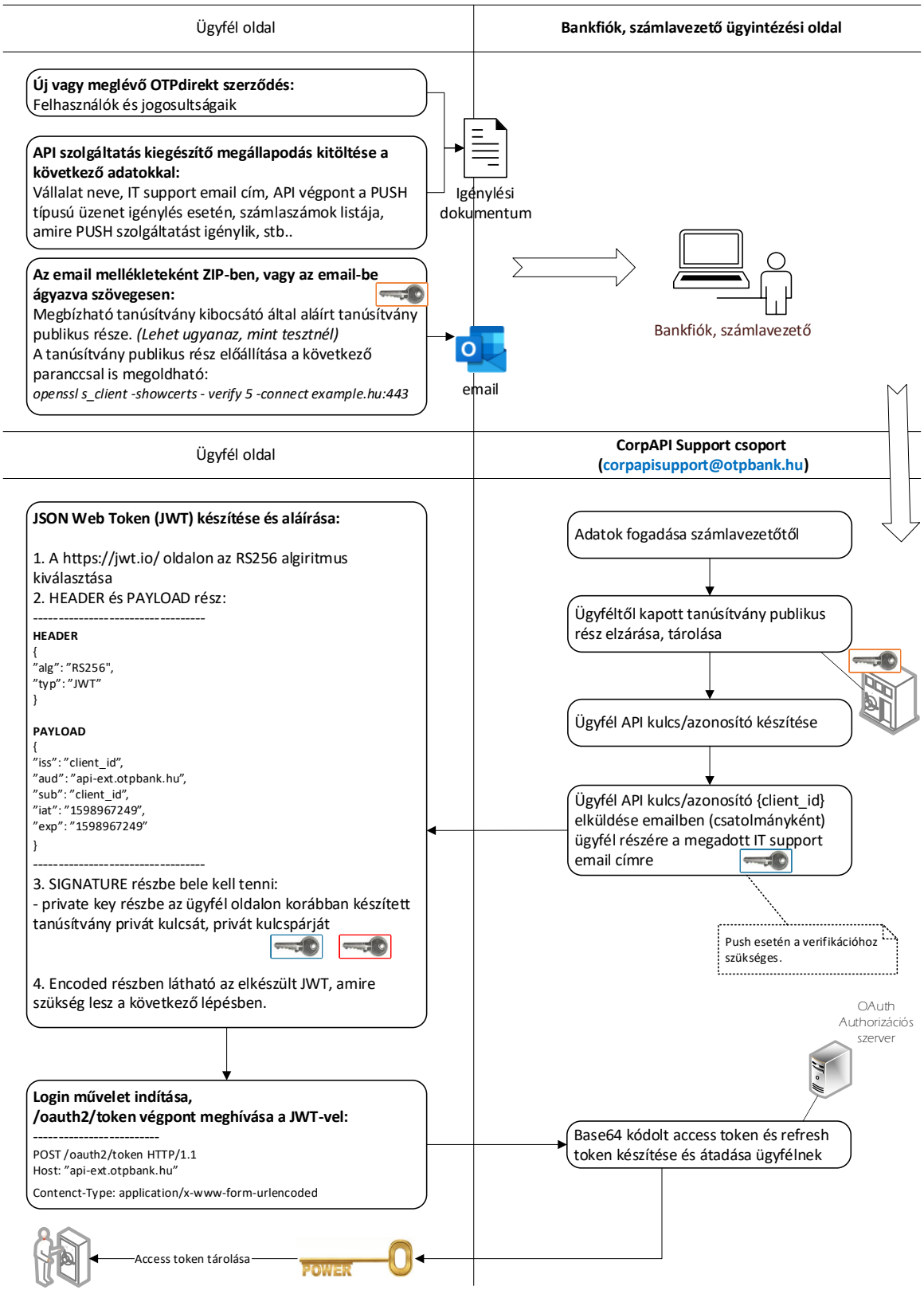
6.1 Corporate API regisztrációs folyamata – Sandbox

(1) Mockup teszteléshez hozzáférés



6.2 Corporate API regisztrációs folyamata – Éles

(2) Éles rendszerhez hozzáférés



6.3 Megbízás beküldése mintafolyamat