Csatlakozás a KKSZB rendszerhez

Publikus – IdomSoft Zrt.

Verzió v2.0.0, 2023-06-12

Tartalomjegyzék

Dokumentum kontroll

Dokumentum jellemzők

Projekt hivatalos neve	Közigazgatási szakrendszerek egységes eléréséhez és interoperabilitásához központi alkalmazás szintű szolgáltatások biztosítása
Projekt rövid neve	KAK SW
Projektazonosító	KÖFOP-1.0.0-VEKOP-15-2016-00025
Dokumentum címe	Csatlakozás a KKSZB rendszerhez
Dokumentum azonosítója	kkszb-docs-$Id:$
Verziószám	v2.0.0
Állapot	Átadott
Kiadás kelte	2023-06-12
Utolsó mentés kelte	2023-06-12
Készítette	Juhász Erzsébet, Pató István
Fájlnév	kkszb_csatlakozas-v2.0.0.pdf
Dokumentum célja	A KKSZB rendszerhez csatlakozás fejlesztési követelményeit definiálja, fejlesztőknek szól.

Változások jegyzéke

Verzió	Dátum	Változtatás rövid leírása
v2.0.0	2023.06.12	Mongo adatbázisra áttérés (RFNY, SZK, Notifier), Statisztika specifikáció
v1.7.0	2023.02.01	Dokumentum frissítése
v1.6.1	2021.02.04	Echo szolgáltatás leírás kiegészítése INT (integrációs teszt) URL-el
v1.6.0	2020.09.03	Bevezetésre kerül a KKSZB rate limit funkció KKSZB rate limit
v1.5.0	2019.06.22	Invalid HTTP kérések esetén a HTTP 400-as válasz bővítése x-kk-gw-status-message fejléccel: HTTP státusz kódok függelék
v1.4.1	2018.12.18	Aszinkron Szolgáltatás 10 MB-os mérethatárának magyarázata: valós üzenet méret nagyobb lehet mint a küldendő üzenet mérete

Verzió	Dátum	Változtatás rövid leírása
v1.4.0	2018.08.15	Bevezetésre került az x-kk-tr-id azonosító, amelyet a KKSZB generál a kérés beérkezésekor és a Szolgáltatásnak továbbadja, valamint a válaszban a Kliens alkalmazás megkapja.
v1.3.6	2018.05.18	A 3.8 Szolgáltatás tesztelése fejezetben hibás hivatkozás javítása, helyesen: *Alkalmazás szintű (szolgáltatás) teszt*
v1.3.5	2018.05.09	Az ASZ szolgáltatás interfész tervének hibajavításai, ASZ válasz függelékben messageId felvétele.
v1.3.4	2018.02.26	Az ASZ szolgáltatás interfész terve végleges.
v1.3.3	2018.01.08	Teszt curl parancs az echo szolgáltatáshoz: Echo szolgáltatás
v1.3.2	2017.12.07	A Fejlesztési tanácsok fejezet bővítése és curl szintaxis javítása.
v1.3.1	2017.10.20	A szolgáltatást elérő HTTP fejlécek hosszainak megadása a KKSZB HTTP kérés fejlécek fejezet táblázatában.
v1.3.0	2017.10.17	x-kk-security-class és x-kk-legal-basis-code bevezetése, valamint új Szolgáltatás tesztelése fejezet.
v1.2.2	2017.10.05	KKSZB ASZ interfészének pontosítása egyeztetések után, válasz kódok módosítása (200 és 202).
v1.2.1	2017.09.25	KKSZB ASZ fejezet áttördelése, apróbb hibajavítások
v1.2.0	2017.08.26	KKSZB Aszinkron Szolgáltatás: tranzakció kezelés leírása.
v1.1.0	2017.08.08	KKSZB Aszinkron Szolgáltatás interfész leírása.
v1.0.0	2017.04.25	Tanácsokkal kiegészített verzió a pilot tapasztalatai alapján.
v0.0.9	2017.01.05	Pilot csatlakozáshoz dokumentum véglegesítése.
v0.0.8	2016.11.05	Egyeztetések utáni, tartalmi véglegesítés.
v0.0.7	2016.10.15	Egyeztetésekhez előkészített verzió.
v0.0.6	2016.08.25	Első munkaverzió, szakmai egyeztetések céljából.

Kapcsolódó dokumentumok

Dokumentum neve	Kapcsolat tartalma - helye
KKSZB magas szintű szoftverarchitektúra	Jelen dokumentum előtt ajánlott ezt elolvasni, több hivatkozás történik rá.

Dokumentum használata

A dokumentum verziószámmal és készítési dátummal rendelkezik, kísérje figyelemmel a verziók változását.

A 0.x formájú verziók **munkaverziót** jelentenek, így azok kidolgozás alatt állnak, csak saját felelősségre használja!

Amennyiben a dokumentumban ezt a jelzést látja  úgy azon a ponton a dokumentum kidolgozás vagy egyeztetés alatt áll. Jellemzően a v0.x verziójú munkaanyagokban talál ilyet.

A kék színben megjelenő, aláhúzott szavak referenciák a dokumentumon belül.

A dokumentáció tartalmaz Definíciókat, Hivatkozásokat és Függelékeket. A dokumentumban szereplő fogalmak jobb megértéséhez a Definíciókban talál rövid fogalom magyarázatot. A Hivatkozások és Függelékek bővebb információt adnak egy-egy fogalomról.

Használt jelzések:

Megjegyzés, további információ.

Tanács, tipp.

Fontos figyelmeztetés.

A dokumentumban használt angol szavak a megértést segítik. Ott használjuk, ahol a magyar változat félreérthető lenne, vagy nem honosult meg annak használata a szakmai nyelvben.

A dokumentumban az alábbi értelemben használunk bizonyos szavakat:

kell, kötelező: az adott mondatban szereplő feltételek, elvárások teljesítése mindenképpen szükséges a KKSZB használatához, üzemszerű működéséhez.

ajánlott, javasolt: az adott mondatban szereplő feltételek, elvárások teljesítése nem szükséges a KKSZB használatához, üzemszerű működéséhez, de nagyban megkönnyíti, vagy egyszerűsíti az egyéb folyamatokat, vagy elvégzendő feladatokat (pl.: hibakeresés).

opcionális, igény szerint: az adott mondatban szereplő feltételek, elvárások teljesítése a felhasználó saját döntésétől függ.

Tárgy

Jelen dokumentum a KKSZB rendszerhez csatlakozás fejlesztési követelményeit definiálja kliens alkalmazás és szolgáltatás esetén. A dokumentum az alkalmazás fejlesztőknek szól.

A *KKSZB magas szintű szoftverarchitektúra* című dokumentum alapján az architekturális tervezés is elvégezhető a kapcsolódó fél oldalán.

Jelen dokumentum elolvasása előtt a KKSZB magas szintű szoftverarchitektúra dokumentum áttekintése mindenképpen szükséges!

Bevezető

A Közigazgatás- és Közszolgáltatás-fejlesztés Operatív Program 2015. évre szóló éves fejlesztési keretének megállapításáról szóló 1004/2016. (I. 18.) Korm. határozatban kiemelt projektként került nevesítésre a "Közigazgatási szakrendszerek egységes eléréséhez és interoperabilitásához központi alkalmazás szintű szolgáltatások biztosítása" című projekt, mely projekt keretein belül valósul meg a Központi Kormányzati Szolgáltatás Busz (továbbiakban: KKSZB).

Ennek célja, hogy a nemzeti adatvagyon részét képező nagy állami nyilvántartások és szakrendszerek a KKSZB keretrendszeren keresztül történő, szolgáltatás orientált szabványos összekapcsolását biztosítsa, az interoperabilitás megteremtésével a rendszerek közötti hatékony kommunikáció egységes szintre emelésével. A KKSZB segítségével lehetőség nyílik a jelenleg különböző technológiai, integráltsági, üzemeltetési szinten működő információs rendszerek egymással való összekapcsolására, a redundáns adattárolás visszaszorítására, és az ebből eredő adatintegritási hibák csökkentésére.

A KKSZB csatlakozáshoz a csatlakoztatandó alkalmazást fel kell készíteni a KKSZB igényeinek megfelelően. A KKSZB tervezésekor szempont volt, hogy a csatlakoztatandó alkalmazások módosítási igénye a lehető legkisebb legyen, ezért a KKSZB alaptechnológiaként a HTTP protokollt használja.

A kliens alkalmazás és szolgáltatás csatlakozáshoz számos követelménynek kell megfelelni:

- jogi, igazgatásszervezési, biztonsági és szerződéses feltételek
- KKSZB szoftveres csatlakozás követelményei: ennek leírását ez a dokumentum tartalmazza.
- infrastrukturális és informatikai biztonság követelményei
- minőségi követelmények teljesítése
- Rendszer Felhatalmazási Nyilvántartás (RFNY) kezelése

A szolgáltatások csatlakozásának további követelménye:

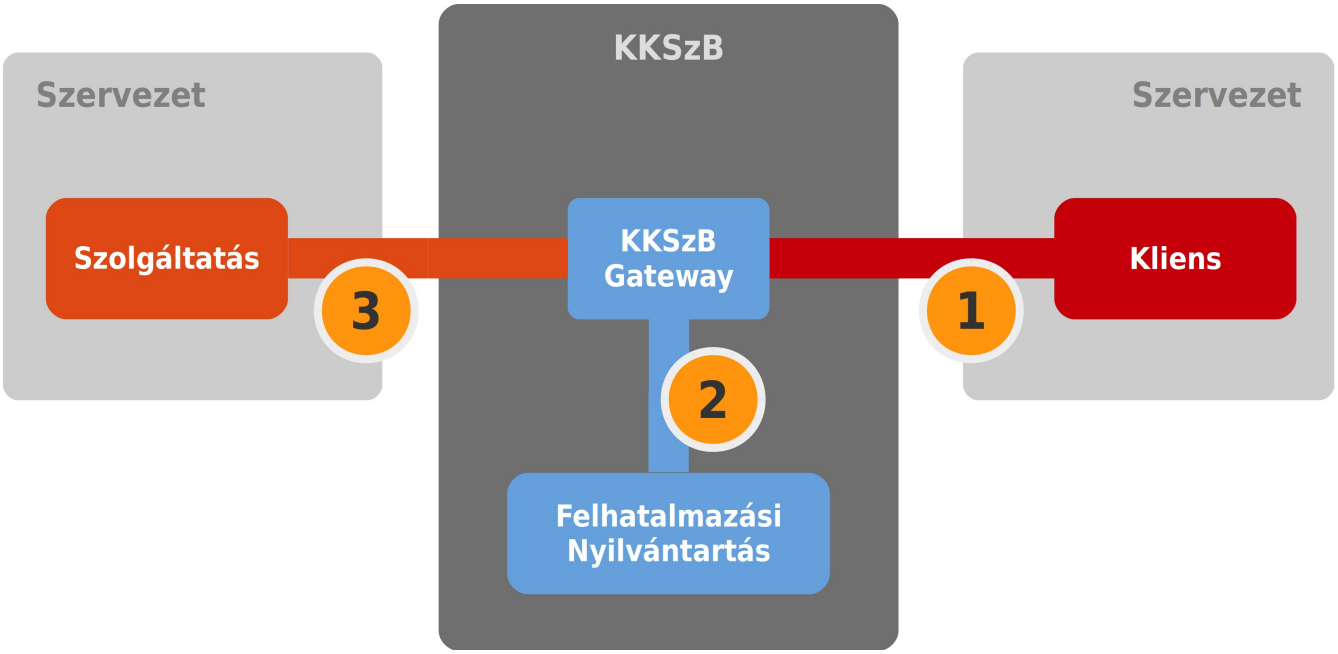
- a Szolgáltatás Katalógus (SZK) kezelése a szolgáltatásra vonatkozóan: külön dokumentum tartalmazza.

A KKSZB nyújt olyan szolgáltatásokat, amelyek igénybevétele a szolgáltatás és kliens alkalmazás igényeitől függ, ilyen az aszinkron üzenetek kezelésének támogatása. Ezen szolgáltatás igénybevételéhez az aszinkron kommunikáció szoftverfejlesztési követelményeit teljesíteni kell.

Jelen dokumentum a csatlakozás fejlesztési követelményekről szól. Definiálja a kliens alkalmazás és a szolgáltatás fejlesztőknek azokat az elvárásokat, amelyeket a KKSZB rendszerhez történő csatlakozáskor teljesíteni kell, valamint bemutatja az opcionálisan felhasználható lehetőségeket is.

Chapter 1. A KKSZB rendszer áttekintése

Az alábbi ábrán látható, a kliens alkalmazás és szolgáltatás elhelyezkedése a KKSZB rendszerben. A kliens alkalmazás a KKSZB gatewayhez csatlakozik, azon keresztül éri el a szolgáltatást. A kliens alkalmazás által küldött kérés `1` autentikálása a KKSZB gatewayen történik meg, a Rendszer Felhatalmazási Nyilvántartás (RFNY) felé küldött `2` üzenettel. Amennyiben az megfelelő, úgy ellenőrzésre kerül a szolgáltatás elérési jogosultság, amelynek sikeressége esetén a szolgáltatás elérhetővé válik `3`.



Ábra 1. KKSZB kliens alkalmazás és szolgáltatás elhelyezkedése a KKSZB rendszerben

A fenti esetben a kliens alkalmazás és a szolgáltatás is implementálja a KKSZB csatlakozási követelményeinek megfelelő működést.

A fogalmak értelmezéséhez a Definíciók fejezetet tekintse át.

1.1. Igénybe vehető szolgáltatások

A Szolgáltatás Katalógus publikusan elérhető oldalán ismerheti meg a KKSZB rendszeren igénybe vehető szolgáltatások listáját, interfész leírását és egyéb kiegészítő információkat.

A KKSZB rendszer biztosít saját szolgáltatást is a szinkron kommunikáció kezelésén felül:

- aszinkron üzenet kezelő KKSZB szolgáltatás

Ennek igénybevétele opcionális, a kliens alkalmazás és szolgáltatás közti interfész implementációjától függ.

1.2. HTTP-n utazó adatok és adatszerkezetek

A KKSZB nem módosítja, nem ellenőrzi és nem köti meg, hogy a HTTP üzenetben mi közlekedjen, csak a KKSZB specifikus HTTP fejlécet igényli. Ennek megfelelően minden jelenlegi SOAP/XML,

SOAP/JSON, SOAP with Attachment, SOAP MTOM, XML, JSON, text, bináris vagy egyéb adatszerkezet közlekedhet rajta.

1.3. HTTP-n utazó adatok mérete

A KKSZB-n keresztül a szolgáltatás felé küldött adatok méretére megkötés nincs, az kizárólag a kliens alkalmazás és szolgáltatás megállapodásától függ.

1.4. Terhelés

A KKSZB akár százezer kérést is átengedhet másodpercenként, így a terhelhetőség gyakorlatilag a szolgáltatás képességétől függ.

1.5. Timeout kezelése

A KKSZB egy kérésben legfeljebb 2 percig várakozik a válasz elkezdésére, utána terminálja a kapcsolatot. Ez a felső korlátja a timeoutnak, a kliens alkalmazás ettől rövidebb időt is megszabhat a kívánt szolgáltatás igényeinek figyelembe vételével.

1.6. Hibakezelés és válaszkódok

HTTP válasz státusz kódok részletes leírását a függelék tartalmazza.

1.7. Titkosítás és letagadhatatlanság

A KKSZB a kliens alkalmazás - szolgáltatás közötti üzenetek tekintetében semleges, azokat nem olvassa, nem tárolja, nem dolgozza fel és csak a címzett szolgáltatás felé továbbítja azt. A KKSZB ugyanakkor a HTTP fejlécben utazó adatok közül a saját **x-kk** kezdetű, valamint az elérésre vonatkozó adatokat, úgymint: *x-forwarded-proto, x-forwarded-host, date és státusz kód* adatokat logolhatja a szolgáltatás minőségének fenntartásának céljából.

A KKSZB nem tárol és nem dolgoz fel semmilyen azonosításra vonatkozó HTTP fejlécet, ilyen például: Authorization

Mivel a KKSZB tekintetében az üzenet transzparens, ezért az tetszőleges titkosítással közlekedhet a KKSZB-n keresztül.

A KKSZB nem biztosít közvetlenül szolgáltatást, hanem csak közvetítheti azokat a titkosítás és időpecsételés esetén. Így a kliens alkalmazás és szolgáltatás felelőssége ennek használata vagy mellőzése.

A KKSZB rendszer a kommunikációt HTTPS (TLS) használatával titkosítja. Az üzenetek egyedi titkosítása (pont-pont titkosítás) továbbra is a feladó kliens alkalmazás és a szolgáltatás joga vagy kötelessége.

1.8. Fejlesztési tanácsok

Az alábbiakban olyan fejlesztési tanácsokat sorolunk fel, amely segítheti a kliens alkalmazás felkészítését a KKSZB-hez történő csatlakozáshoz. Ezek nem kötelező érvényűek, ugyanakkor segítenek a szoftver minőségének emelésében.

- Mielőtt elkezdi a fejlesztést figyelmesen olvassa át a kapcsolódó dokumentumokat: jelen dokumentumot és a [kkszb-magas-szintu-szoftverarchitektura] dokumentumot.
- A fejlesztett programkódban a kommenteknél jelezze, hogy melyik verziójú dokumentumokból dolgozott.
- Ajánlott a HTTP protokoll esetén a státusz kódok, a HTTP fejléc leírásával megismerkedni.
- A küldött kéréshez tartozó **x-request-id** értékét a logban és a napló adatbázisban is szerepeltesse, a **timestamp** és a megszólított rendszer azonosítójával együtt.

Ha kliensként kíván szolgáltatást igénybe venni, akkor:

- A kommunikációhoz szükséges [kliens autentikációs token]-ot alkalmazás konfigurációba helyezze, mert a teszteléshez, valamint az éles környezetben történő használathoz más-más tokeneket kell használni.
- **Az éles [kliens autentikációs token]-ot soha se tárolja publikusan elérhető helyen, pl.: verziókövető rendszer, hibakezelő rendszer, dokumentációk.**
- Mielőtt elkezdi a tesztelést bizonyosodjon meg arról, hogy a teszt szerveréről eléri a KKSZB teszt környezetét: ezt egy egyszerű curl hívással teheti meg. **Ne az alkalmazással próbálja meg az első tesztet, hogy az esetleges alkalmazás hibát és az elérési hibát egyértelműen szét tudja választani!**

```
curl -I -H "x-request-id: 00000000-1111-4317-9c0b-1c2b75421410" -H "x-kk-
authentication:<a kapott autentikációs token>" <megszólítandó szolgáltatás URL>
```

- Ellenőrizze, hogy **az interfész igényeinek megfelelő összes HTTP fejléc átadásra kerül-e,** különösen a **Content-Type** (például: text/xml), **HTTP ige** (-X kapcsoló) és a **SOAPAction vagy Action** paraméterekre ügyeljen.
- Ha saját gépéről adja ki a curl utasítást, akkor a GovCA nem biztos hogy be lett állítva mint megbízható tanúsítvány (ajánlott beállítani), ekkor a **-k** kapcsolóval tudja elfogadni a nem megbízható tanúsítványt.
- Ne feledje a HTTP kérés és válasz **timeoutokat** az alkalmazás és az interfész igényei szerint **beállítani és kezelni!**
- Győződjön meg róla, hogy a **KKSZB rendszer HTTP státusz kódjait** és az **interfész hibaüzeneteit** megfelelően kezeli.
- **Vegye fel megbízható *webserver* tanúsítványként a KKSZB rendszerhez kiadott tanúsítványt,** hogy biztos legyen abban, hogy a KKSZB rendszer felé intézi a kérését.
- Vegye figyelembe, hogy a [kliens autentikációs token] egy interfészhez és egy jogalaphoz tartozik. Amennyiben az alkalmazásból ugyanazt az interfészt szólítja meg eltérő jogalapokkal,

akkor eltérő tokeneket kell a kérésben felhasználni. A konfigurációban ennek megfelelően szerepeltesse a tokeneket.

- Ellenőrizze a tesztek közben, hogy a KKSZB rendszer hibaüzeneteit és a Szolgáltatás hibaüzeneteit meg tudja különböztetni.
- Több példányban futó alkalmazás esetén ugyanazokat a tokeneket használhatja.

Ha Szolgáltatást fejleszt:

- Használjon automatikus teszteket a szolgáltatás ellenőrzéséhez!
- Új szolgáltatás fejlesztése esetén tekintse át, hogy a KKSZB HTTP kérés fejlécek felhasználhatóak-e az új szolgáltatásban (lásd polimorf válasz).
- Lehetőleg kerülje a SOAP 1.1 használatát, mert kliens hiba esetén (validálási hiba) is *500 - Internal Server Error* HTTP státuszkódot ad vissza, amely félrevezető és az üzemeltetési feladatokat nehezíti, bővebben: https://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383529
- A kapcsolódó felek azonosítását bízza a KKSZB rendszerre, ne használjon második szintű alkalmazás azonosítást!
- **Fogadja el a KKSZB rendszer kliens tanúsítványát megbízhatónak** a szolgáltatási végponton, hogy biztos legyen abban hogy a KKSZB szólította meg.
- **Telepítse a kapott webserver tanúsítványát** és szolgáltassa a KKSZB által kezdeményezett kérés esetén (HTTPS), így a KKSZB rendszer biztos lehet abban, hogy a megfelelő szolgáltatás szólította meg!
- A szolgáltatás állapotát a KKSZB rendszeresen ellenőrzi, ehhez ne felejtsen el HTTP URL végpontot biztosítani a specifikáció szerint.
- Verziózza a szolgáltatását! Készüljön fel arra, hogy legalább két szolgáltatás verziót is kell párhuzamosan futtatnia.

Chapter 2. Szinkron kommunikáció kezelése kliens alkalmazás esetén

Az alábbi KKSZB specifikus követelményeket kötelezően teljesítenie kell a KKSZB kliens alkalmazásnak.

A KKSZB kliens alkalmazásnak:

- kezelnie **kell** a HTTP protokollt
- a kérdésben kötelezően küldenie **kell** a **x-request-id** HTTP fejlécet UUID v4 tartalommal
- a kérdésben kötelezően küldenie **kell** a **x-kk-authentication** HTTP fejlécet, amely a kliens autentikációs tokent tartalmazza. A kliens autentikációs tokent az RFNY alkalmazás webes felületén keresztül igényelheti megfelelő jogosultság birtokában.
- a kérdést a KKSZB rendszerben létező KKSZB végpont URL-re **kell** küldenie
- kezelnie **kell** a válaszban érkező HTTP státusz kódokat

Természetesen a megszólított szolgáltatás további követelményeket is támaszthat, amelyek azonban nem részei a KKSZB követelményeinek. A megszólított szolgáltatás követelménye nem állhat ellentétben a KKSZB követelményével, attól nem lehet megengedőbb és nem tehet egyéb előírást a KKSZB követelményekre vonatkozóan.

2.1. Kliens alkalmazás autentikáció

Az azonosítás minden kérésben megtörténik, ezért a kliens alkalmazásnak minden kérésben küldenie **kell** a kliens autentikációs tokent. Ebből a tokenből a KKSZB egyértelműen meg tudja állapítani, hogy ki az aki a Szolgáltatást megcímzi.

A KKSZB rendszerhez történő csatlakozás nem követeli meg a kliens alkalmazástól külön autentikációs végpont igénybevételét vagy azonosítási folyamat implementálását.

A kliens autentikációs token egy Json Web Token (JWT), amelyet az RFNY webes felületén keresztül, az arra feljogosított felhasználó válthat ki.

A kliens alkalmazásnak minden HTTP kérésben az **x-kk-authentication** HTTP fejlécben küldenie **kell** a kliens autentikációs tokent.

Hiányzó kliens autentikációs token esetén hibaüzenetet kap a kliens alkalmazás a következő státusz kóddal és üzenettel: *400, x-kk-authentication http header required*

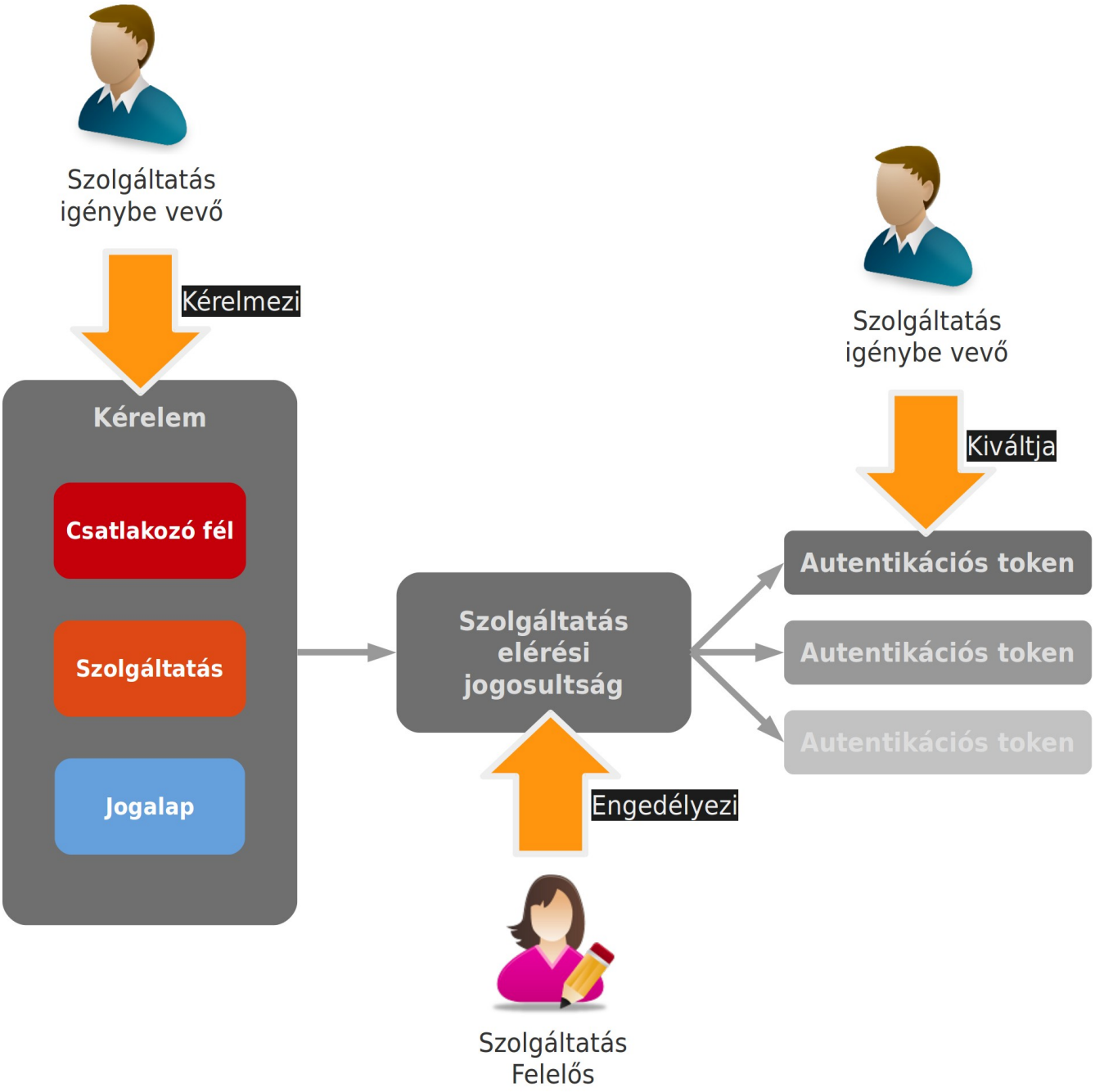
A kliens autentikációs token nem kerül ki harmadik fél részére, csak a kliens alkalmazás és a KKSZB rendszer ismeri meg.

Soha ne adja ki a kapott tokent, kezelje azt bizalmasan! Csak az adott alkalmazás példányok ismerhetik azt és csak azok használhatják.

További információ Kliens autentikációs token fejezetben található.

2.2. A kliens autentikációs token használata

A szolgáltatás igénybe vevője szolgáltatás elérési jogosultság kérelmet nyújt be az RFNY webes felületén, amelyet a szolgáltatásért felelős személy engedélyez. A kérelem tartalmazza a szolgáltatás igénybe vevőt, az elérendő szolgáltatást és a szolgáltatás elérésének törvényi hivatkozását (jogalapját). Az engedélyezés után a szolgáltatás igénybe vevője megkapja a szolgáltatás elérési jogosultságot, amelyhez igénye szerint egy vagy több kliens autentikációs tokent tud kiváltani.



Ábra 2. Kliens autentikációs token megszerzése

A kliens autentikációs token **egy kapcsolódó félhez, egy szolgáltatáshoz (URL) tartozik és egy jogalapra** vonatkozik a *kérelemben* megadott adatoknak megfelelően.

Egy adott kapcsolódó fél számára kiadott autentikációs tokent így még két paraméter határoz meg: a szolgáltatás és az elérés jogalapja.

A kliens alkalmazás **minden kérésben** küldi a kliens autentikációs tokent, az **x-kk-authentication** HTTP fejlécben, ez alapján **a hívó fél egyértelműen azonosítható.** A hívó fél azonosítása után a KKSZB ellenőrzi, hogy **van-e érvényes szolgáltatás elérési jogosultsága.**

A következő példák a tipikus használati eseteket mutatják be **egy adott kapcsolódó fél** esetén. Mindkét esetben két kérelmet nyújt be, és annak engedélyezése után két szolgáltatás elérési jogosultság jön létre.

- a kapcsolódó fél két **eltérő szolgáltatást** (eltérő URL-ek) használ a KKSZB rendszeren keresztül: ekkor legalább két kliens autentikációs tokent kell használni, hiszen a kliens autentikációs token csak egy szolgáltatáshoz tartozhat, ezért a két szolgáltatáshoz két kliens autentikációs token szükséges.
- a kapcsolódó fél ugyanazt a szolgáltatást használja, de **eltérő jogalapon** veszi azt igénybe - például egyik esetben nyomozati céllal, a másik esetben ügyintézési lekérdezés céllal -, ekkor jogcímenként eltérő tokenek kerülnek kiadásra ugyanarra az interfészre.

A szolgáltatástól és jogalaptól függő token kiadás a szolgáltatások elérésének finoman szabályozhatósága céljából jött létre. Ezzel a szabályozással adott kapcsolódó féltől az adott szolgáltatásra, vagy - törvényi változás esetén - adott jogalapra vonatkozó tokent lehet visszavonni, így ez a visszavonás nem érinti a további szolgáltatás igénybevételét.

A szolgáltatásokat igénybe vevő szervezet, ha adott szolgáltatáshoz, az adott jogalappal kapott elérési jogosultságot, akkor a jogszabályi előírások betartásával és céljával összhangban használhatja csak fel a hozzáférést.

A szolgáltatásokat igénybe vevő szervezet, adott szolgáltatáshoz és adott jogalappal kiadott elérési jogosultságához több autentikációs tokent is ki tud váltani (vagy vissza tud vonni), ezek egyenértékűek. Ugyanakkor eltérő alkalmazásokban is használhatja őket, így szolgáltatás igénybe vevőnek is van lehetősége arra, hogy az RFNY-en keresztül visszavonjon egy tokent, miközben a többi kiváltott tokent ez nem érinti. Ez opcionális, a szolgáltatásokat igénybe vevő szervezet hatóköre és döntése.

Token lejárat előtt célszerű új tokent kiváltani, így átmenetileg legalább két érvényes token lesz az adott szolgáltatáshoz.

Javasolt a tokeneket az alkalmazás konfiguráció részeként, a szolgáltatás URL és kérelemazonosítóval együtt azonosítva tárolni, így adott token az RFNY-ben is könnyen azonosítható, valamint az alkalmazásban könnyen kiválasztható, hogy melyik tokent kell küldeni a kérésben (szolgáltatás URL és jogalap).

2.3. Kötelező HTTP fejlécek

A KKSZB rendszer minimális követelményeket támaszt a Kliens *alkalmazások* felé a kapcsolódáshoz: mindössze a megfelelő HTTP fejlécek beállítása szükséges az adott kérésben. Megjegyzendő, hogy a KKSZB rendszerhez csatlakozásnak egyéb, *nem fejlesztési jellegű* követelményei is vannak.

Minden KKSZB felé érkező kliens alkalmazásnak kötelezően küldenie **kell** a **x-kk-authentication** és a **x-request-id** HTTP fejléceket.

Az **x-request-id** egy olyan HTTP fejléc, amely egy [UUID] v4 értéket tartalmaz, egyértelműen azonosítja a kérést. Ugyanazon kérés újraküldésekor (például megszakad kapcsolat esetén) **x-request-id** értéke is ugyanaz marad.

Az **x-request-id** célja, hogy az üzenetek nyomon követésére fel lehessen használni a rendszerek között (pl.: hibajelentésre, logolásra), valamint lehetőséget adjon a Szolgáltatásoknak az idempotens működés implementálására.

Ajánlott az **x-request-id** értékét minden egyes log sorban, és napló rekordban - kliens alkalmazás és Szolgáltatás oldalon is - tárolni, ezzel egyértelműen ellenőrizhető, hogy adott kéréshez mely log vagy napló bejegyzések tartoznak.

Az **x-request-id** általánosan elterjedt megoldás a kérések egyedi azonosítására, ezért nem tartalmazza az **x-kk-** előtagot (nem KKSZB specifikus).

Mindig az adott Szolgáltatás dönti el, hogy milyen mértékben épít az **x-request-id** használatára. Amennyiben épít rá, úgy az interfész leírásában kell a kliens alkalmazásokat érintő leírást publikálnia.

Praktikus, ha a Szolgáltatás implementációja idempotens módon történik.

Amennyiben a kliens alkalmazás nem kezeli megfelelően az **x-request-id** értékét, akkor eltérő kérések is ugyanazzal a kérés azonosítóval kerülhetnek létrehozásra. Ez kliens alkalmazás hiba, amelyet javítani kell, hiszen a Szolgáltatás azt feltételezheti - anélkül hogy elemezné magát a kérést -, hogy ugyanaz a kérés érkezett be hozzá, így újra - implementációtól függően - már nem dolgozza azt fel.

2.4. KKSZB HTTP válasz fejlécek

A KKSZB segíti a kliens alkalmazás és a szolgáltatás közötti üzenetek nyomon követését, ezért nem csak megkövetel, de szolgáltat is KKSZB specifikus HTTP fejléceket a válaszban, amelyet a kliens alkalmazás saját igénye szerint, opcionálisan felhasználhat.

A kliens alkalmazás minden kérésére az adott válasz az alábbi HTTP fejléceket tartalmazza.

Táblázat 1. HTTP válasz fejlécei

HTTP fejléc	Leírás
x-kk-env	Környezet azonosítója, ahonnan a válasz érkezett.
x-request-id	A kérés azonosítója, amelyhez a válasz tartozik, UUIDv4.
x-kk-tr-id	Tranzakció azonosító, amelyet a KKSZB rendszer generál minden beérkező kérés esetén, UUIDv4.
x-kk-gw-ts	A KKSZB gateway 'epoch time' értéke a kérés beérkezésekor.
x-kk-runtime	Komplett kiszolgálási idő milliszekundumban, amely tartalmazza a KKSZB idejét is.

Az x-kk-tr-id értékét célszerű logolni a kliens oldalon, így ennek valamint az x-request-id megadásával a KKSZB rendszerben az üzenetek megtalálhatók és összerendelhetők. **Amennyiben a KKSZB rendszer fejlesztőivel felveszi a kapcsolatot, mindig hivatkozza meg az x-kk-env, x-kk-tr-id és az x-request-id értékét!**

Bizonyos hibák esetén ezek az értékek nem állnak rendelkezésre, ilyen például, ha a kliens nem éri el a KKSZB rendszert, vagy ha invalid HTTP üzenetet küld.

A válasz fejlécek felhasználhatók kliens alkalmazás oldalon logolási célokra is, amely segíti az üzenetek nyomon követhetőségét.

A futási idők mérése segíti a hibakeresést és a szolgáltatás minőség monitorozását.

2.5. Egyéb HTTP fejlécek

Egy szolgáltatás tetszőleges egyéb kérés vagy válasz HTTP fejlécet definiálhat, kivéve az **x-kk-...** kezdetű HTTP fejléceket, amelyek a KKSZB részére vannak fenntartva. Ezen Szolgáltatás specifikus fejlécek kezelésének módját kizárólag a szolgáltatásról szóló dokumentáció specifikálja.

A webszerverek eltérő HTTP fejléc hosszakat fogadnak el. Ezt az egyéb HTTP fejlécek használatakor figyelembe kell venni a Szolgáltatás oldalon és ellenőrizni kell a webszerver konfigurációt.

2.6. Kapcsolat ellenőrzése: echo szolgáltatás

Amennyiben a kapcsolódó fél megkapta a hozzáférést a KKSZB rendszerhez, úgy a Rendszer Felhatalmazási Nyilvántartás webes felületén keresztül be tud lépni. A kapcsolódó fél, automatikusan megkapja a KKSZB rendszer *echo szolgáltatásához* a hozzáférést, azt nem kell külön igényelni.

2.6.1. Echo szolgáltatás

A KKSZB rendszer a **/kkszb/echo/v1** címen biztosít egy alapértelmezetten engedélyezett szolgáltatást, amellyel a kliens alkalmazás **a KKSZB rendszert és saját kapcsolatát ellenőrizheti.**

A KKSZB echo szolgáltatás a beküldött adat tartalmat adja vissza, GET és POST kérésekkel hívható a **PROD:** **https://gw.kkszb.gov.hu/kkszb/echo/v1** (éles környezet) és a **INT:** **https://gw.int-kkszb.gov.hu/kkszb/echo/v1** (integrációs teszt környezet) szolgáltatás URL-eken is.

Például **GET** kérés esetén:

- **https://gw.kkszb.gov.hu/kkszb/echo/v1/test?key=value** esetén a válasz (*response body*) a "/echo/test?key=value" sztring lesz.
- **POST** kérés esetén a beküldött *body* lesz visszaadva, amelynek tartalma az *enctype* függvénye:

enctype="x-www-form-urlencoded" kódolás esetén:

POST **https://gw.kkszb.gov.hu/kkszb/echo/v1/** esetén a válasz: "key1=value1&key2=value2" lesz.

enctype="multipart/form-data" kódolás esetén:

POST **https://gw.kkszb.gov.hu/kkszb/echo/v1/** esetén a válasz:

enctype="multipart/form-data" válasz

```
----------------------------975731998104297348364664 Content-Disposition: form-data;
name="key1" value1 ----------------------------975731998104297348364664 Content-
Disposition: form-data; name="key2" value2
----------------------------975731998104297348364664--
```

Az egyéb kódolás esetén (pl.: text/plain, binary) pontosan az elküldött tartalom kerül visszaadásra.

A szolgáltatás tesztelésekor az autentikációs tokent, *x-kk-authentication* és az *x-request-id* értékét meg kell adni.

A szolgáltatás tesztelése történhet akár parancssorból is például a *curl* program használatával is.

HTTP kérés KKSZB echo szolgáltatás végpontra curl segítségével

```
curl -k -H "x-request-id: aaaabbbb-1111-4d89-b26c-5dbe98b976f9" -H "x-kk-
authentication: autentikacios_token_helye" https://gw.int-
kkszb.gov.hu/kkszb/echo/v1?test=1234
```

A fenti kérésben az *x-request-id* értékét az UUID v4-es szabvány szerint állítsa be, lehetőleg kérésenként eltérőre, így a hibakeresés könnyebb.

Az alábbi válaszok lehetnek a fenti kérésre:

- HTTP 200 OK esetén /echo?test=1234
- HTTP 403 Forbidden esetén, ha szerepel a válasz HTTP fejlécben a *x-kk-gw-status-message* mező és értéke "Permission denied", akkor az x-kk-authentication token nem megfelelő (pl.: szintaxis hiba, nem ehhez a szolgáltatáshoz tartozik, stb.)

- HTTP 403 Forbidden esetén, ha nem szerepel a válasz HTTP fejlécben a *x-kk-gw-status-message* mező akkor a kérést indító gép IP címe nincs beengedve a KKSZB rendszerbe, vegye fel a kapcsolatot a KKSZB rendszer felelőseivel
- további válaszkódokhoz a HTTP státusz kódok függelék tartalmaz leírást

Chapter 3. Szinkron kommunikáció kezelése szolgáltatás esetén

A KKSZB rendszer tervezésekor szempont volt, hogy a követelményeket minimálisra csökkentsük a kliens alkalmazásokkal és szolgáltatásokkal szemben. Ennek megfelelően a KKSZB a szolgáltatásokkal szemben az alábbi követelményt támasztja:

- kezelnie **kell** a HTTP protokollt
- a KKSZB által alkalmazott azonosítási rendszert **kell** használnia
- szolgáltatnia kell az állapot információ végpontot

A KKSZB rendszerre kötött szolgáltatásnak be kell tartani néhány szabályt, amely a KKSZB használatának feltétele:

- A szolgáltatás követelményei nem állhatnak ellentétben a KKSZB követelményével, attól nem lehet megengedőbb és nem tehet egyéb előírást a KKSZB követelményekre vonatkozóan.
- A HTTP fejlécekben az **x-kk-...** kezdetű névtereket nem használhatja.

3.1. KKSZB azonosítási rendszer

A KKSZB rendszerre kötött összes kliens alkalmazás azonosítva van, így a Szolgáltatás biztos lehet abban, hogy a számára beérkező kérés az adott klienstől érkezett. A KKSZB átvállalja az alkalmazás szintű azonosítást és az elérési jogosultság ellenőrzését a szolgáltatástól.

Mivel a KKSZB elvégzi az alkalmazás szintű azonosítást, így a meglévő szolgáltatásban azt meg **kell** szüntetni. Új szolgáltatás esetén **nem kell** saját megoldást tervezni és implementálni, hanem a KKSZB rendszerét **kell** megvalósítani.

A szolgáltatást végző alkalmazást úgy kell megírni, hogy azt a továbbiakban a KKSZB rendszerre bízza és a tőle kapott kliens alkalmazás azonosítás eredményét használja.

Így a továbbiakban az adott szolgáltatás mentesül a kliens alkalmazások azonosításával járó adminisztráció alól és a kapcsolódó kliens alkalmazások felé nem követel meg egyedi azonosítási igényeket. Meglévő szolgáltatások esetén ez az eddigi azonosítást, elérési jogosultság ellenőrzést végző kód törlését, átalakítását vagy inaktívvá tételét és a KKSZB erre vonatkozó HTTP fejléc adatainak használatát jelenti.

A KKSZB azonosítási rendszer csak a kapcsolódó kliens alkalmazást azonosítja és ellenőrzi, hogy egyáltalán létrejöhet-e a kliens alkalmazás és a szolgáltatás között a kapcsolat. Semmilyen ellenőrzést nem végez felhasználók személyére, vagy a szolgáltatás saját, egyedi jogosultság kezelésére vonatkozóan.

A fentiek alapján továbbra is az alkalmazás belügye marad az, hogy milyen felhasználó személyeket enged, milyen adatokkal vagy adatkörökre és milyen algoritmus szerint engedélyezi a szolgáltatás használatát.

3.2. KKSZB azonosítási rendszer használata

A Szolgáltatás az **x-kk-client-id** HTTP fejlécben megkapja a kérést kezdeményező Kliens alkalmazás azonosítóját.

Az **x-kk-client-id** használata opcionális, szolgáltatás saját, belső igényeitől függ. **Ajánlott** logolásra és naplózásra felhasználni, így könnyebb az üzenetek azonosítása és az esetleges hibakeresés.

A KKSZB az azonosítást még a szolgáltatáshoz történő hozzáférés előtt elvégzi, így az nem terheli a szolgáltatást. **Sikertelen azonosítás esetén a Szolgáltatást nem éri el az üzenet.**

A KKSZB a sikeres azonosítás eredményét a HTTP protokollon kezeli: a HTTP fejlécbe helyezi el az **x-kk-client-id** és értékét.

A Szolgáltatás a továbbiakban az **x-kk-client-id** értéke alapján tudja eldönteni, hogy ki szólította meg.

Az **x-kk-client-id** szerkezetéről és Szolgáltatás oldali használatának lehetséges módjáról a x-kk-client-id szerkezete fejezetben található bővebb leírást.

A Szolgáltatás használatához a *Szolgáltatás Felelős* adja a hozzáférést az RFNY webes alkalmazáson keresztül. A Szolgáltatás felelőssége, hogy az őt használó Kliensek részére a hozzáférést engedélyezze vagy visszavonja.

Egy KKSZB Szolgáltatás csak a KKSZB gatewayen keresztül érhető el. A *KKSZB végpont* az ahol az adott szolgáltatás elérhető a kliens alkalmazás számára. Ez a szolgáltatási pont van megfeleltetve a valós szolgáltatási végpontnak, ahol a szolgáltató alkalmazás fogadja a kéréseket.

Amennyiben a kliens alkalmazás nem rendelkezik a megfelelő jogosultságokkal, úgy a Szolgáltatás elérése meg lesz tagadva. Bővebben a kliens alkalmazáson tapasztalható hibaüzenetekről a Hibakezelés és válaszkódok fejezetben talál.

3.2.1. Bővebb információk a kliens alkalmazásról

A KKSZB a kliens alkalmazás sikeres azonosítása esetén az **x-kk-access-token** HTTP fejlécbe helyezi Base64url kódolt formában a Kliensről szóló információkat.

Az **x-kk-access-token** használata opcionális, szolgáltatás saját, belső igényeitől függ.

A KKSZB elhelyezi az úgynevezett kliens access tokent a HTTP fejlécben az **x-kk-access-token** kulccsal. Ezt a szolgáltatás olvashatja dekódolás után (base64url). A tokenben szereplő értékek lehetőséget biztosítanak arra, hogy további információt kapjunk a kliensről, úgy mint a token birtokos szervezet, kapcsolattartó neve. A KKSZB a HTTP kérés fejlécébe helyezi a fontosabb adatokat a tokenből, amelyek részletes leírása a KKSZB HTTP kérés fejlécek fejezetben található.

Részletes információ az Kliens access token oldalon található.

A KKSZB rendszer az **x-kk-access-token** értékeit a HTTP kérés fejlécbe helyezi, így

a hívott szolgáltatás a token kibontása nélkül azonnal fel tudja használni ezeket. Ugyanakkor ezek az adatok - ellentétben az **x-kk-access-token**-nel - nincsenek aláírva, ezért a felhasználásának kockázatát a szolgáltatás oldalon mérlegelni kell.

3.3. x-request-id használata

Az **x-request-id** egy olyan HTTP fejléc, amely egy [UUID] v4 értéket tartalmaz, egyértelműen azonosítja a kérést. Ugyanazon kérés újraküldésekor (például megszakad kapcsolat esetén) **x-request-id** értéke is ugyanaz marad. Az **x-request-id** célja, hogy az üzenetek nyomon követésére fel lehessen használni a rendszerek között (pl.: hibajelentésre, logolásra), valamint lehetőséget adjon a Szolgáltatásoknak az idempotens működés implementálására.

Amennyiben a kliens alkalmazás nem kezeli megfelelően az **x-request-id** értékét, akkor eltérő kérések is ugyanazzal a kérés azonosítóval kerülhetnek létrehozásra. Ez kliens alkalmazás oldali hiba, amelyet javítani kell, hiszen a Szolgáltatás azt feltételezheti - anélkül hogy elemezné magát a kérést -, hogy ugyanaz a kérés érkezett be hozzá, így - implementációtól függően - újra már nem dolgozza azt fel.

Javasolt a kérés egyértelmű nyomon követhetősége céljából naplózni és logolni az **x-request-id** értékét Szolgáltatás oldalon is.

Bizonyos nem üzemszerű esetekben előfordulhat az azonosító ismétlődése, ekkor is célszerű naplózni és/vagy logolni az **x-request-id** értékét.

Mivel az **x-request-id** előállítását a Kliens alkalmazás végzi, ezért nem garantált annak a megfelelő rendelkezésre állása (pl.: nem UUIDv4, ismétlődik), **ezért a KKSZB rendszer minden kéréshez generál egy egyedi tranzakció azonosítót UUIDv4 formában, amelyet a Szolgáltatás az x-kk-tr-id HTTP fejlécben kap meg.** Ezt az értéket - hibátlan kommunikáció esetén - a Szolgáltatást hívó Kliens alkalmazás is megkapja.

Javasoljuk, hogy igény szerint az x-kk-tr-id is kerüljön logolásra nyomon követés céljából.

3.4. KKSZB HTTP kérés fejlécek

A KKSZB segíti a kliens alkalmazás és a Szolgáltatás közötti üzenetek nyomon követését, ezért nem csak megkövetel, de szolgáltat is KKSZB specifikus HTTP fejléceket a kérésben, amelyet Szolgáltatás opcionálisan, saját igénye szerint felhasználhat.

A hívás **x-kk-access-token** JWT formátumú, amelynek középső része (két pont között) tartalmazza a hívás metaadatait, amelyből az adatokat a KKSZB a HTTP fejlécbe helyezi **x-kk-** előtaggal, hogy ezzel a szolgáltatásnak már ne keljen foglalkoznia (kényelmi funkció). Mindemellett az AccessToken is megküldésre kerül az **x-kk-access-token** mezőben, ezzel biztosítva azt, hogy a HTTP fejlécben küldött adatok hitelesíthetők legyenek (a Szolgáltatás nyújtója dönt).

Az AccessToken pontos szerkezetét a Kliens access token függelék tartalmazza.

Táblázat 2. HTTP kérés fejlécei

HTTP fejléc	Leírás
x-request-id	A kérés egyedi azonosítója, sztring, UUID formában, a szolgáltatás igénybe vevője adja meg.
x-kk-tr-id	Tranzakció azonosító, amelyet a KKSZB rendszer generál minden beérkező kérés esetén, UUIDv4. Egyediségét a KKSZB rendszer garantálja.
x-kk-access-token	A tokenben szereplő értékek lehetőséget biztosítanak arra, hogy további információt kapjunk a Kliensről, sztring, JWS formában (base64 kódolt), max. 2000 karakter, kényelmi szempont miatt ebből számos értéket a HTTP fejlécbe helyez a KKSZB.
x-kk-env	Környezet azonosítója, ahonnan a kérés érkezett, sztring, max. 25 karakter, lehetséges értékei: PROD, INT, STAGING, értékét a KKSZB futtató környezete definiálja.
x-kk-gw-ts	A KKSZB gateway 'epoch time' értéke a kérés beérkezésekor, numerikus, pl.: 1504703686, a KKSZB futtató környezete állítja be.
x-kk-client-id	A kérést küldő egyedi azonosítója, sztring, max. 100 karakter, URN formában, pl.: urn:pid:kkszb:xxxx, az AccessToken sub mezőjével egyezik meg.
x-kk-sap-id	A szolgáltatás elérési jogosultság egyedi azonosítója, sztring, UUID formában, az AccessToken sapId mezőjével egyezik meg.
x-kk-sap-name	A szolgáltatás elérési jogosultság neve, amely a könnyebb azonosíthatóságot és nyomon követhetőséget segíti, sztring, max. 30 karakter, az AccessToken sapName mezőjével egyezik meg.
x-kk-token-name	Az autentikációs token neve, szabad szöveges string, max. 20 karakter, például: az alkalmazás címkéje, amelyben felhasználásra kerül, az AccessToken authtokenName mezőjével egyezik meg.
x-kk-legal-basis-id	Jogalap azonosító, 8 karakter hosszú hexadecimális sztring, amelyet a KKSZB generál, sztring, hexadecimális érték, pontosan 8 karakter, pl.: a70d2dfd, az AccessToken legalBasisId mezőjével egyezik meg.
x-kk-legal-basis-code	Jogalapkód, opcionális (ha nincs értéke, akkor a fejlécben nem szerepel), **amelyet az RFNY webes felületén a Szolgáltatás felelős ad meg, így közvetlenül használható a Szolgáltatást nyújtó alkalmazásban,** a következő karaktereket tartalmazhatja: a-z, A-Z, 0-9, -, _, / és . (pont) karakterek, max. 20 karakter, az AccessToken legalBasisCode mezőjével egyezik meg.

HTTP fejléc	Leírás
x-kk-security-class	A szolgáltatást hívó fél kérésének IBTV biztonsági osztály értéke, numerikus érték 2-5 között (implicit), az AccessToken securityClass mezőjével egyezik meg, a Szolgáltatás fel tudja használni arra, hogy a kérést a biztonsági szintnek megfelelő útvonalra terelje. Értéke a Szolgáltatás Elérési Jogosultság Kérelem kitöltésekor kerül megadásra a Kapcsolódó fél által.

A HTTP fejlécek közül az **x-kk-sap-name** és az **x-kk-token-name** értékét a HTTP szabvány követelménye szerint *escape-elni* kell, amelyre a KKSZB a javascript **encodeURI()** függvényét használja. A megadott karakterhosszúságok a dekódolt értékre vonatkoznak, a **kódolt értékek hosszabbak lehetnek.**

A válasz fejlécek felhasználhatók logolási célokra is, amely segíti az üzenetek nyomon követhetőségét.

Példa HTTP fejlécre, amely a szolgáltatást eléri (csak x-kk- névtérbe tartozóak)*

```
x-kk-access-token=
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjEifQ.eyJqdGkiOiJkNzNlYzBmYy1lZTU2LTQwZD
ktYjVhYS1iZDE3NDZlMmJhM2MiLCJpc3MiOiJ1cm46c3lzOmtrc3piOmZueSIsInN1YiI6InVybjpwaWQ6a2tz
emI6cGVlcjEiLCJ0eXBlIjoidXJuOnRva2VuOmtrc3piOmNsaWVudDphY2Nlc3MiLCJpYXQiOjE0OTMxMTM2NT
MsIm5iZiI6MTQ5MzExMzY1MywiZXhwIjoxNDkzMTE0MjUzLCJzZXJ2aWNlSWQiOiJMMnBoY20xMUwzTmxjblpw
WTJVMUwzWXgiLCJzZXJ2aWNlVXJpIjoiL2phcm11L3NlcnZpY2U1L3YxIiwiYXV0aHRva2VuSnRpIjoiMmQ4Mj
VmNmQtMWZhYi00ZmNhLThiMTItMGVmMTQ1OGUwZDAwIiwic2FwSWQiOiI0MWY3NWU5Ni1lYjQ2LTQ1MGMtOTQ2
ZS01MTdiZWNmNWVjOWEiLCJzYXBOYW1lIjoiZGVmYXVsdCIsImF1dGh0b2tlbk5hbWUiOiJUb2tlbjEiLCJsZW
dhbEJhc2lzSWQiOiJkY2RmMWU4MiIsImxlZ2FsQmFzaXNDb2RlIjoiSkFSMTIwMkEiLCJzZWN1cml0eUNsYXNz
Ijo0LCJ2ZXJzaW9uIjoxfQ.UUnMFj2oyG3tsDRHQCKB5yFL2rSX-
SZpHsAVie522FSAhAUXePZmJJNCYq8wVqkzrBvQ_VmlM3X5jyKuDR_TrKeNAOWVrCG4B0VwsRc9U0hBNuaxYLe
jktH_z-8SFdgIF7nlBUoQ2JnFOxZQLlpKQY7-OHObUj6aJBDbnqt8O731Jga2QjY-
1HM9e9dbONLXBY_FrhXYpQZWDvokRNndAqZwNQ2YeIy8ONds6cUhQTh_8-
wRjOBvkqvQAjXvpraJMG5_WtWYWY2n_wMQtZgdaszxjdnT191KbQoLqoO-P-
YVAmR7FrF6yXdJvvXZrnNZM4kIs4gCcXn_xH3njomB-Q"
x-kk-env="INT"
x-kk-tr-id="abc75e96-1246-4566-12f2-a1bbecf5ff10"
x-kk-gw-ts=1504703686
x-kk-client-id="urn:pid:kkszb:peer1"
x-kk-sap-id="41f75e96-eb46-450c-946e-517becf5ec9a"
x-kk-sap-name="default"
x-kk-token-name="Token1"
x-kk-legal-basis-id="dcdf1e82"
x-kk-legal-basis-code="JAR1202A"
x-kk-security-class=4
```

A Szolgáltatónak lehetősége van eldöntenie, hogy az adott, alacsonyabb IBTV (lásd [it-biztonsag]) osztályba sorolt rendszertől érkező kérést kiszolgálja vagy nem szolgálja ki, ezt **az RFNY webes felületén a szolgáltatás elérési jogosultság engedélyezési folyamata során teheti meg.**

Amennyiben adott Szolgáltatáshoz egy hálózati ponton keresztül kezeli az eltérő biztonsági szinteket - egy Szolgáltatás URL van az összes több IBTV szinthez -, úgy a **beérkező kérés feldolgozásakor** az **x-kk-security-class** HTTP fejléc alapján tudja az útvonalválasztást biztosítani.

3.5. Polimorf válasz adása

Vannak Szolgáltatások amelyek URL-ben nem különböztetik meg az interfészüket eltérő válaszok esetén. Például egy Szolgáltatás ugyanazon az URL-en várja a Rendőrség és a Biztosítók által indított lekérdezéseket. Ebben az esetben a Szolgáltatás oldali alkalmazás egy *algoritmus és a kérésben érkező adatok alapján* eldönti, hogy kiadható-e a az adott adat, vagy sem.

Természetesen a fenti eset jogszabály által szabályozott, így az adatszolgáltatáshoz meghatározható a jogalap. A KKSZB rendszer a **x-kk-legal-basis-id és x-kk-legal-basis-code** értékben az adott szolgáltatáshoz mindig közli a **jogalap azonosítóját, és kódját** így a szolgáltatás ezen érték alapján meg tudja határozni a válasz tartalmát.

A szolgáltatás joga és felelőssége eldönteni, hogy milyen *algoritmus* alapján, milyen *adatok* felhasználásával állítja elő a válaszüzeneteket.

Azokat a szolgáltatásokat, amelyek **ugyanazon lekérdezésre, ugyanazon végponton más választ adnak** attól függően, hogy mely rendszertől (vagy felhasználótól, vagy csatornán) érkezett a kérés, polimorf szolgáltatásoknak nevezzük.

Vannak olyan szervezetek, ahol a szervezeten belül is eltérő választ kapnak, így maga az **x-kk-client-id** alapján nem eldönthető, hogy ki kell-e adni az adott értéket vagy sem.

Ebben az esetben a Szolgáltatás számára javasolt megoldások a következők lehetnek:

1. a Szolgáltatás URL-eket eltérően kell felvenni, így elérhető, hogy a szolgáltatás ne legyen polimorf,
2. ha URL-ben megegyező, akkor a Szolgáltatás az interfész leírás szerint az **üzenet testben** várhat olyan értéket, amely alapján különbséget kell tennie (a közigazgatási rendszerek többségénél használt megoldás)
3. ha URL-ben megegyező, akkor a Szolgáltatás az interfész leírás szerint az **HTTP fejlécben** várhat olyan értéket, amely alapján különbséget kell tennie

Az 1. megoldás átlátható, értelmezhető Szolgáltatás URL-t eredményez, hátránya, hogy a Szolgáltatás Katalógusban többször szerepel az interfész leírás, apró eltérésekkel. Ez a megoldás akkor ajánlott, ha a két válasz jelentősen eltér, vagy az interfész leírás a Szolgáltatás Katalógusban védendő információ: azonosításhoz kötött interfész leírás megtekintés.

A 2. megoldás esetén egy interfész leírás szükséges a Szolgáltatás Katalógusban, az üzenet testben jelenleg szereplő mező - ami alapján a polimorf válasz képzés történik - továbbra is felhasználható. A jelenlegi (2017) rendszerek nagy része így működik.

A 3. megoldás esetén a Szolgáltatás Katalógusban az interfész leírás tartalmazza a HTTP fejléc kezelés módját, az üzenet test módosítására nincs szükség.

A polimorf válasz képzéséhez akár a KKSZB által nyújtott fejlécek is felhasználhatók:

- **x-kk-client-id:** azonosítható általa, hogy mely szervezet, szerv, szervezeti egység, stb., küldte a kérést.
- **x-kk-sap-id:** az engedélyezett [szolgáltatás elérési jogosultság] egyedi azonosítója az adott Szolgáltatásban felhasználható, így az engedély és a válasz tartalom egyértelműen megfeleltethető.
- **x-kk-legal-basis-id:** jogalap egyedi azonosítója a KKSZB rendszerben, amely az adott szolgáltatás esetén egyedi, a KKSZB rendszer képezi, és a RFNY felületén a Szolgáltatás Felelős meg tudja tekinteni az értékét.
- **x-kk-legal-basis-code:** jogalapkód, **amelyet az adott szolgáltatás közvetlenül tud értelmezni,** a Szolgáltatás Felelős adja meg az RFNY felületén keresztül. Segítségével az adott jogalapkódhoz egyértelműen meghatározható a válasz tartalma.

A Szolgáltatás Katalógusban szereplő interfész leírás esetén a Szolgáltatónak jeleznie kell azt, hogy a válasz polimorfizmusát mely mezők és/vagy HTTP kérés fejlécek befolyásolják.

3.6. Állapot információ végpont

A szolgáltatásnak nyújtania **kell** egy olyan végpontot, amelynek lekérdezésekor az alkalmazás 200 OK HTTP státusz kóddal válaszol, **akkor és csak akkor,** ha üzemszerűen működik az alkalmazás. Ezen a végponton a KKSZB rendszer adott időintervallumonként rendszeres ellenőrzést végez, amely alapján meghatározza a szolgáltatás állapotát.

A KKSZB rendszer ezen állapot információ végpontot (*healthcheck url*) egy HTTP GET kéréssel szólítja meg.

Követelmények a HTTP állapot információ végpontra nézve:

- a HTTP GET kérésre válaszolnia **kell** (természetesen, ha az alkalmazás nem fut, akkor ez nem történik meg)
- a GET kérésre 200 OK HTTP státuszkóddal **kell** válaszolnia, amennyiben a szolgáltatás üzemszerűen használható
- az elérési URL-nek statikusnak **kell** lennie: nem lehet benne változó adat, pl.: sessionId, stb.
- az elérési URL **nem tartalmazhat** szenzitív, személyes vagy változó adatot, pl.: elérési token (szenzitív), adószám (tv. szinten változhat a szerkezete), név, stb.
- konzisztensen, egyértelműen **kell** reprezentálnia a szolgáltatás állapotát: nem lehet olyan eset, hogy 200 OK a válasz, ha a szolgáltatás üzemszerűen nem érhető el a KKSZB-ből
- a GET kérésre adott válasz a KKSZB oldalon ignorálva van, de **nem tartalmazhat** semmilyen szenzitív, személyes információt, pl.: adatbázis URL, process id, stb. Mivel a KKSZB a választ (HTTP body) nem dolgozza fel és lezárja a kapcsolatot, így a szolgáltatás oldalon a kapcsolat megszakadásának hibájára utaló üzenet keletkezhet, amely üzemszerű állapot, figyelmen kívül kell hagyni.
- HTTPS végpont legyen

- a GET kérést **nem szabad** autentikálnia: az állapot információ végpont URL-je szakrendszeri, egyedi azonosítás nélkül elérhető legyen úgy, hogy azt csak a KKSZB kliens autentikációs tanúsítványával lehessen meghívni (HTTPS).

A KKSZB rendszer a 200 OK válasz esetén - a válasz tartalmától függetlenül - a szolgáltatást üzemszerűen működőnek tekinti.

A szolgáltató alkalmazás fejlesztőjének a felelőssége, hogy a szolgáltatás elérhetőségét valós értéken közvetítse az állapot információ végponton keresztül.

A KKSZB 1 perces időközönként ellenőrzi a szolgáltatások elérhetőségét.

Az *Állapot információ végpont* URL-t a szolgáltatás KKSZB rendszerbe történő bekötéséhez közölni kell a *szolgáltatás létrehozási kérelem* űrlapon. Amennyiben változás szükséges, azt a változtatás előtt be kell jelenteni.

3.7. Egyéb HTTP fejlécek

Egy Szolgáltatás tetszőleges egyéb kérés vagy válasz HTTP fejlécet definiálhat, kivéve az **x-kk-...** kezdetű HTTP fejléceket, amelyek a KKSZB részére vannak fenntartva. Az egyedi Szolgáltatás fejlécek kezelésének módját kizárólag a Szolgáltatásról szóló dokumentáció specifikálja.

A KKSZB rendszerben maximum fejléc hossz (az URL hosszt is beleértve) 80 kbyte. Az UTF kódolás miatt a használható karakter szám 80 000 karakter alatti lehet.

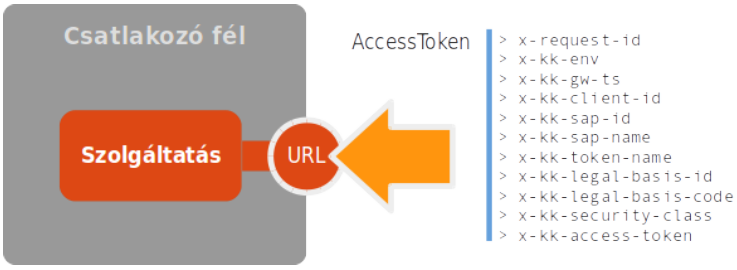
A szolgáltatás oldali webszerverek eltérő HTTP fejléc hosszakat fogadnak el. Ezt az egyéb HTTP fejlécek használatakor figyelembe kell venni és ellenőrizni kell a Szolgáltatás oldali webszerver konfigurációt.

3.8. Szolgáltatás tesztelése

A KKSZB csak a HTTP fejléceken keresztül befolyásolja a Szolgáltatásokat, így ezen adatok megadásával elvégezhető a tesztelés. A tesztelést érdemes automatizálni, így nagyobb eséllyel kiszűrhetők a regressziós hibák.

Mielőtt a Szolgáltatást összekötné a KKSZB rendszerrel, **a Szolgáltatását saját környezetében a KKSZB nélkül tesztelje le,** ehhez segítséget és útmutatást az Alkalmazás szintű (szolgáltatás) teszt fejezetben talál.

A képen látható, hogy a Szolgáltatás azokkal az adatokkal van meghívva - AccessToken, HTTP fejléc értékek) - amelyeket a KKSZB rendszertől megkap egy kérés folyamán, **ezzel a Szolgáltatás ellenőrizhető, hogy megfelelően kezeli-e a KKSZB rendszertől kapott HTTP fejléc adatokat.**



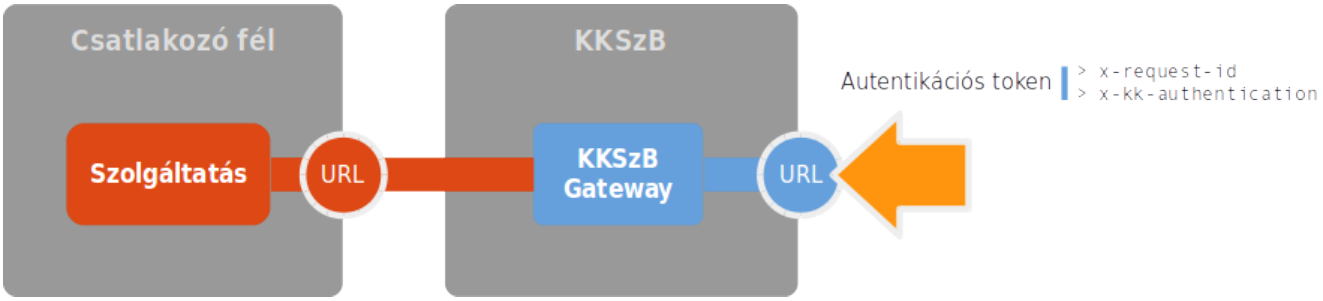
Ábra 3. Alkalmazás szintű teszt: valós szolgáltatási végponton

A fenti teszten ellenőrizhető az alkalmazás megfelelő működése, **így kiszűrésre kerülnek az alkalmazásban rejlő hibák,** amely szükséges feltétele annak, hogy a későbbi integrációs tesztelés folyamán valóban az integrációt tesztelhessük.

Amennyiben az alkalmazás szintű teszt sikeres, csak utána folytassa a teszteket az Integrációs teszttel.

Amikor a Szolgáltatás összekötésre kerül a KKSZB integrációs (INT) környezetével, a korábban összeállított teszteket ott is le lehet futtatni, ehhez a Szolgáltatás tesztelése fejezetben talál további információt.

A képen látható, hogy úgy kerül a Szolgáltatás meghívásra, mintha valós működés lenne - Autentikációs token és x-request-id kerül átadásra. **Az integrációs teszt alatt az integrált rendszer - a Szolgáltatás és KKSZB -, együtt kerül tesztelésre, ezzel ellenőrizhető a működés valós körülmények között.**



Ábra 4. Integrációs teszt: KKSZB szolgáltatási végponton

Amennyiben az alkalmazás szintű teszt folyamán nem tapasztalt hibát, de az integrációs teszt folyamán igen, úgy ellenőrizze a KKSZB Szolgáltatás beállításait az RFNY rendszer webes felületén.

Mivel a KKSZB nem olvassa, nem módosítja, nem értelmezi, nem logolja a Szolgáltatást elérő kéréseket és válaszokat, ezért ezek tesztelése továbbra is a Szolgáltatás belső ügye és feladata.

3.8.1. Alkalmazás szintű (szolgáltatás) teszt

Az alábbiakban csak a KKSZB rendszerhez köthető HTTP fejlécek küldését mutatjuk be, az alkalmazás által fogadott és küldött üzenetek előállítása a Szolgáltatás feladata.

Tesztadatok előállítása

A Szolgáltatást a KKSZB HTTP kérés fejlécek fejezetben felsorolt adatok érik el, amelyek a

következőképpen épülnek fel:

- AccessToken adatai alapján az x-kk-access-token JWT generálása, lásd Kliens access token
- adott AccessToken adatok kiemelése a HTTP fejlécbe, lásd KKSZB HTTP kérés fejlécek
- x-kk-env és x-kk-gw-ts adatokkal bővítés a kkszb-gateway alkalmazásban

A fentiek alapján egy AccessToken adatai birtokában előállítható a tesztadat.

Az AccessToken KKSZB HTTP kérés fejlécek fejezetben leírt értékeit a teszteléshez tetszőlegesen beállíthatja.

Az alábbi AccessToken alapján generálunk egy JWT kódolt AccessTokent a https://jwt.io oldalon, de természetesen tetszőleges JWT generátor alkalmazás használható ehhez.

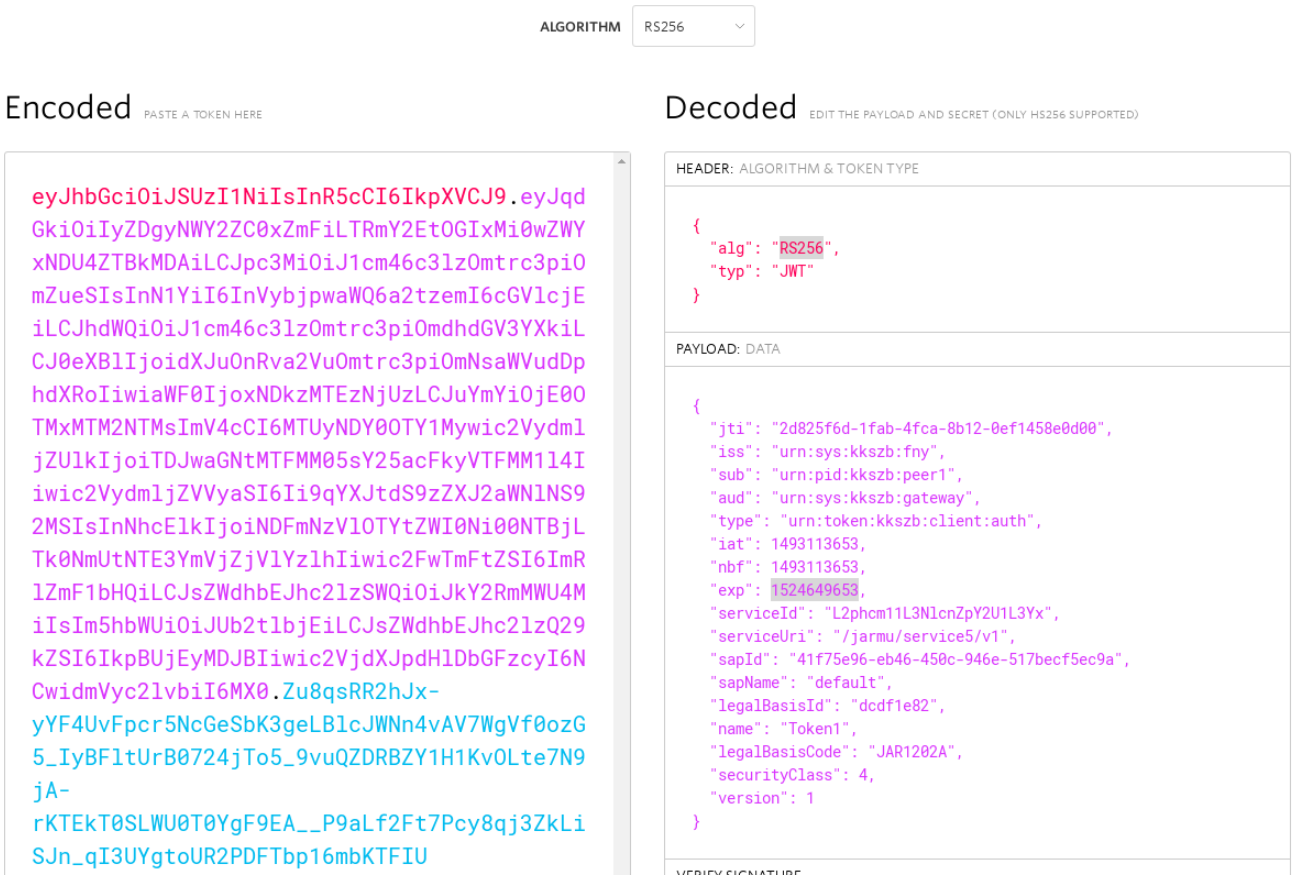
Soha ne adjon meg éles adatokat idegen weboldalakon!

Másolja be az alábbi tartalmat a https://jwt.io oldal `PAYLOAD: DATA` részbe, és válassza ki a fenti legördülő listából az `RS256` értéket, az eredmény JWT base64 kódolva bal oldalon azonnal megjelenik.

AccessToken


```
{
  "jti": "d73ec0fc-ee56-40d9-b5aa-bd1746e2ba3c",
  "iss": "urn:sys:kkszb:fny",
  "sub": "urn:pid:kkszb:peer1",
  "type": "urn:token:kkszb:client:access",
  "iat": 1493113653,
  "nbf": 1493113653,
  "exp": 2224649999,
  "serviceId": "L2phcm11L3NlcnZpY2U1L3Yx",
  "serviceUri": "/jarmu/service5/v1",
  "authtokenJti": "2d825f6d-1fab-4fca-8b12-0ef1458e0d00",
  "sapId": "41f75e96-eb46-450c-946e-517becf5ec9a",
  "sapName": "default",
  "authtokenName": "Token1",
  "legalBasisId": "dcdf1e82",
  "legalBasisCode": "JAR1202A",
  "securityClass": 4,
  "version": 1
}
```


Ne feledje, hogy a valós működés folyamán - integrációs (INT) és éles (PROD) környezetekben -, **csak a sapName, legalBasisCode és serviceUri (névtér utáni rész) értékét tudja közvetlenül beállítani,** az authtokenName értékét a szolgáltatását igénybe vevő fél adhatja meg, minden egyéb mást a KKSZB rendszer állít elő.



Ábra 5. jwt.io oldalon előállított JWT kódolt AccessToken

KKSZB függő mezők kezelése teszteléskor

Amennyiben a szolgáltatásának logikája épít a következő adatok valamelyikére, úgy **ezen adatokat mindenképpen tegye konfigurálhatóvá az alkalmazásában** és az automata tesztek futtatásakor ezen adatokat *igénye szerint állítsa be:*

- x-kk-sap-id (sapId),
- x-kk-sap-name (sapName),
- x-kk-token-name (authtokenName),
- x-kk-legal-basis-id (legalBasisId),
- x-kk-security-class (securityClass).

A x-kk-legal-basis-code (legalBasisCode) értékét Ön adja meg, nem szükséges mockolni, így nem kötelező konfigurálhatóvá tenni.

Azért kell konfigurálhatóvá tenni a fenti adatokat, mivel a KKSZB rendszerben a Szolgáltatás létrehozásakor vagy az elérési jogosultság megadásakor kapnak értéket. Mivel még a csatlakozás előtt szeretnénk fejleszteni, tesztelni a szolgáltatást biztosító alkalmazásunkat, ezért ezeket az értékeket a tesztekhez igényünk szerint beállítjuk és amikor INT vagy PROD környezetben elindítjuk az alkalmazást akkor már csak konfigurációban kell a beállítani azt - ezzel elkerüljük a kódok és értékek *bedrótozását* a forráskódba.

Amennyiben az **id** végű mezőket nem használja és helyette a **name** végűeket használja, akkor nem kötelező konfigurációként megadni az adatokat, de ajánlott, pusztán a *bedrótozott* adatok elkerülése végett.

A **x-kk-sap-name** a Szolgáltatás Felelős adja meg a Szolgáltatás Elérési Jogosultság Kérelem engedélyezésekor, így garantálható, hogy előre ismert legyen és minden környezetben megegyezzen.

A **x-kk-token-name** értékét a szolgáltatás igénybe vevője adja meg, így az eltérhet környezetenként, ha előtte a Szolgáltató és az igénybe vevő nem egyeztet.

Teszt futtatása

Amennyiben sikerült előállítani az AccessTokent JWT kódolva és tisztázott a szolgáltatás valós végpontja (URL-je), úgy az alábbiakban a `curl` programmal könnyen meghívható a szolgáltatás.

Tetszőleges teszt program használható (pl.: Postman, SoapUI, JMeter, Mocha, stb.), ajánlott olyat választani, amely az automatizált tesztelést is támogatja.

Teszt HTTP kérés

```
curl -kv -H "x-request-id: c3c8a9cf-c6ec-4a12-ae30-4edb34e200c9" -H "x-kk-env: INT" -H
"x-kk-gw-ts: 2006014772" -H "x-kk-client-id: _AccessToken sub mezője_" -H "x-kk-sap-
id: 3e017415-3679-48c8-a540-3d4b84f86ec7" -H "x-kk-sap-name: default" -H "x-kk-token-
name: Token1" -H "x-kk-legal-basis-id: dcdf1e82" -H "x-kk-legal-basis-code: _a
szolgáltatásnál értelmezett jogalapkód_" -H "x-kk-security-class: 4" -H "x-kk-access-
token: _jwt kódolt AccessToken_" _szolgáltatás URL-je_
```

A *szolgáltatás URL-je* ebben az esetben a KKSZB nélküli, **valós szolgáltatási végpont** (ahol a szolgáltatás elérhető).

Amennyiben SOAP v1.1 vagy SOAP v1.2 kérést indít, ne feledje el a SOAPAction (v1.1) vagy Action (v1.2) HTTP fejléceket megfelelően megadni, valamint a SOAP szabványnak megfelelően POST kérést indítson.

Példa SOAP v1.1 HTTP kérésre

```
curl -kv -X POST -H "SOAPAction: _action neve_" -H "x-request-id: c3c8a9cf-c6ec-4a12-
ae30-4edb34e200c9" -H "x-kk-env: INT" -H "x-kk-gw-ts: 2006014772" -H "x-kk-client-id:
_AccessToken sub mezője_" -H "x-kk-sap-id: 3e017415-3679-48c8-a540-3d4b84f86ec7" -H
"x-kk-sap-name: default" -H "x-kk-token-name: Token1" -H "x-kk-legal-basis-id:
dcdf1e82" -H "x-kk-legal-basis-code: _a szolgáltatásnál értelmezett jogalapkód_" -H
"x-kk-security-class: 4" -H "x-kk-access-token: _jwt kódolt AccessToken_"
_szolgáltatás URL-je_
```

A fenti parancsra és futásának eredményére egy példa:

Példa HTTP kérésre

```
curl -kv -H "x-kk-env: INT" -H "x-kk-gw-ts: 1506014289" -H "x-kk-client-id:
urn:pid:kkszb:peer1" -H "x-kk-sap-id: b8b2c562-ad29-466a-b35d-65ecf2622abd" -H "x-kk-
sap-name: default" -H "x-kk-token-name: Token1" -H "x-kk-legal-basis-id: dcdf1e82" -H
"x-kk-legal-basis-code: JAR1202A" -H "x-kk-security-class: 4" -H "x-kk-access-token:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjEifQ.eyJqdGkiOiI2MzNhZTYzOC0zYjExLTRhZjY
tYjViNC05ZGExMGRlZWJmMjYiLCJpc3MiOiJ1cm46c3lzOmtrc3piOmZueSIsInN1YiI6InVybjpwaWQ6a2tze
mI6cGVlcjEiLCJ0eXBlIjoidXJuOnRva2VuOmtrc3piOmNsaWVudDphY2Nlc3MiLCJpYXQiOjE0OTMxMTM2NTM
sIm5iZiI6MTQ5MzExMzY1MywiZXhwIjoyMDkzMTE0MjUzLCJzZXJ2aWNlSWQiOiJMMnRyYzNwaUwyVmphRzh2Z
GpFIiwic2VydmljZVVyaSI6Ii9ra3N6Yi9lY2hvL3YxIiwiYXV0aHRva2VuSnRpIjoiOGM2MDM2YzctMjk4Yy0
0ZDE2LTkzMzItYjk4NWYwOGU2YzViIiwic2FwSWQiOiJiOGIyYzU2Mi1hZDI5LTQ2NmEtYjM1ZC02NWVjZjI2M
jJhYmQiLCJzYXBOYW1lIjoiZGVmYXVsdCIsImF1dGh0b2tlbk5hbWUiOiJUb2tlbjEiLCJsZWdhbEJhc2lzSWQ
iOiJkY2RmMWU4MiIsImxlZ2FsQmFzaXNDb2RlIjoiSkFSMTIwMkEiLCJzZWN1cml0eUNsYXNzIjo0LCJ2ZXJza
W9uIjoxfQ.NnzejJv9zdR1zUTE9kYrLJNTVQZZF-5Deq_CmPk0QvKlmmsp-POr_dE5mP13ejZAMUTafyPT-
E8WWhVrAsKNp4zeViCaQNxAXXJpQ8HFrCbM5kdBBNcmzPgDAKj9GxeEXZrXXSllXJOORFrPlTLhcgd6LtjnOok
qNxc4yosc381dztKvdiDyprOt52nmP7XvI76RoaRoTqglT36omyLYOOFPmG31FVNrfv9EcO8qYdH-
WMUldzwn1HEME0NOw7zo8OYuk9PpbhgDQMFK6BFUvwkHOd250Cx0SLCMP7nswpSUMEt2QibKZXWBhmTY60Y44_
Q_j2PLnooR-u3N7dLiWw" http://10.128.1.37/leksz/rest/v1/vallalkozas

*   Trying 10.128.1.37...
* Connected to 10.128.1.37 (10.128.1.37) port 80 (#0)
> GET /leksz/rest/v1/vallalkozas HTTP/1.1
> Host: 10.128.1.37
> User-Agent: curl/7.47.0
> Accept: */*
> x-kk-env: INT
> x-kk-gw-ts: 1506014289
> x-kk-client-id: urn:pid:kkszb:peer1
> x-kk-sap-id: b8b2c562-ad29-466a-b35d-65ecf2622abd
> x-kk-sap-name: default
> x-kk-token-name: Token1
> x-kk-legal-basis-id: dcdf1e82
> x-kk-legal-basis-code: JAR1202A
> x-kk-security-class: 4
> x-kk-access-token:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjEifQ.eyJqdGkiOiI2MzNhZTYzOC0zYjExLTRhZjY
tYjViNC05ZGExMGRlZWJmMjYiLCJpc3MiOiJ1cm46c3lzOmtrc3piOmZueSIsInN1YiI6InVybjpwaWQ6a2tze
mI6cGVlcjEiLCJ0eXBlIjoidXJuOnRva2VuOmtrc3piOmNsaWVudDphY2Nlc3MiLCJpYXQiOjE0OTMxMTM2NTM
sIm5iZiI6MTQ5MzExMzY1MywiZXhwIjoyMDkzMTE0MjUzLCJzZXJ2aWNlSWQiOiJMMnRyYzNwaUwyVmphRzh2Z
GpFIiwic2VydmljZVVyaSI6Ii9ra3N6Yi9lY2hvL3YxIiwiYXV0aHRva2VuSnRpIjoiOGM2MDM2YzctMjk4Yy0
0ZDE2LTkzMzItYjk4NWYwOGU2YzViIiwic2FwSWQiOiJiOGIyYzU2Mi1hZDI5LTQ2NmEtYjM1ZC02NWVjZjI2M
jJhYmQiLCJzYXBOYW1lIjoiZGVmYXVsdCIsImF1dGh0b2tlbk5hbWUiOiJUb2tlbjEiLCJsZWdhbEJhc2lzSWQ
iOiJkY2RmMWU4MiIsImxlZ2FsQmFzaXNDb2RlIjoiSkFSMTIwMkEiLCJzZWN1cml0eUNsYXNzIjo0LCJ2ZXJza
W9uIjoxfQ.NnzejJv9zdR1zUTE9kYrLJNTVQZZF-5Deq_CmPk0QvKlmmsp-POr_dE5mP13ejZAMUTafyPT-
E8WWhVrAsKNp4zeViCaQNxAXXJpQ8HFrCbM5kdBBNcmzPgDAKj9GxeEXZrXXSllXJOORFrPlTLhcgd6LtjnOok
qNxc4yosc381dztKvdiDyprOt52nmP7XvI76RoaRoTqglT36omyLYOOFPmG31FVNrfv9EcO8qYdH-
WMUldzwn1HEME0NOw7zo8OYuk9PpbhgDQMFK6BFUvwkHOd250Cx0SLCMP7nswpSUMEt2QibKZXWBhmTY60Y44_
Q_j2PLnooR-u3N7dLiWw
>
< HTTP/1.1 200 OK
```

```
< Server: Apache-Coyote/1.1
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< Pragma: no-cache
< Expires: 0
< X-Frame-Options: DENY
< X-Application-Context: application:local:8080
< Content-Type: text/plain;charset=UTF-8
< Content-Length: 38
< Date: Thu, 21 Sep 2017 17:36:51 GMT
<
* Connection #0 to host 10.128.1.37 left intact
EVNY-LEKSZ REST WebService v1 (teszt1)%
```

A fenti példában **látható a szolgáltatás válasza** is, ahol a státuszkód `< HTTP/1.1 200 OK`, a válasz tartalma pedig `EVNY-LEKSZ REST WebService v1 (teszt1)%`.

3.8.2. Integrált teszt

A KKSZB rendszerbe kötött szolgáltatás tesztelése integrációs tesztet jelent, amely eltér a fentebb említett teszttől. Ebben az esetben a KKSZB rendszeren keresztül kell megszólítani a Szolgáltatást, **így nem csak a Szolgáltatás, de a KKSZB követelményeit is be kell tartani.**

Az integrációs teszt előfeltételei:

- **KKSZB Integrációs (INT) környezetben létrehozott Szolgáltatás:** ehhez szükséges a KKSZB rendszerhez csatlakozni és utána a Szolgáltatás Létrehozási Kérelem benyújtásával a Szolgáltatást létrehozni a KKSZB (INT) környezetben.
- **működő Szolgáltatás**
- **az alkalmazás szintű tesztek sikeresek**
- a szolgáltatást igénybe vevő, azt elérő (KKSZB rendszerbe kötött) **teszt kliens**
- teszt kliens alkalmazás számára **engedélyezett** Szolgáltatás Elérési Jogosultság alapján kiváltott **érvényes autentikációs token**

A teszt kliens bármilyen HTTP kommunikációra alkalmas teszt program lehet, pl.: SoapUI, curl, Postman, JMeter, mocha. Célszerű automatizált futtatásra alkalmas eszközt választani.

A teszt kliensben a **HTTP headerben mindössze két paramétert kell beállítani:**

- az **érvényes autentikációs tokent** az x-kk-authentication mezőben, ezt az RFNY felületén lehet kiváltani
- a szabványos **x-request-id** mezőt tetszőleges UUIDv4 értékkel

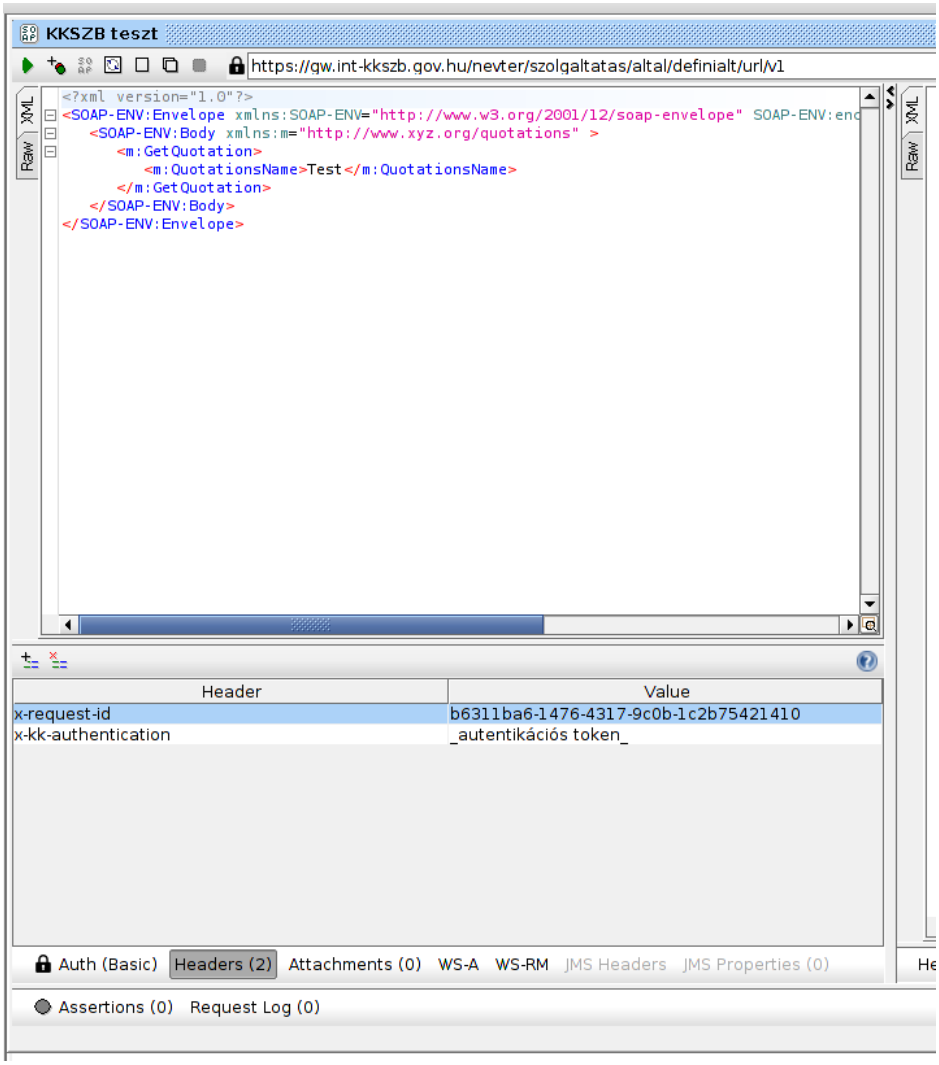
A továbbiakban a KKSZB rendszerben értelmezett **szolgáltatási végpont URL-re,** kell küldeni az összeállított kérést.

Ezek beállítása után bármilyen tetszőleges üzenettartalom küldhető, az alábbi példában a `curl` hívás látható.

HTTP kérés KKSZB végpontra curl segítségével

```
curl -kv -H "x-request-id: b6311ba6-1476-4317-9c0b-1c2b75421410" -H "x-kk-
authentication: _autentikációs token_" https://gw.int-
kkszb.gov.hu/nevter/szolgaltatas/altal/definialt/url/v1
```

Az alábbiakban egy elterjedt, grafikus felülettel rendelkező teszt program beállítása látható. A képernyő alján a HTTP Header két értékét kell megadni, valamint a képernyő tetején az URL-t, valamint értelemszerűen az interfésznek megfelelő SOAP kérést, ha szükséges ne feledjük a SOAPAction és Action értékét is megadni. A SoapUI további beállítási lehetőségeinek leírását itt találja.



Ábra 6. SoapUI tesztprogram beállítás példa

Ne feledje a Szolgáltatás interfésze által megkövetelt további fejléceket beállítani! Például SOAP használata esetén a SOAPAction, vagy az Action mezők értékét, ha szükséges.

Chapter 4. KKSZB Aszinkron Szolgáltatás

A KKSZB Aszinkron Szolgáltatás segítségével a kapcsolódó rendszerek értesítéseket tudnak küldeni egymásnak úgy, hogy a küldő felet nem blokkolja a végrehajtásban.

Az Aszinkron Szolgáltatás (röviden: ASZ) HTTP protokollon keresztül támogatja az aszinkron üzenetküldés (értesítés) funkciót, így könnyen integrálható a modern, webes technológiákat használó környezetekbe. Minden aszinkron rendszer implementáció más és más szolgáltatásokat valósít meg, az ASZ esetén a funkcionalitás az alábbi kritériumok szerint került meghatározásra, ezen kritériumok többsége magából, a KKSZB rendszer céljából is ered.

- **magas rendelkezésre állás:** az ASZ elérhetősége legyen a lehető legmagasabb, hogy ne blokkolja az üzenet küldő szolgáltatás működését.
- **nagy teljesítmény:** az üzenetek minimális késlekedéssel, nagy tömegben legyenek befogadhatók.
- **megbízhatóság:** a feladott üzenetek nyomon követhetők legyenek, biztonságban tárolódjanak - pl.: ne sérüljön, ne vesszen el -, és megbízhatóan jussanak el az azt felhasználó végpontra.
- **architektúra tervezés segítése:** a kialakított funkciók átgondolt kompozíciójával tetszőleges kommunikációs architektúrák legyenek kialakíthatóak.

Az előzőekben a legfontosabb *termékkel szembeni szempontok* kerültek definiálásra, a további, *általánosan érvényes informatikai szempontokat* - skálázhatóság, bővíthetőség, tesztelhetőség, stb. -, külön nem soroljuk fel.

A KKSZB Aszinkron Szolgáltatás a teljesítmény és a magas rendelkezésre állás miatt nem garantálja az üzenetek sorrendben történő kézbesítését, ezt az interfész tervezéskor figyelembe kell venni.

A KKSZB Aszinkron Szolgáltatás célja, hogy a *küldő fél* az üzenetet akkor is fel tudja adni, amikor a *fogadó fél* nem elérhető, vagy lassan dolgozza fel az üzenetet, ezzel biztosítva azt, hogy a *küldő fél* oldalán a folyamat azonnal folytatható legyen, azt a *fogadó fél* állapota ne befolyásolja, ne blokkolja. **A KKSZB Aszinkron Szolgáltatásnak nem célja a *fogadó fél* teljesítmény problémájának megoldása és az üzenetek hosszútávú perzisztens tárolása - az adatokat csak átmenetileg perzisztálja, az üzenet kézbesítés céljából.**

4.1. A KKSZB Aszinkron Szolgáltatás használatának előnyei

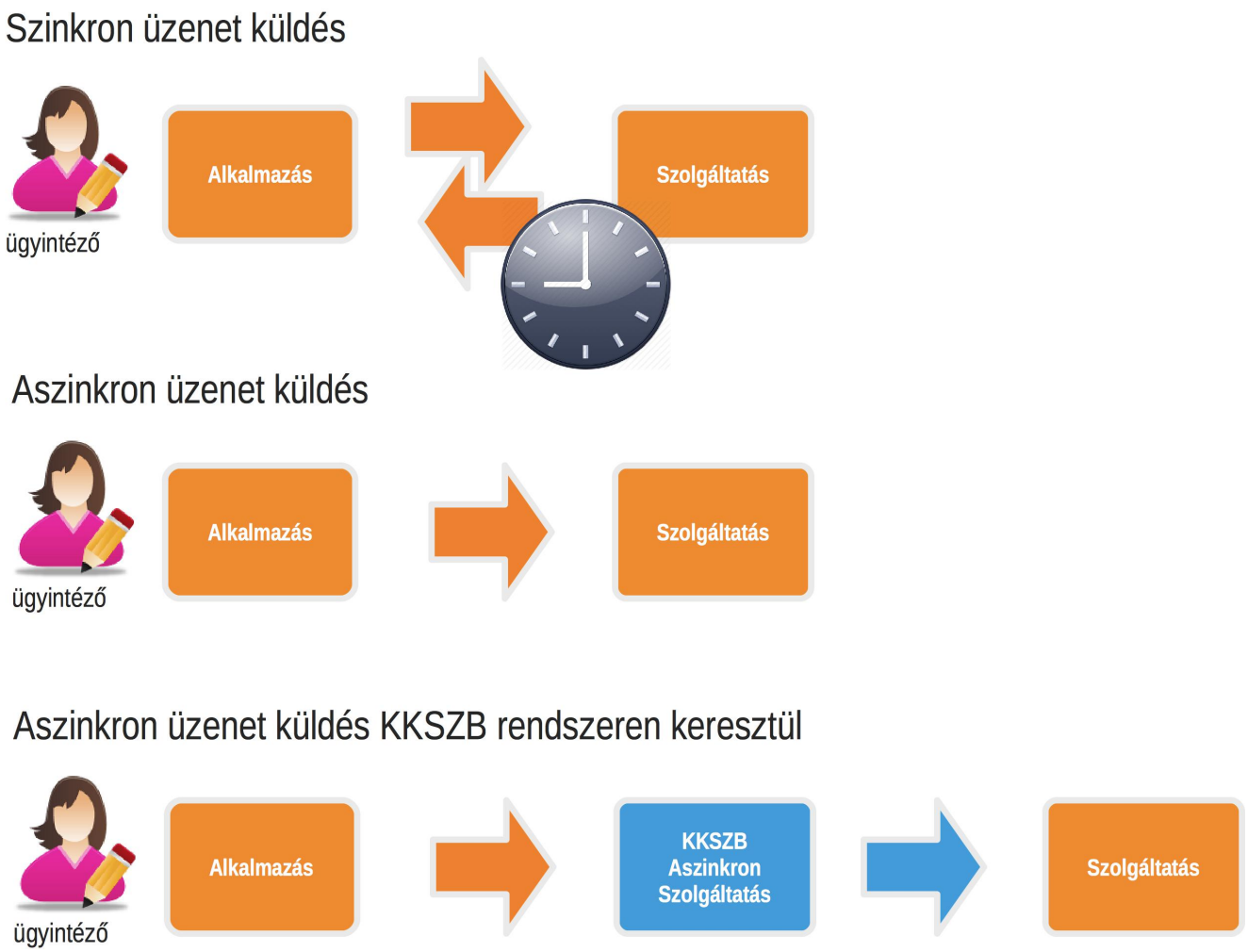
A szolgáltatás használata a következő előnyökkel jár egyéb, egyedi termékkel szemben:

- **Standard kommunikációs protokoll használata:** az elérés HTTP alapú, így platformfüggetlen módon, bármely informatikai környezetből standard web szolgáltatásként érhető el az alkalmazás.
- **Transzparens aszinkron szolgáltatás:** az üzenet küldő félnek ugyanúgy kell küldeni a HTTP

kérésben az üzenetet, mintha közvetlenül a szolgáltatásnak küldené szinkron módon.

- **Központilag, a KKSZB által biztosított szolgáltatás jelentős megtakarítással jár:** nem kell külön hardvert üzemeltetni, nem szükséges üzemeltető személyzetet kiképezni és alkalmazni, nem kell rendszeres licenc díjat fizetni, nem kell a rendelkezésre állásról és mentésről gondoskodni.
- **Platform függetlenséget biztosít:** a kapcsolódó feleknek nem kell beszerezniük és üzemeltetniük ugyanazt az egyedi terméket (többszörös költségek), elégséges csak kapcsolódniuk a KKSZB rendszerhez és bármilyen szoftverplatformról (pl.: Java, NodeJS, .Net, Perl, PHP, stb) tudják a szolgáltatást használni.
- **Egyszerűsíti az architektúrát:** nem szükséges minden kliens számára saját queue-t definiálni, hanem egy aszinkron szolgáltatási végponton keresztül kezelhető az összes bejövő üzenet.

Az alábbi ábrán látható, hogy az ügyintéző *szinkron üzenetküldés* esetén meg kell, hogy várja a választ. Az olyan esetekben, ahol ez jelentős késlekedéssel jár, vagy üzletileg nem használható, ott aszinkron üzenetküldést kell alkalmazni.



Ábra 7. Szinkron és Aszinkron üzenetküldés

A KKSZB Aszinkron szolgáltatása transzparens módon jelenik meg (kék ábra), mert az adott alkalmazás ugyanúgy éri el a KKSZB rendszerben a szolgáltatást, mintha magát a szolgáltatást hívná.

A KKSZB Aszinkron Szolgáltatásról a transzparens viselkedése és a HTTP protokoll miatt könnyen lehet átállni szinkron üzenetküldésre.

4.2. Mikor érdemes igénybe venni a KKSZB Aszinkron Szolgáltatást?

A KKSZB által biztosított Aszinkron Szolgáltatás használata az alábbi esetekben javasolt:

- platform semleges megoldás szükséges
- az üzleti logika nem igényli az üzenetek sorrendbe állítását - megoldható a sorrendbe állítás, de erről az üzenet *küldő és fogadó félnek* kell gondoskodnia a megfelelő interfész kialakításával
- költségcsökkentés céljából
- ha már KKSZB-re csatlakozott az adott szervezet
- gyorsan kell kialakítani az aszinkron szolgáltatást
- magas rendelkezésre állású, nagy teljesítményű üzenetküldés a cél
- architektúra egyszerűsítése a cél: több queue helyett egy aszinkron szolgáltatási végpont

Az alábbi esetekben nem ajánlott az Aszinkron Szolgáltatás használata:

- saját, szervezeten belül, egy adott rendszerben futó alkalmazáskomponensek közötti kommunikációban
- *PULL queue* használata esetén, ekkor a *fogadó fél* viszi el az üzenetsorban található adatokat

A gyors és megbízható működés érdekében lehetőleg soha ne használjon aszinkron üzenetküldést, amennyiben a válasz üzenetet a küldő félnek azonnal meg kell kapnia. Tipikusan ilyen esetek az adatlekérdezések.

A *fogadó fél* oldali adatlekérdezési teljesítmény problémák megoldására az aszinkron kommunikáció csak jelentős kompromisszumok árán alkalmas, amelyek:

- a szolgáltatás minőségromlása - tipikusan megnövekedett válaszidő és kérés timeout
- jelentős fejlesztési többlet ráfordítás, és emiatt megnövekedett hibalehetőség *küldő és fogadó fél* oldalon

A fentiek miatt más architekturális elem bevezetése javasolt adatlekérdezések esetén, például **Szolgáltatás oldali adat gyorsítótárazás,** amely jelentősen növeli a teljesítményt.

4.3. Szolgáltatás által nyújtott megoldások és kialakításuk

Az ASZ olyan megoldásokat nyújt, amelyek a gyakori informatikai problémákra adnak választ, mindezt úgy, hogy az alap szolgáltatás készletből tetszőleges komplex kommunikációs architektúra

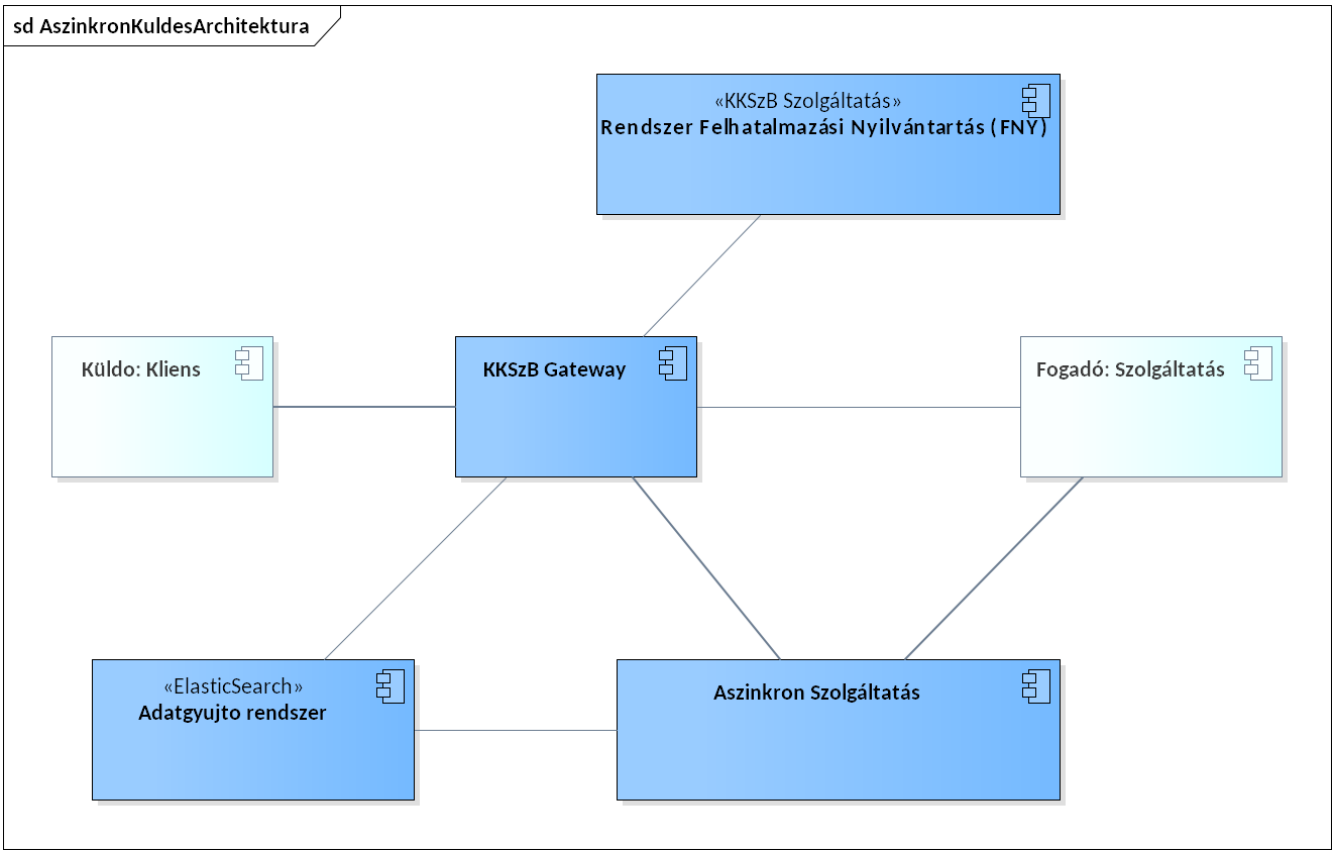
is felépíthető.

Az alábbiakban az üzenet küldő (*producer*, röviden: küldő fél) és az üzenet fogadó felek (*consumer*, röviden fogadó fél) közötti kommunikációs mintákat mutatjuk be.

A *küldő fél* mint KKSZB kliens alkalmazás csatlakozik, a *fogadó fél* pedig KKSZB szolgáltatásként jelenik meg, ahol a szolgáltatás egy *aszinkron* szolgáltatási végpont.

Az Aszinkron Szolgáltatás nem *tartós perzisztencia* réteg, **csak átmeneti ideig - az üzenet lejárati időpontig, vagy a sikeres kézbesítésig - perzisztálja az adatokat a háttértáron.**

Az aszinkron szolgáltatási végpont standard HTTP végpont a KKSZB rendszerben.



Ábra 8. KKSZB Aszinkron Szolgáltatás magas szintű architektúrája

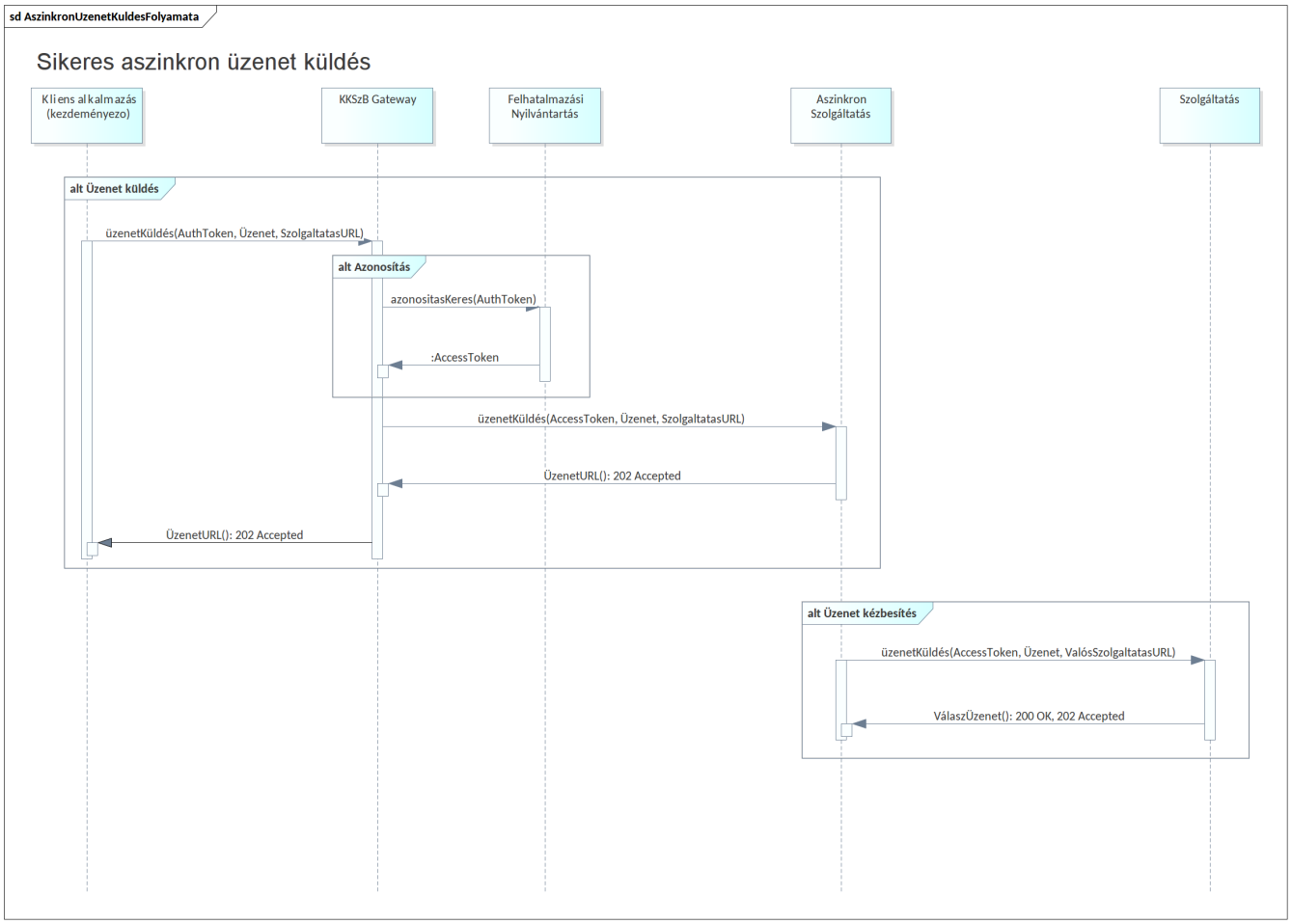
A KKSZB Aszinkron Szolgáltatása nem naplózza az üzeneteket, csak a kommunikációra vonatkozó adatokat.

A KKSZB ASZ önmagában nem váltja ki az alkalmazás-alkalmazás közötti aszinkron üzenetküldést, hiszen az interfésznek és használatának figyelembe kell vennie az *aszinkron HTTP kommunikáció* sajátosságait és ennek megfelelően kell tervezni a működést és elvégezni implementálást.

A KKSZB ASZ használata esetén a **tranzakció tervezést** az elosztott rendszerek tranzakciókezelése és a Tervezési segédlet fejezetben foglaltak figyelembe vételével kell elvégezni.

4.4. Értesítő üzenet kezelésének áttekintése

Az ASZ úgy lett kialakítva, hogy az értesítéseket ugyanúgy kell küldeni, és ugyanúgy kell fogadni, mintha WebService kérést küldenénk. **Ezzel a megoldással sem a küldő, sem a fogadó oldalon nem kell eltérni a hagyományos WebService (HTTP) fejlesztési módszerektől,** így a fejlesztés gyorsan, jól tesztelhető módon - ugyanúgy mint a szinkron WebService fejlesztés esetén -, elvégezhető.



Ábra 9. KKSZB Aszinkron Szolgáltatás sikeres üzenetküldési folyamata

A küldő fél egyszerű HTTP kérést küld az ASZ felé, amely befogadja az üzenetet, ezzel le is zárult a küldő oldali feladat. Az ASZ ezt az üzenetet egyszerű HTTP kérésként küldi tovább a fogadó fél felé (*push*), aki az üzenetet befogadja, ezzel az üzenet kézbesítetté válik és az ASZ-ből törlődik.

Az Átviteli hibakezelés fejezetben leírtak szerint kell a küldő és fogadó félnek kezelnie az átviteli hibákat.

A HTTP üzenetküldéskor a KKSZB rendszerre vonatkozó csatlakozási szabályokat be kell tartani, amelyet a Szinkron kommunikáció kezelése kliens alkalmazás esetén fejezet definiál.

Az ASZ az üzenetet befogadja, majd azonnal megpróbálja kézbesíteni. Sikertelen kézbesítés után ismételten megpróbálja eljuttatni az üzenetet a *fogadó fél* végpontjára.

A *fogadó fél* oldalán be **kell** fogadni az üzenetet és a befogadás után kell elkezdeni a feldolgozást.

Az üzenetek **újraküldhetőségéről** a *küldő félnek* célszerű gondoskodnia, hogy a *fogadó fél* oldalon esetlegesen felmerülő feldolgozási hibákat (programhibákat) kivédhesse.

A *fogadó fél* oldalon úgy kell felkészülni, hogy **ugyanazon üzenet akár többször is megérkezhet,** akár más egyedi azonosítóval is, így a *küldő fél* oldalán felmerülő, ismétlődő üzenetküldési hibák kezelhetők.

4.4.1. Feldolgozás eredményének kezelése

A feldolgozás eredményéről - a *fogadó fél* által megadott Szolgáltatás interfész leírásnak megfelelően - *vagy kell válasz üzenetet küldeni vagy nem.*

Amennyiben kell válaszüzenetet küldeni (pl.: sikertelen feldolgozás), úgy a Szolgáltatás interfész leírásnak megfelelően kell eljárni, amely igény szerint tetszőleges lehet, az alábbiakban a leggyakoribbakat soroljuk fel:

- A Szolgáltatás interfész leírása szerint igényli, hogy a Kliensnek legyen egy *aszinkron* KKSZB végpontja, ahová aszinkron módon a *feldolgozás eredményét* elküldheti - lásd Kétirányú értesítés fejezet.
- A *fogadó fél* biztosít egy másik, *szinkron* KKSZB végpontot (Szolgáltatást), ahol a Kliens által bármikor lekérdezhető a feldolgozás eredménye vagy maga az adat (polling).
- A Szolgáltatás interfész leírása szerint igényli, hogy a Kliens oldal rendelkezzen egy *szinkron* KKSZB végponttal, amelyet meghívhat a *feldolgozás eredményével.*

A fenti módszerek kiválasztásakor mindig a rendszer (fogadó és küldő fél együtt) igényeinek megfelelően kell dönteni, gondos elemzés után.

4.4.2. Aszinkron Szolgáltatással kialakítható minták

A KKSZB ASZ használatával tetszőleges, összetett architektúrák alakíthatóak ki. A legkisebb "egység" az **egyirányú értesítés,** amely a minimális aszinkron rendszerekre jellemző működést definiálja. A leggyakrabban használt, aszinkron rendszerekre jellemző működés a **kétirányú értesítés,** amely az adatkarbantartások rendszer-rendszer közötti tipikus megvalósítását definiálja.

A KKSZB rendszer *push queue* megoldást használ, amelyben a *fogadó fél* az adatokat úgy kapja meg, mintha HTTP hívás történne, így a fogadó oldalon semmilyen speciális, gyártófüggő kód felhasználására vagy beépítésére nincs szükség.

Egyirányú értesítés

A *küldő fél* egy üzenetet küld az ASZ felé és semmilyen formában nem érdekli a *fogadó fél* üzenetre adott válasza.

Magyarázat: ebben az esetben a *küldő fél* nem akarja megismerni a *fogadó fél* válaszának eredményét, még akkor sem, ha az hibaüzenetet jelent.

Tipikus használati esetei az üzemeltetési állapot adatok rendszeres küldése, a tömeges e-mail küldés vagy az adatváltozásokról történő értesítés több *fogadó fél* felé.

Kétirányú értesítés

A *küldő fél* egy üzenetet küld az ASZ felé, amelyre a *fogadó fél* válaszol szintén egy értesítés típusú üzenettel.

Magyarázat: ebben az esetben a *küldő fél* visszaigazolást vár a *fogadó féltől* a feldolgozás eredményéről. A feldolgozás eredményére adott válasz az interfész jellegétől függhet, például csak a hibás feldolgozások esetén kap értesítést a *küldő fél.*

Tipikus használati esete a kétirányú adatkarbantartások rendszerek között.

A kétirányú értesítések esetén mindkét fél *küldő és fogadó fél* is egyben, így mindkét oldalnak mint kliens és mint aszinkron szolgáltatási végpont kell csatlakoznia a KKSZB rendszerre.

A *küldő fél* által létrehozott aszinkron szolgáltatási végpont fogadhatja a beküldött üzenetek feldolgozási eredményét, de ezen interfész kialakítása a kommunikációs felek egymás közötti megállapodásától függ.

Természetesen bármilyen, tetszőleges architektúra kialakítható mind aszinkron, mind szinkron üzenetküldési módszerekkel, lásd a korábbi Feldolgozás eredményének kezelése fejezetben.

4.4.3. Kialakítható aszinkron architektúrákra példák

Az alábbiakban az aszinkron üzenetküldés által kialakítható architektúra megoldások leírását adjuk meg. Ezek mint minta kezelendők, természetesen ettől eltérő, igény szerinti megoldások is kialakíthatóak.

1. példa: **értesítés adatkarbantartásról;** a küldő fél adatokat közöl a fogadó féllel, aki azt feldolgozza és a feldolgozás eredményéről válasz üzenetet küld. Tipikus használati eset, amely a **kétirányú értesítés** mintát használja. Mindkét fél egyben küldő és fogadó fél is, így mindkettő félnek kell egy-egy *aszinkron szolgáltatási végponttal* rendelkeznie a KKSZB rendszerben, ahová a másik kapcsolódó fél az üzeneteket küldi.
2. példa: **értesítés adatkarbantartásról több fél részére (*broadcast*);** ekkor a küldő fél ugyanazt az üzenetet több fogadó fél részére is elküldi. A küldő fél ebben az esetben nem biztos, hogy értesülni akar a fogadó fél feldolgozásának eredményéről, így a küldő félnek nem szükséges ekkor a KKSZB rendszerben saját - a válaszüzenetek fogadására alkalmas - aszinkron szolgáltatási végponttal rendelkeznie. Tipikus használati eset, amely az **egyirányú értesítés** mintát használja.

A KKSZB jellegéből fakadóan szigorúan megköveteli a legmagasabb fokú

biztonságot, ezért a hagyományos *publish-subscribe* típusú modellben értelmezett feliratkozás csak az RFNY rendszerben történő szolgáltatás elérési jogosultság engedélyezésével történik.

Mivel a KKSZB nem *pull queue,* hanem *push queue* elvek mentén működik, így a feliratkozási logika épp megfordul: nem az adatfelhasználó - *fogadó fél* - iratkozik fel az adatok fogadására, hanem a *küldő fél* igényli, hogy adott végpontra küldhesse az adatokat.

A fentiekből következik, hogy egy aszinkron szolgáltatási végpontra akár több kliens is tud üzenetet küldeni, de akár kliensenként is kialakítható több szolgáltatási végpont is. Az adminisztráció csökkentése végett javasolt egy aszinkron szolgáltatási végpontot definiálni és a kliensek szolgáltatás elérési jogosultságát ezen kezelni.

A fejlesztés és tervezés támogatásához a Tervezési segédlet fejezetben találhatók minták.

4.4.4. Kézbesítési algoritmus

Kézbesítési algoritmus alatt azt a logikai folyamatot értjük, amely alapján az üzenet kézbesítésre kerül a *fogadó fél* számára az ASZ rendszerből, valamint kézbesítési hiba esetén annak kezelését jelenti.

A Kézbesítési algoritmus transzparens az üzenet kezelő felek számára, azt a KKSZB ASZ önállóan hajtja végre, jelen leírás csak tájékoztató jellegű.

Az alábbi logika mentén történik a kézbesítés:

- az üzenet sikeres befogadása után kézbesítésre kerül
- amennyiben hibás a kézbesítés, úgy ismételten, **többször** megpróbálja az ASZ kézbesíteni az üzenetet, a *üzenet lejárat időpontjáig* (lásd későbbi fejezetben).
- amennyiben a sikertelen kézbesítés után az *üzenet lejárat időpontjában* ismételten sikertelen a kézbesítés, úgy utána az üzenet törlésre kerül, továbbiakban a kézbesítés már nem ismétlődik
- sikeresen kézbesített üzenet többé nem kerül kézbesítésre, törlődik

Az üzenet sikeres kézbesítése nem jelenti, hogy a *fogadó fél* a feldolgozást elkezdte vagy elvégezte, csupán azt jelenti, hogy az ASZ sikeresen átadta az üzenetet a *fogadó fél* részére.

4.4.5. Átviteli hibakezelés

Az értesítés típusú üzenetek esetén legalább három szereplő kapcsolata szükséges az üzenet célba juttatásához: *küldő fél*, Aszinkron Szolgáltatás és a *fogadó fél.* Emiatt nem csak a *fogadó fél* által jelzett hibákat (pl.: szintaxis-, validálási-, vagy belső alkalmazás hibákat), hanem az Aszinkron Szolgáltatás eseményeit is kezelni kell a *küldő és fogadó félnek (ASZ interfész kezelés).*

Az Aszinkron Szolgáltatás által létrehozott hibajelzéseket *átviteli hibáknak* nevezzük, elkülönítve a *fogadó fél* hibaüzeneteitől. Az átviteli hibák technikai jellegű hibák, míg a *fogadó fél* üzenetei üzleti és technikai jellegűek.

Átviteli hibák az alábbi esetekben fordulhatnak elő:

- **Az ASZ nem érhető el:** a KKSZB kapcsolati problémát ekkor ki kell vizsgálni, mert a *küldő fél* működését az adott funkció esetén akár gátolhatja is.
- **Az ASZ elérhető, de nem 202 Accepted státuszkóddal válaszolt**: az ASZ nem vette át sikeresen az üzenetet. Ebben az esetben a hiba státuszkódjának, és a *küldő fél* üzleti logikájának függvényében a *küldő fél* eldönti, hogy az üzenetet újraküldi-e. Egyes hibakódok kezelését a dokumentáció további részében külön is tárgyaljuk.
- **Az ASZ adott időn belül nem válaszolt 202 Accepted státusz kóddal**: itt előfordulhat, hogy az üzenet az ASZ-be be sem került, de az is, hogy az üzenetbefogadás sikeresen megtörtént, csak a 202 Accepted státuszkód nem jutott el a *küldő félhez*. A lényeg, hogy az üzenet státusza nem állapítható meg. Ez esetben a *küldő fél* üzleti logikája határozza meg, hogy az üzenetet újra kell-e küldeni az ASZ felé.
- **A *fogadó fél* nem érhető el:** ekkor a *küldő fél* feladja az üzenetét, de az üzenettovábbító rendszer (ASZ) nem éri el a célrendszert (*fogadó felet*). Ebben az esetben az üzenethez bejegyzésre kerül az átviteli hiba és adatai, a kézbesítést ismét megpróbálja az ASZ.
- **A *fogadó fél* elérhető, de nem *200 OK* vagy *202 Accepted* státuszkóddal válaszolt:** a *fogadó fél* nem vette át sikeresen az üzenetet. Ebben az esetben az üzenethez bejegyzésre kerül az átviteli hiba és adatai, a kézbesítést ismét megpróbálja az ASZ.
- **A *fogadó fél* adott időn belül nem válaszolt *200 OK* vagy *202 Accepted* státusz kóddal:** az ASZ rendszer maximum 2 percig (üzenetküldés timeout) várakozik a fogadó féllel történő kommunikáció során. Amennyiben a feldolgozás nem fejeződik be ennyi idő alatt, úgy a kapcsolat bontásra kerül. **A fogadó félnek az ilyenkor keletkező kommunikációs hibajelzést (kapcsolat megszakadása) és az ASZ felől az ismételt üzenetküldést kezelnie kell.** Ebben az esetben az üzenethez bejegyzésre kerül az átviteli hiba és adatai, a kézbesítést ismét megpróbálja az ASZ.

4.5. Aszinkron Szolgáltatás üzenetkezelése

Az ASZ két interfészt definiál: egyet a *küldő fél* részére és egyet a *fogadó fél* részére.

A HTTP protokoll esetén a HTTP szabvány szerinti üzenetkódokat használja a rendszer.

Az üzenet *küldő fél* mindig HTTP POST üzenetként küldi el a HTTP kérését, amelyet az üzenetet *fogadó fél* ugyanígy HTTP POST kérésként kap meg.

Tetszőleges HTTP üzenet tartalom küldhető, de a tartalomra vonatkozó *Content Type* értékét meg kell adni.

Amennyiben nem HTTP POST üzenet lett küldve, úgy a **405 Method Not Allowed** üzenettel válaszol az ASZ.

4.6. Üzenet élethossza

A KKSZB rendszer Aszinkron Szolgáltatása nem tárolja örökké az adatot, ezért mindenképpen van az üzenetek megtartásának egy maximális ideje, amit *üzenet lejárat időpontnak* hívunk.

Az *üzenet lejárat időpontja* elteltével az üzenet automatikusan törlődni fog az ASZ adatbázisából függetlenül attól, hogy sikeres vagy sikertelen volt az üzenetküldése.

Az *üzenet lejárat időpontja* a KKSZB rendszerben 7 nap, amely az ASZ-be történő befogadásától számított.

4.7. Üzenetküldés oldali interfész - küldő fél

Az üzenetküldés oldali interfész az alábbi funkciót biztosítja az *küldő fél* számára:

- üzenetküldése (üzenet létrehozás)

A beküldött üzenet *lekérdezésére* nincs lehetőség, a rendszer azt kézbesíti és automatikusan törli. A beküldött üzenet *törlésére* sincs lehetősége a *küldő félnek,* mert a rendszer azt azonnal kézbesíti (kivétel, ha a túloldal ezt nem tudja fogadni) - a beküldött üzenet úgy tekintendő, mintha a *fogadó fél* azt *befogadta* volna.

A *befogadás* nem egyezik meg a *feldolgozással.*

4.7.1. Üzenetküldés (üzenet létrehozás)

Az üzenetet a szolgáltatás URL címére kell küldeni, ugyanúgy mint a szinkron üzenetküldés esetén, a Szinkron kommunikáció kezelése kliens alkalmazás esetén fejezet szabályait kell betartani.

Sikeres üzenetküldés esetén mindig *202 Accepted* választ kap a *küldő fél,* melynek jelentése: üzenet befogadva ASZ által, de nincs kézbesítve. A válasz üzenet az Aszinkron Szolgáltatás válasz függelékben foglaltak szerint lesz felépítve.

Az üzenetküldés standard HTTP kérés műveletként történik *POST* üzenettel, ahol a *Content Type* értéke *tetszőleges.* A Content Type érték meghatározása az adott interfész feladata. Egy csatornán (szolgáltatás végponton) keresztül akár többféle Content Type is kezelhető.

A Content Type értéke lehet akár *multipart/form-data* is, amely több állomány átvitelét teszi lehetővé egyszerre, a szabványról bővebben: https://www.w3.org/TR/html401/interact/forms.html#h-17.13.4

Az ASZ az alábbi HTTP fejléceket (nem kisbetű, nagybetű érzékenyen) kézbesíti a *fogadó félnek:*

- az összes **x-kk-** kezdetű fejlécet, csak a KKSZB helyezhet el ilyet az üzenetben, az szinkron üzeneteknek megfelelően,
- **x-kk-message-id** az üzenet egyedi azonosítója, amelyet a KKSZB ASZ képez (messageId)

- a standard HTTP **Content-Length, Content-Type és X-Request-ID** fejlécet
- a standard HTTP **Accept-** kezdetű és **Accept** fejlécet
- a SOAP specifikus **SOAPAction** (SOAP 1.1) és a Content-Type HTTP fejléccel együtt az **action** (SOAP 1.2) fejlécet

A küldő fél felelőssége a standard HTTP fejlécek megfelelő, a szolgáltatás és HTTP szabvány által elvárt módon való töltése, kivétel az x-kk- kezdetű fejlécek esetén.

Ismételt kérés kezelése ASZ-ben: a KKSZB rendszerben használt egyedi üzenet azonosító az **x-request-id**, ismételt felhasználásakor az üzenet ismét rögzítve lesz és ismét ki lesz kézbesítve, új **messageId** értékkel.

Amennyiben a válasz 5xx státuszkódú, úgy az üzenet küldést ismételten meg kell próbálni.

A feladható maximális üzenet hossz 10MB. A 10 MB az a HTTP *body* tartalmát jelenti, amely nagyobb lehet, mint maga a küldő fél által értelmezett adat mérete. Például a *multipart/form-data* kódolás esetén a part-ok adata is bekerül az üzenetbe (HTTP *body*), így egy pontosan 10 MB-os adat nagyobb *body* méretet eredményez.

Maximális üzenet hossz elérésekor az átvitel megszakad, és **413 Payload Too Large** választ küld vissza a szerver PAYLOAD_TOO_LARGE üzenetkóddal.

Sikeres kézbesítés után az üzenet törlésre kerül. Az *üzenet lejárat időpontja* után az üzenet akkor is törlésre kerül, ha sikertelen a kézbesítés.

4.7.2. Beküldött üzenet törlése

Az üzenet törlés automatikusan megtörténik a sikeres kézbesítéskor, de legkésőbb az *üzenet lejárati időpontjában.*

Az üzenet törlése végleges, ha szükséges, akkor az üzenetek tartós perzisztenciájáról a küldő és/vagy fogadó félnek kell gondoskodnia!

4.7.3. Tranzakciókezelés

A KKSZB Aszinkron Szolgáltatása esetén a küldő félnek a *távoli elosztott tranzakciókezelés* szabályainak megfelelően kell eljárnia.

A hagyományos *Message Queue (MQ)* alkalmazások a *küldő fél* felügyelete alatt állnak és az adott szerver zónán belül közel helyezkednek el a küldő alkalmazáshoz. Ebben az esetben *gyártó függő* kódok és megoldások kerülnek az alkalmazásba, leggyakrabban a *driver* és használatával összefüggően. Ilyenkor az alacsony hálózati késleltetés miatt, valamint az egyedi, gyártó specifikus megoldástól függően a tranzakciót a hagyományos tranzakció-kezelő szoftverekkel lehet kezelni.

Mivel az ASZ webszolgáltatásként érhető el, így itt ez a megoldás nem alkalmazható. Ugyanakkor nagyon egyszerűen megvalósítható a tranzakciókezelése, néhány szabály betartásával.

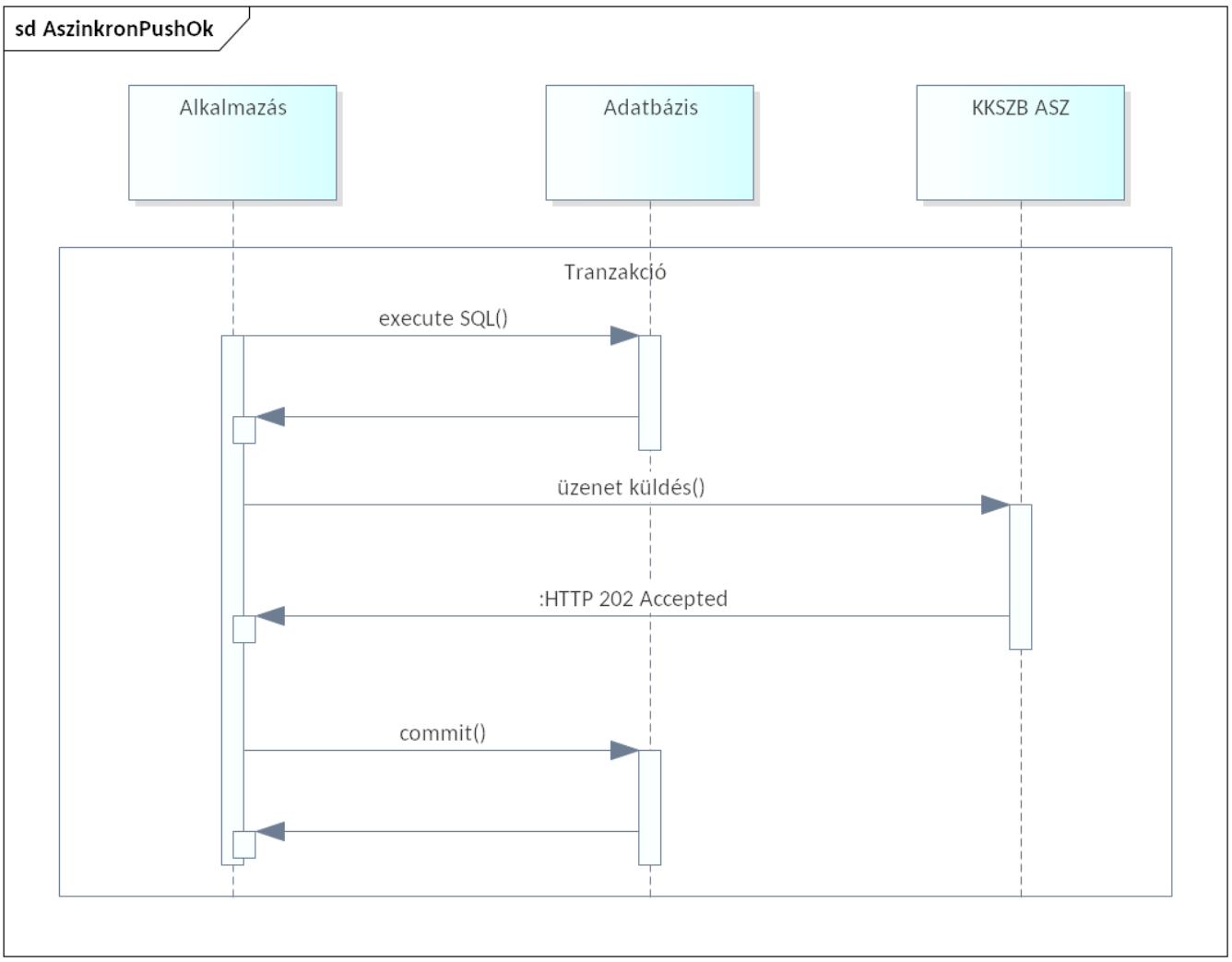
Az ehhez szükséges leírások a Tervezési segédlet fejezetben találhatók.

Egyszerű tranzakciókezelés

Az egyszerű tranzakciókezelés esetén az *küldő fél* az alkalmazásában végrehajtja a saját, lokális környezetében kezelt erőforrásokon a műveleteket (*execute SQL*), majd a *commit* fázis megkezdése előtt *elküldi az üzenetet* az ASZ rendszernek.

Amennyiben az ASZ rendszertől sikeresen megkapja a válaszüzenetet *HTTP 202 Accepted* státusz kóddal, úgy folytatja a műveletét a *commit* fázissal.

Az alábbi ábrán a sikeres átvitel esetén végbemenő folyamatra mutatunk példát. Természetesen nem csak egy, hanem tetszőleges számú erőforrás használatával is ugyanígy lehet eljárni.



Ábra 10. KKSZB Aszinkron Szolgáltatás sikeres üzenetküldés esetén

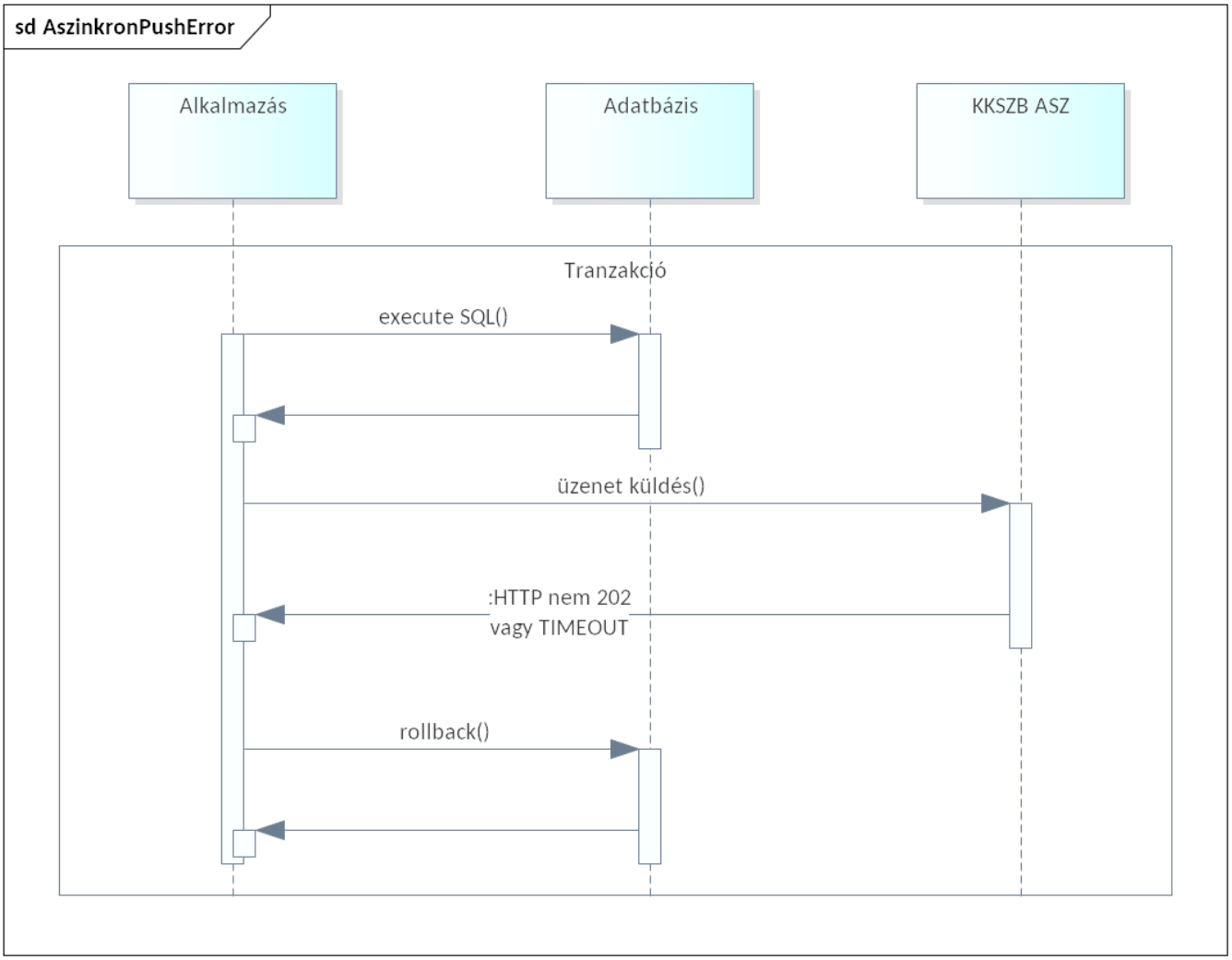
Megjegyzendő, hogy a *commit* fázis alatt is lehetnek hibák, amely azonban független probléma az ASZ rendszer használatától, ehhez ajánlott az egy, két és három fázisú kommitok kezelésével tisztában lenni.

Amennyiben az ASZ felé küldött üzenetre nem *HTTP 202 Accepted* státuszkód a válasz vagy egyéb

hiba (timeout, kapcsolat megszakadás, stb.) történt, akkor az ezután következő tranzakciókat vissza lehet görgetni.

Az erőforrások tranzakciókezelése, mindig az adott alkalmazás igényétől függ.

Az alábbi ábrán egy sikertelen üzenetküldés esetén történő folyamat látható.



Ábra 11. KKSZB Aszinkron Szolgáltatás sikertelen üzenetküldés esetén

Természetesen tetszőleges módon, igény szerint alakítható ki a tranzakciókezelése, azt az interfész definiálja saját igényei szerint, például *kompenzáló tranzakció* kezeléssel az elosztott, független rendszerek között.

A *kompenzáló tranzakciókról* bővebben itt olvashat:

- Wikipedia - Compensating transaction
- Microsoft(TM) - Compensating Transaction pattern

Az egyszerű tranzakciókezelés nem garantálja a rendszerek közötti konzisztenciát, bővebben erről a Tervezési segédlet fejezetben olvashat.

Garantált üzenetküldés

A garantált üzenetküldés esetén az üzenet a küldő és fogadó fél oldalán is perzisztensen tárolásra kerül. A gyakorlatban ezek a feldolgozás után is megőrzésre kerülnek, naplózás és hibakeresés céljából.

Bővebben a garantált üzenetküldésről: Garantált üzenetküldési pattern

Bár a *garantált üzenetküldés* feltételezi, hogy üzenet nem veszhet el, ez nem jelenti azt, hogy az üzenetet a *fogadó fél* helyesen dolgozza fel (programhibák). Ezért ajánlott a *küldő fél* oldalán az elküldött üzeneteket perzisztálni, vagy az üzenet előállítás reprodukálhatóságát biztosítani.

Jó gyakorlat az is, ha a *fogadó fél* oldalán az üzenetek első lépésben azonnal, perzisztensen mentésre kerülnek és csak a biztos mentés után történik meg a feldolgozásuk (garantált átvitel ezt kívánja meg). Így az egyszerű üzenetbefogadó programkód és a jóval bonyolultabb üzleti feldolgozás programkódja különválik, és egy esetleges üzleti feldolgozásban keletkező hiba nem befolyásolja az üzenet biztonságos tárolását. Az esetleges hiba kijavítása után az üzenet így, a *fogadó oldalon* újra feldolgozható, most már sikeresen.

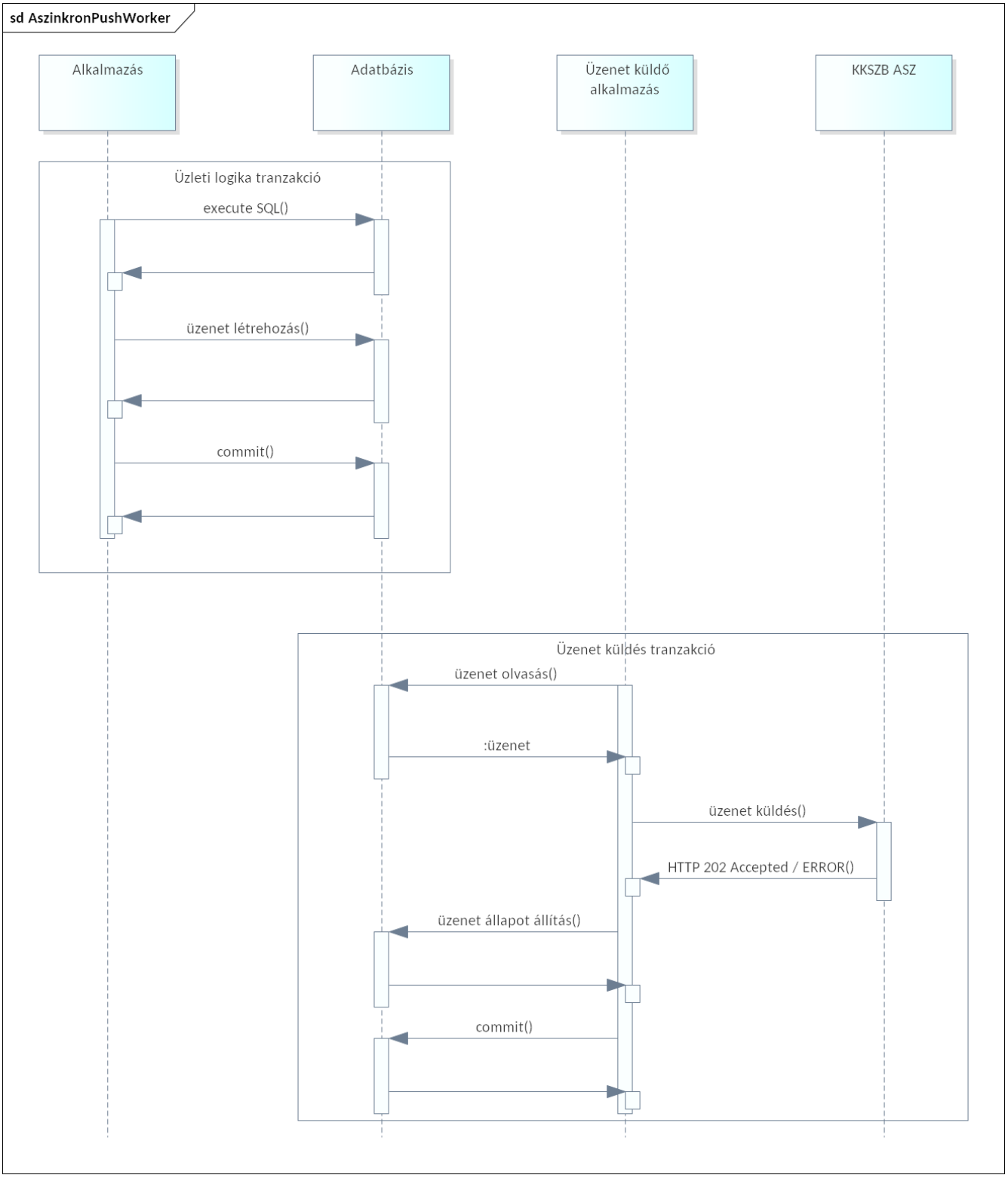
Az alábbiakban bemutatunk egy példát, ahol az *üzleti tranzakció* függetlenítve van (nem blokkolt) az *üzenetküldés tranzakciójától,* így az alkalmazás, amely az üzleti műveletet végzi, **még akkor is végre tudja hajtani a műveleteit, ha éppen hálózati probléma miatt nem éri el a KKSZB rendszert.**

A hálózati kapcsolat helyre állása után *végülis konzisztensé (eventually consistent) válik* a *küldő fél* és a *fogadó fél* oldala, mert az üzenet kézbesítésre és feldolgozásra kerül.

Az ASZ-hez hasonló rendszereket *eventually consistent* kategóriába sorolják, amelyről bővebben itt olvashat: Data Consistency Primer, Eventually consistent fejezet

Az alábbi ábrán egy adott eseményre - például időzítés hatására - elinduló *üzenet küldő alkalmazás* oldja meg az üzenet kézbesítését, amely a kézbesítés eredményének megfelelően az elküldött üzenethez bejegyzést tesz.

A megoldás előnye, hogy a küldendő üzenet bármikor újraküldhető, vagy megtekinthető - naplózás, hibakeresés, kézi újraküldés céljából - mivel a *küldő fél* oldalán perzisztensen tárolásra kerül.



Ábra 12. KKSZB Aszinkron Szolgáltatás: nem blokkoló üzenetküldés

A garantált üzenetküldés biztosítja a legnagyobb valószínűséggel a konzisztenciát a rendszerek között, bővebben erről a Tervezési segédlet fejezetben olvashat.

4.8. Üzenet fogadás oldali interfész - fogadó fél

Az ASZ az üzenetet HTTP kérésként adja át a fogadó félnek, így a standard HTTP POST alapú hívásnak megfelelően kell a fogadó oldali interfészt - a szolgáltatást - implementálni. Ugyanakkor a

KKSZB rendszer magas szintű teljesítményének, megbízhatóságának és rendelkezésre állásának megőrzése miatt a Átviteli hibakezelés fejezetben leírtak szerint, a megadott időn belül válaszolnia kell a *fogadó félnek*, különben az üzenet kézbesítetlen állapotban marad az ASZ-ben. Az alábbi fejezetben ezt az esetet külön tárgyaljuk.

4.8.1. A *fogadó fél* adott időn belül nem válaszolt *200 OK vagy 202 Accepted* státusz kóddal

Az ASZ rendszer maximum 2 percig (*üzenetküldés timeout*) várakozik a fogadó féllel történő kommunikáció során (kapcsolat felvétel és válasz érkezés). Amennyiben a feldolgozás nem fejeződik be ennyi idő alatt, úgy a kapcsolat bontásra kerül az ASZ oldalon. A fogadó félnek az ilyenkor keletkező kommunikációs hibajelzést (kapcsolat megszakadása) és az ASZ felől az ismételt üzenetküldést kezelnie kell.

A fenti esetben szélsőséges helyzetben előfordulhat olyan eset, hogy a *fogadó fél* elkezdi az üzenet feldolgozását és sikeresen be is fejezi, de mire a választ képezné addigra az ASZ bontja a kapcsolatot (kérés elküldése után 2 percig nincs válasz).

Ekkor az ASZ az üzenetet kézbesítetlen állapotként kezeli, miközben akár megtörténhet az is, hogy a *fogadó fél* feldolgozta sőt, lehet hogy értesítő üzenetet is küldött a küldő félnek a sikeres feldolgozásról. Későbbiekben az ASZ ismételten megpróbálja kézbesíteni az üzenetet, ekkor a *fogadó félnek*, ha már sikeresen feldolgozta az üzenetet, akkor is úgy kell válaszolnia, mintha sikeresen befogadta volna az üzenetet.

4.8.2. Üzenet fogadással szemben támasztott követelmények

A megfelelő nyomon követhetőség és a megbízhatóság miatt a *fogadó fél* oldalon jó minőségű kódnak kell rendelkezésre állnia, amely az alábbi követelményeket betartja:

- **megfelelő tranzakciókezelés:** a *fogadó fél* az ASZ-től érkező üzenetet perzisztálja (*commit*) és *utána* küldje a HTTP 200 OK vagy 202 Accepted választ, bármely feldolgozást ez után kezdjen meg. Így kézbesítéskor az üzenet nem veszhet el.
- **ismételt kézbesítés kezelése:** ugyanazon azonosítóval rendelkező üzenet többször elérheti a *fogadó felet*, ekkor egy sikeres feldolgozás után, minden további ismétlődő kérésre sikeres kézbesítés üzenettel (HTTP 200 OK vagy 202 Accepted) **kell** válaszolni
- **elvárt időn belül a tranzakció lezárása:** a *fogadó félnek* a kézbesítés tranzakcióját az *üzenetküldés timeout* szerint meghatározott idő alatt be **kell** fejeznie. Amennyiben a *fogadó oldalon* ez nem lehetséges, úgy az üzleti logikába épített módon, az *üzenet átvételéről* gondoskodni **kell** a *üzenetküldés timeout* szerint meghatározott idő alatt, és az üzenet feldolgozását később, *a kézbesítéstől függetlenül* kell elvégeznie.
- **a kézbesítésre adott válasz a kézbesítésre vonatkozik, nem a tartalomra:** ezért csak HTTP 200 OK vagy 202 Accepted válaszkódot küldhet vissza a kézbesítéskor a *fogadó fél*, legfeljebb a válasz *timeout* időn belül (2 perc). Bármely eltérő válaszkód esetén kézbesítetlenként kerül megjelölésre az üzenet, és ismételten kézbesítést fog végezni az ASZ rajta mindaddig, amíg az üzenet törlésre nem kerül. A *fogadó félnek* saját magának kell gondoskodnia alkalmazásának monitorozásáról, hogy kiderüljön az az eset, amikor nem HTTP 200 OK vagy 202 Accepted állapottal válaszol.

Amennyiben a feldolgozás eredményét közölni kell a *küldő féllel*, úgy *kétirányú értesítésnek* megfelelően lehet eljárni.

4.8.3. Sorrendtartó üzenetek kezelése

Az ASZ rendszer önmagában nem támogatja az üzenetek sorrend tartását, de ez nem jelenti azt, hogy nem oldható meg rendszer szinten.

A weben számos anyag található a garantált és sorrendtartó átviteli mintákról: [cloud-message-delivery]

Ugyanakkor az üzenetek sorrendjére is igaz az elosztott rendszerekre jellemző *eventually consistent* állapot, amikor is a sorrend előbb-utóbb konzisztensé válik. Ugyan nem sorrendben érkeznek be az üzenetek, de az üzenetbe elhelyezett *dátum, üzenet kapcsolat azonosító és sorrend értékek* szerint *sorba állíthatóak*. A megfelelő sorrend (konzisztencia) elérésének ideje implementáció és üzleti logika függő, néhány milliszekundumtól akár néhány nap is lehet.

Ajánlott megoldás az, hogy minden üzenetet fogadjon be a *fogadó fél*, majd amikor arra igény van (pl.: lekérdezés esetén) a perzisztált adatokat állítsa sorrendbe az *üzenet kapcsolat azonosító* és a *sorrend* értéke szerint (például rendszám és sorrend sorszám).

Előfordulhat, hogy egy adott érték nem érkezett még be, például: 1,2,...,4 a sorrend, látható hogy a 3-as sorszámú még nem létezik. Ekkor ajánlott az üzleti logikát úgy felkészíteni és úgy tervezni hogy ezt az állapotot kezelje, például felhívja a lekérdező figyelmét arra, hogy jelenleg még nem konzisztensek az adatok, vagy adott válaszüzenettel megtagadja a lekérdezést.

Javasoljuk, a sorrendtartó üzenetek kezelésének mellőzését és helyette az *idempotens* működésnek megfelelő tervezést ajánljuk, bővebben: Duplikált üzenet kezelési pattern

4.9. Hibás kézbesítés kezelése

Ritkán, de előfordulhat olyan helyzet, hogy az *üzenet lejárat időpontjáig* nem sikerül kézbesíteni az adatot és azt az ASZ saját adatbázisából emiatt törli. Ekkor a *küldő fél* biztosan nem észleli az *üzenet törlési eseményt*, a *fogadó fél* oldalán pedig implementáció függő az, hogy észlelik-e azt.

Ilyen *hibás kézbesítési* helyzeteket megakadályozni nem lehet, de előfordulási esélyüket jelentősen lehet csökkenteni körültekintő tervezéssel és teszteléssel.

Amikor egy üzenetet több napig nem lehet kézbesíteni a *fogadó félnek*, akkor általában **üzemeltetési vagy implementációs probléma áll a háttérben (fogadó fél oldali program hiba).**

Ebből az implementációs hibákat úgy lehet csökkenteni, hogy az *üzenet befogadását* és az *üzenet feldolgozását* kettéválasztjuk, így az üzeneteket befogadó alkalmazás jelentősen kevesebb hibalehetőséget tartalmaz mintha magát az üzenet feldolgozását is kezelné.

Ajánlott a *fogadó fél* oldalon az üzenetek befogadását és az üzenetek feldolgozását kezelő programrészeket határozottan kettéválasztani és külön

tranzakcióként kezelni.

A másik eset, amikor üzemeltetési beállítások miatt (például hibás tűzfal beállítás) nem lehet a *fogadó fél* részére az üzenetet kézbesíteni. Mivel az üzemi hibák sok félék lehetnek, ezek kivédése nehezebb és változatosabb.

Ajánlott az aszinkron szolgáltatások használatának elkezdésekor funkcionális ellenőrzést végezni, ahol a *küldő fél* és a *fogadó fél* oldaláról is fejlesztői és üzemeltetői közreműködéssel ellenőrzik, hogy az adott üzenetek a fogadó fél oldalán megfelelően megjelennek.

Mivel gyakran üzemeltetési vagy alkalmazás hiba áll a háttérben, így **az automatikus újraküldés nem oldja meg a tartós problémát, mindenképpen emberi beavatkozásra van szükség.**

A hiba elhárítása után lehet az üzeneteket érdemben újraküldeni. Ehhez szükséges egyedi üzenet azonosítókat az ASZ tárolja és rendelkezésére bocsátja a küldő és fogadó félnek, így a hiba kivizsgálását és az újraküldést is támogatja.

Azoknak az üzeneteknek az egyedi azonosítóját, amelyeket az üzenet lejárat időpontjáig sem sikerült sikeresen kézbesíteni az ASZ, az üzenet törlésekor rögzíti (üzenet kézbesítési hibalista) és emailben naponta megküldi a küldő és fogadó félnek. Azokról az üzenetekről is napi jelentést küld, amelyeket nem sikerült átküldeni legalább 6 próbálkozással, hogy a *fogadó fél* figyelmét felhívja az üzenetbefogadási problémára.

Technikailag a *küldő fél* **Kapcsolattartói** és a *fogadó fél* oldalon a **Szolgáltatás Felelősök** kapják meg az üzenetet.

Amennyiben az emailben szereplő üzenet azonosítóval ismételten üzenetküldés történik, ha sikeres a kézbesítés, akkor többé már nem szerepel az emailben.

Amennyiben ismételten hibás az üzenetküldése, akkor ismét az *üzenet kézbesítési hibalistára* kerül az üzenet, amennyiben az *üzenet lejárat időpontjáig* (jelenleg 7 nap) nem sikerült sikeresen kézbesíteni.

Az ASZ az *üzenet kézbesítési hibalistára* csak metaadatokat (például az üzenet azonosítója) helyez el, magát az üzenetet nem tárolja.

4.10. Tervezési segédlet

A KKSZB ASZ igénybevételének megkönnyítése céljából az alábbiakban **tervezési mintákat** mutatunk be, amelyek a leggyakrabban előforduló eseteket fedik le. Minden mintát egy használati eseten keresztül vezetünk le és pszeudokóddal definiálunk a jobb érthetőség kedvéért. A példák magyarázatokat tartalmaznak, kitérve a tranzakciókezelésre is.

Vannak olyan minták, amelyek közvetlenül nem érintik a KKSZB ASZ megoldást, ugyanakkor szükségesnek láttuk kiemelni a félreértések elkerülése végett.

4.10.1. Egyszerű üzenetküldés

Az *egyszerű üzenetküldés* **csak akkor használható, ha** a bemutatott üzleti logika és az üzenetküldés programrész **biztosan ismételhető művelet,** vagy ha a későbbiekben ismertetett kapcsolati hibákból eredő inkonzisztencia **nem okoz problémát az üzleti logika számára,** ha ez nem biztosított, akkor a Garantált üzenetküldés fejezet szerint kell eljárni.

Használati eset: gyakori eset, hogy adatbázisunkba rögzítjük az adatokat és erről **egy** üzenetet küldünk a kapcsolódó (pl.: statisztikai) rendszer felé.

Adatbázis művelet és ASZ üzenetküldés 1 fázisú kommittal

```
tr = tranzakcio_inditas;
try { [1]
  uzenet_eloallitasa();
  tr.rekord_adatbazisba_irasa(); [2]
  ASZ.statisztika_uzenet_kuldese(); [3]
  tr.commit(); [4]
} catch (hiba) {
  tr.rollback(); [5]
}
```

[1] Hibakezelés indítása a lehetséges ASZ kommunikációs, vagy adatbázis művelet miatt.

[2] Adatbázis művelet(ek) elvégzése.

[3] ASZ üzenetküldése, ha nem *HTTP 202 Accepted* a válasz, akkor hibát dob.

[4] Amennyiben a [2] adatbázis művelet sikeresen végrehajtásra került és a [3] művelet is sikeres (*HTTP 202 Accepted* a válaszkód) , úgy a tranzakció kommitolható.

[5] Amennyiben a [2] vagy [3] művelet bármelyike hibás, úgy a vezérlés a catch ágra ugrik, ahol a rollback művelet lefut.

Magyarázat: Az adatbázis műveletek megelőzik az ASZ karbantartó üzenetküldését, **amely közvetlenül a commit művelet hívása előtt áll,** így ha az ASZ üzenetküldés sikertelen, úgy az adatbázis műveleteink is rollback művelettel visszaállításra kerülnek.

Több erőforrás együttes tranzakciókezelése esetén (pl.: XA, ChainedTransaction) a fentiek szerint kell eljárni, ilyen eset, ha például több adatbázisba kell írni egyszerre.

Arra kell figyelni, hogy az **ASZ hívása közvetlenül a commit előtt álljon.**

4.10.2. Konzisztencia

A konzisztenciát leggyakrabban a saját rendszerünkre értelmezzük (lokális konzisztencia), például az saját adatbázisunkon belüli adatok egymáshoz képesti viszonyaként. Tipikusan az *erős konzisztenciát* tranzakciókezeléssel tudjuk csak biztosítani, például az adatbázison elvégzett

különböző műveleteket összefogjuk és egy tranzakcióként kezeljük. Bonyolultabb tranzakciókezelésre van szükség, amikor kettő vagy több erőforrásunkat (pl.: adatbázist, MQ szervert, stb.) kell konzisztensen tartani, ekkor *elosztott tranzakciókat* használunk. Amikor üzenetet küldünk egy *másik rendszer* felé és a másik rendszer alacsony szintű erőforrását (pl.: adatbázis, MQ szerver) nem érjük el, akkor nem tudjuk az *elosztott tranzakció* kezelést a hagyományos módon használni, *tipikusan ilyen eset amikor HTTP kérésen (WebService) keresztül küldünk karbantartó üzeneteket.*

Ekkor már kétféle konzisztenciáról beszélünk: a saját erőforrásaink belső konzisztenciáról, illetve a rendszerünk és a távoli rendszer együttes konzisztenciájáról.

A saját erőforrásaink konzisztenciáját mindenképpen biztosítanunk kell (tranzakció menedzsment) valamint törekednünk kell arra, hogy távoli rendszerrel is *végül konzisztensek* (*eventually consistent*) legyünk.

Az Egyszerű üzenetküldés nem garantálja a teljes konzisztenciát (nem lesz végül konzisztens) a távoli rendszerrel, amelynek okát az alábbiakban mutatjuk be.

A fenti példában az ASZ üzenetküldésekor [3] feltételeztük, hogy az eredménye sikeres vagy sikertelen. Ugyanakkor van egy harmadik állapot is, amikor az **üzenetküldés [3] eredménye nem ismert.**

Ez úgy állhat elő, hogy az üzenetküldés [3] megtörténik, de a hálózati kapcsolat megszakad még mielőtt a válasz visszaérkezne, így az üzenetküldés eredményéről nem kapunk semmilyen visszajelzést. **Ekkor vagy sikerült elküldeni az üzenetet vagy nem, de egyik állapotban sem lehetünk biztosak, így azt sem tudjuk biztosan, hogy az adatbázis tranzakciót commit vagy rollback művelettel kell-e lezárnunk.** Bármelyik tranzakció műveletet is választjuk előállhat az az eset, hogy az adatbázisunk és az üzenetünket fogadó rendszer együtt inkonzisztens állapotban van.

Ilyen jellegű hiba a kapcsolat megszakadása (hálózati hiba válasz fogadásakor) vagy a kapcsolat időtúllépése (*timeout*) esetén fordulhat elő.

Táblázat 3. Az alábbi esetekben válik inkonzisztensé a rendszerünk és a távoli rendszer

Üzenetküldés [3] eredménye	Adatbázis művelet	Kézbesítés állapota	Magyarázat
sikeres	commit hiba	Kézbesítésre kerül	Az üzenet elküldésre kerül, de az adatbázis commit sikertelen, így az adatbázisunkban nem szerepel az az adat, amelyet elküldtünk.
ismeretlen	commit	Nem kézbesíti	Az üzenetküldés közben a kapcsolat megszakadt, az ASZ nem perzisztálta az adatokat, ugyanakkor az adatbázisba mégis bekerült az adat, de a távoli rendszert nem érte el az üzenet.

Üzenetküldés [3] eredménye	Adatbázis művelet	Kézbesítés állapota	Magyarázat
ismeretlen	rollback	Kézbesítésre kerül	Az üzenet beküldése megtörtént, az ASZ perzisztálta az adatokat (és kézbesítette), válaszolt a küldő félnek, de a választ a küldő fél nem kapta meg. Ekkor az adatbázisunk nem tartalmazza a megfelelő adatot, miközben a távoli rendszer megkapta az üzenetet.

A fenti esetek különböző statisztikai valószínűség szerint valósulnak meg, de **potenciálisan inkonzisztencia hibáját rejtik magukban.**

Amennyiben az üzleti logikánk a saját és a távoli rendszer között **konzisztenciát kíván meg, úgy a garantált üzenetküldést kell használni.**

4.10.3. Garantált üzenetküldés

A garantált üzenetküldést úgy kell megvalósítani, hogy szét kell választani az **üzleti tranzakciót** (üzleti műveletek és üzenet létrehozás) az **üzenetküldési tranzakciótól,** az elküldendő üzenetet perzisztens tárolóban tárolni kell (tipikusan az adatbázisunkban).

Az inkonzisztencia nem csak az előzőekben ismertetett kapcsolati hibákból, hanem több üzenetküldése esetén is előfordulhat, ha az egyik üzenet - sorban későbbi -, küldése sikertelen lesz, miközben az előtte álló(k) sikeres(ek). Az alábbi példa egy ilyen esetet szemléltet.

Használati eset: adatbázisunkba rögzítjük az adatokat és erről **több karbantartó üzenetet küldünk** a kapcsolódó rendszerek felé a KKSZB ASZ-en keresztül.

A célunk, hogy a távoli rendszer és a mi rendszerünk egy adott idő elteltével, de **végülis konzisztensé** váljanak, Data Consistency Primer, Eventually consistent fejezet

Az előző fejezetben ismertetett egyszerű üzenetküldés esetén, ha kapcsolati hiba van és inkonzisztencia keletkezik a két rendszer között, akkor az soha sem válik konzisztensé (kivéve az előző fejezetben említett újraküldés esetén).

Hibás megoldás több ASZ üzenetküldés esetén

```
tr = tranzakcio_inditas;
try { [1]
  uzenet_eloallitasa();
  tr.rekord_adatbazisba_irasa(); [2]
  ASZ_karbantarto_uzenet_kuldese("A"); [3]
  ASZ_karbantarto_uzenet_kuldese("B"); [4]
  tr.commit(); [5]
} catch (hiba) {
  tr.rollback(); [6]
}
```

[1] Hibakezelés indítása a lehetséges ASZ kommunikációs, vagy adatbázis művelet miatt.

[2] Adatbázis művelet(ek) elvégzése.

[3] ASZ üzenetküldése az **"A" rendszer felé,** ha nem *HTTP 202 Accepted* a válasz, akkor hibát dob.

[4] ASZ üzenetküldése az **"B" rendszer felé,** ha nem *HTTP 202 Accepted* a válasz, akkor hibát dob.

[5] Amennyiben a [2] adatbázis művelet sikeresen végrehajtásra került és a [3], [4] művelet is sikeres (*HTTP 202 Accepted* a válaszkód), úgy a tranzakció kommitolható.

[6] Amennyiben a [2], [3] vagy [4] művelet bármelyike hibás, úgy a vezérlés a catch ágra ugrik, ahol a rollback művelet lefut.

Magyarázat: A példa azért **hibás megoldás,** mert amennyiben a [2] és [3] művelet sikeres és a [4] művelet sikertelen, úgy megtörténik ugyan a rollback, de a [3] pontban küldött üzenet kimegy a rendszerből.

Az alábbiakban bemutatunk egy lehetséges jó megoldást, amely a konzisztenciát megőrzi. Ebben az esetben az *üzleti tranzakciók és üzenetképzés* valamint az *üzenetek küldése* külön-külön programrészben (tranzakcióban) történik.

Üzleti tranzakció végrehajtása és üzenetképzése

```
tr = tranzakcio_inditas;
try { [1]
  uzenet_eloallitasa();
  tr.uzenet_perzisztalasa("A"); [2]
  tr.uzenet_perzisztalasa("B"); [3]
  tr.rekord_adatbazisba_irasa(); [4]
  tr.commit(); [5]
} catch (hiba) {
  tr.rollback(); [6]
}
```

[1] Hibakezelés indítása az adatbázis művelet miatt.

[2] Az "A" üzenet rögzítése adatbázisba.

[3] Az "B" üzenet rögzítése adatbázisba.

[4] Adatbázis művelet(ek) elvégzése.

[5] Amennyiben a [2], [3] és [4] adatbázis művelet sikeresen végrehajtásra került úgy a tranzakció kommitolható.

[6] Amennyiben a [2], [3] vagy [4] művelet bármelyike hibás, úgy a vezérlés a catch ágra ugrik, ahol a rollback művelet lefut.

A fenti programkód az adatbázisunk konzisztenciáját biztosítja, de mi azt szeretnénk, hogy a távoli

rendszerrel is konzisztensé váljon a rendszerünk, ehhez az adatbázisba mentett üzeneteket ki kell küldeni.

Üzenetek küldése

```
uzenetek = uzenetek_felolvasasa(); [1]
for each uzenet in uzenetek { [2]
  tr = tranzakcio_inditas;
  try { [3]
    tr.rekord_adatbazisba_irasa(UZENET_ELKULDVE); [4]
    ASZ_karbantarto_uzenet_kuldese(uzenet); [5]
    tr.commit(); [6]
  } catch (hiba) {
    tr.rollback(); [7]
  }
}
```

[1] Az kiküldendő üzeneteket (azonosítóit) felolvassuk.

[2] Minden felolvasott üzeneten végigmegyünk egy ciklussal.

[3] Hibakezelés indítása az adatbázis művelet miatt.

[4] Az adatbázisban bejelölésre kerül, hogy az üzenet el lett küldve, így legközelebb nem küldjük. Ha itt hiba történik, akkor [7] a rollback végrehajtódik.

[5] ASZ felé üzenetküldése, sikertelen küldés esetén hiba dobása és [7] rollback végrehajtása.

[6] Amennyiben a [4], és [5] művelet sikeresen végrehajtásra került úgy a tranzakció kommitolható.

[7] Amennyiben a [4] vagy [5] művelet sikertelen, úgy a tranzakció rollback lefut.

A fenti programrész minden üzenetet előbb-utóbb sikeresen kiküld, így *végülis konzisztensé* válnak a kapcsolódó rendszerek. **Ebben a megoldásban garantált az üzenet legalább egyszeri kézbesítése.**

A fenti megoldás esetén, ha több alkalmazás példányról szükséges az üzenetek párhuzamos kiküldése, akkor gondoskodni kell arról, hogy ugyanaz az üzenet többször ne kerüljön kiküldésre.

Ez a megoldás biztosítja a konzisztenciát, de további előnyöket is rejt.

A fenti megoldás további előnye, hogy:

- az *üzleti tranzakciót végző* programrész **egyszerűsödik, függetlenné válik az üzenetküldéstől, így jobban tesztelhető,** valamint
- nincs kapcsolatban más erőforrással, csak az adatbázissal (jelen példában), így a **tranzakciókezelés jóval egyszerűbb**
- minden üzleti entitásunkhoz egyértelműen kapcsolhatóak és **visszakereshetőek (pl.: napló, adatszolgáltatás) az üzenetek,**

- igény esetén pontosan **ugyanaz az üzenet újra is küldhető,** ez a *fogadó fél* oldalán lévő szoftverhibából eredő inkonzisztenciát is ki tudja küszöbölni (hibajavítás után újraküldött üzenet)

Az *üzenetküldő* programrész **nagyon egyszerű, újrafelhasználható alkalmazás modult jelent,** amelybe akár az üzenetek tervezett újraküldését is be lehet építeni.

A tradicionális, *lokális tranzakciókezelést* megszoktuk, így felvetődik a kérdés, hogy Miért nem biztosít az ASZ tradicionális tranzakció menedzsmentet?, amelyet ebben a függelékben mintával és példával illusztrálva bemutatunk.

A fenti megoldással akár szinkron KKSZB szolgáltatásra közvetlenül is küldhető az üzenet, de mérlegelni kell a kapcsolódó fél **rendelkezésre állását,** azt hogy **mennyi idő alatt képes átvenni az üzenetet,** valamint hogy **mennyi üzenetet tud átvenni (üzenet szám/s).**

A KKSZB ASZ célszoftver, amely magas rendelkezésre állást biztosít, nagy mennyiségű üzenetet tud átvenni és mindezt a lehető legkisebb idő alatt próbálja elvégezni, ezek előnyök lehetnek a közvetlen szinkron végpont választásával szemben, amelyet a fejlesztőknek kell mérlegelni.

4.10.4. Sorrendtartó üzenet kezelés

Az alábbi tervezési minta nem kötődik szorosan a KKSZB ASZ rendszerhez, csupán a sorrendtartás tervezésére mutat egy példát.

A sorrendtartó üzenetek esetén az **üzenetbe integráltan** kell gondoskodni a megfelelő sorrend megtarthatóságáról, **nem szabad az üzenetküldés vagy célba érés sorrendjével dolgozni.** A gyakorlatban ez azt jelenti, hogy az üzenetben meg van nevezve az **entitás egy egyedi azonosítóval és az ahhoz kapcsolódó sorszám,** amely az üzenet sorrendjét jelenti, például:

Üzenetek küldése független programrésszel

```
...
    <okmanyId>ABC00001</okmanyId>
    <uzenetSorszam>3<uzenetSorszam>
...
```

A fenti üzeneteket így a fogadó fél mindegy milyen sorrendben kapja meg, az *uzenetSorszam* értéke szerint sorba tudja állítani az üzeneteket. Értelemszerűen a sorrendbe állításhoz szükséges a perzisztens tárolás, hiszen bármikor képesnek kell lennie a sorrend szerinti lekérdezés kiszolgálására.

Célszerű gondoskodni arról az esetről, ha nem minden üzenet érkezett még meg. Ebben az esetben például egy lekérdezés esetén, ha a sorozatszám köztes eleme maradt ki, akkor az felismerhető, és jelezhető a lekérdező személynek, hogy *még nem konzisztens adatokat lát.* Gyakorlatban ezt úgy oldják meg, hogy a hiányzó sorszámú sort is szerepeltetik a lekérdezésben és mellette például "üzenet beérkezés/feldolgozás folyamatban" feliratot jelenítenek meg. Amennyiben az utolsó tétel nem érkezett még meg, úgy arról értelemszerűen a *fogadó fél* még nem tudhat.

Ezt a megközelítést fel lehet használni időpont alapú lekérdezésekhez is, amikor az adott időpontra vetített adatokra van szükség, **ekkor az üzenetben kell közölni egy időpontot,** mert ez

alapján történik a lekérdezés és sorba állítás. Ha az üzenetek a teljes entitás adatát tartalmazzák, akkor elégséges lehet a lekérdezési időpontban ismert utolsó üzenet megjelenítése.

4.10.5. Egyszeres kézbesítés

Az egyszeres kézbesítés nem garantált a KKSZB ASZ esetén, akár a *küldő fél* ugyanazt az üzenetet többször is - akár eltérő üzenet azonosítóval is -, feladhatja. Az egyszeres kézbesítés - az ismételt üzenet kezelése -, a KKSZB ASZ hatáskörén kívül áll, azt csak a *fogadó fél* tudja *megbízható módon* kezelni, ezért a *fogadó fél* oldalán kell az ismételt üzenetek kezelését megoldani.

A legegyszerűbb megoldás, ha az *üzenet azonosítót* tárolja a *fogadó fél* és minden bejövő üzenet feldolgozása előtt egyediség vizsgálatot végez, például rögzíti az üzenetet az adatbázisban *üzenet azonosítóval,* amely egyedi kulcs. Összetettebb a probléma megoldása, ha ugyanaz az üzenet érkezik más azonosítóval és a *fogadó félnek* ezt ki kell szűrnie. Ekkor meghatározzák az egyezőség feltételeit - például az üzenet időpont nem számít bele -, majd hash képzéssel, vagy mezők összehasonlításával elvégzik az egyezőség vizsgálatát feldolgozás előtt.

Chapter 5. KKSZB rate limit

A KKSZB rendszer képes arra, hogy az adott Szolgáltatást elérő kérések számát korlátozza (rate-limiting). A korlát (rate-limit) beállítása a Szolgáltatás Felelős feladata, alapértelmezetten nincs korlátozás.

A korlátozás beállításával ugyan csökkenthető a Szolgáltatásra jutó terhelés, de ettől még a kérést küldő fél – amennyiben ezt túllépi –, célja nem teljesül, nem fog válaszhoz jutni, így a felhasználók kiszolgálási minősége sérül. Javasoljuk, hogy a korlátozást csak átmenetileg, vagy végső esetben használja, és helyette a Szolgáltatás felskálázásával, vagy gyorstárak használatával próbálja teljesíteni a felhasználói igényeket.

A korlátozást kérések számával lehet megadni az adott percre vonatkozóan, például a 60 érték jelentése: 1 perc alatt maximum 60 kérés érheti el a Szolgáltatást, ha ettől több kérés érkezik, akkor **HTTP429 Too Many Requests** üzenetet kap válaszként a kérdező fél, a HTTP fejlécben a beállított korlátozás értékét visszakapja a **x-kk-rate-limit** mezőben.

A Szolgáltatás Felelős a korlátozást az engedélyezett Szolgáltatás Elérési Jogosultságnál tudja megadni, és bármikor, igénye szerint módosítani.

Ezzel a korlátozás hatása három paramétertől függ:

- a beállított korlát értékétől, 0 esetén nincs korlátozás (alapértelmezett)
- a kliens rendszer azonosítójától, amely a SZEK-ben szerepel
- az elérendő Szolgáltatás azonosítójától, amely a SZEK-ben szerepel
- a jogalaptól, amely a SZEK-ben szerepel

A fentiek alapján például lehetséges a szabályozást úgy beállítani, hogy ugyanannak a lekérdező félnek, ugyanarra a Szolgáltatásra, a jogalapjától függően eltérő korlátozást alkalmazunk.

A KKSZB rate-limit szolgáltatása felmenő rendszerben kerül bevezetésre, a jelenleg működő rendszereket nem befolyásolja, változtatás nem szükséges részükről, amennyiben továbbra sem veszik igénybe ezt a szolgáltatást.

A KKSZB rate-limit szolgáltatás – amennyiben be van kapcsolva –, érinti a Szolgáltatást és érinti az azt igénybe vevő kliens rendszert is, ezért az alábbiakban összefoglaljuk azt, hogy mit kell figyelembe vennie ezen szereplőknek.

5.1. Szolgáltatás esetén

A KKSZB rendszer a rate-limiting funkcióval korlátozza a Szolgáltatás terhelését, így **a korlátot meghaladó kérésszám esetén, a kérés nem éri el a Szolgáltatást,** ezért ott log bejegyzés a kérésről nem történik!

Szolgáltatás felelősként bizonyosodjon meg arról, mielőtt bekapcsolja a korlátozást

(0-tól eltérő értéket ad meg), hogy a kapcsolódó fél képes a **HTTP429 Too Many Requests** üzenetet kezelésére.

Szolgáltatás felelősként vegye figyelembe, hogy milyen megállapodás (szerződés, törvényi, egyéb előírás) vonatkozik a Szolgáltatás elérésére, mielőtt bekapcsolja a korlátozást (0-tól eltérő értéket ad meg) – praktikusan a szerződés, vagy megállapodás, vagy engedély tartalmazza a kérés/perc értéket.

Szolgáltatás felelősként bármikor lehetősége van módosítani a korlátozás értékét (csökkenteni, növelni, vagy 0 értékre állítva kikapcsolni), amely legfeljebb 2 percen belül életbe lép.

A beállított korlátozás 1 percre vonatkoztatott kérés számot jelent. Ez azt is jelenti, hogy szélsőséges esetben akár 1-2 másodperc alatt is beérkezhet ez a kérés szám, az ez utáni időszakban viszont újabb kérés nem fogja elérni már a Szolgáltatást.

5.2. Kliens rendszer esetén

Amennyiben nincs korlátozva a kérés szám az adott elérésében, úgy nincs tennivalója.

Ha a korlátozás bekapcsolásra kerül, akkor szükséges implementálni a kérést indító alkalmazásban a **HTTP429 Too Many Requests** üzenetet kezelését.

Ha lehetősége van, akkor úgy implementálja a kérést küldő alkalmazást, hogy a **HTTP429 Too Many Requests** üzenetet kezelje. Ha biztosan tudja, hogy mennyi kérést intézhet az adott Szolgáltatás elé, akkor célszerű önkorlátozást bevezetni, amennyiben az technikailag megoldható.

Amennyiben a kérések száma túllépi a korlátot, akkor a **HTTP429 Too Many Requests** üzenetet kapja mindaddig, amíg a kérések száma az utolsó percre vonatkoztatva le nem csökken a korlát alá.

A **HTTP429 Too Many Requests** üzenetet a KKSZB rendszer küldi, a Szolgáltatás oldalon ezt nem fogják érzékelni, nem kapja meg a Szolgáltatás a kérést, így ott a kérésről semmilyen információ nem keletkezik.

Meg lehet különböztetni azt, hogy a HTTP429 válasz a KKSZB-től, vagy a mögötte lévő Szolgáltatástól (vagy egyéb KKSZB mögötti eszköztől, szoftvertől) érkezett-e. **Ha a KKSZB rendszertől érkezik a válasz,** akkor a **x-kk-gw-status-message** és a **x-kk-rate-limit** HTTP fejlécek szerepelnek a válaszban.

Az **x-kk-rate-limit** HTTP fejléc értéke a beállított korlátozás (rate-limit) értékét tartalmazza.

5.2.1. Egyéb korlátozások

Előfordulhat, hogy a KKSZB rendszeren kívül más komponens – akár maga a Szolgáltatás –, szintén (vagy csak az) tartalmaz korlátozó funkciót, amely ugyanúgy a **HTTP429 Too Many Requests** választ adhatja a KKSZB rendszertől függetlenül.

Amennyiben problémája adódik a korlátozással kapcsolatban, akkor vegye fel a kapcsolatot a

Szolgáltatás Felelősével.

Definíciók

KKSZB

Központi Kormányzati Szolgáltatás Busz

[Szerződő fél]

Egy tetszőleges (piaci vagy közigazgatási) szervezet, amely csatlakozik a KKSZB rendszerhez.

[Kapcsolódó fél]

Egy tetszőleges kapcsolódó fél, mely csatlakozik a KKSZB-hez, és az ott lévő szolgáltatásait (szolgáltató), vagy az ott igénybe vett szolgáltatások (kliens) tekintetében a kapcsolódó félhez rendelt személyek eljárhatnak.

[Kliens]

(Kliens) Kapcsolódó fél, aki a KKSZB rendszerben nyújtott szolgáltatásokat használja HTTP protokollon keresztül.

[Szolgáltató]

(Szolgáltató) Kapcsolódó fél, aki a KKSZB rendszerben szolgáltatásokat publikál HTTP protokollon keresztül.

[Kapcsolattartó]

Olyan természetes személy, akit a kapcsolódó fél felhatalmazott, hogy a KKSZB rendszerben a kapcsolódó felet képviselje.

[Szolgáltatás felelős]

Az a személy, aki a KKSZB rendszerben a Szolgáltatás felett teljes jogkörrel rendelkezik: szolgáltatásokat futtat, publikálja a KKSZB rendszerben, a Kliensek Szolgáltatáshoz történő hozzáférését szabályozza, stb.

[Szolgáltatás Katalógus Kezelő]

Az a személy, aki a KKSZB rendszerben a *Kapcsolódó fél* által a Szolgáltatás interfész leírásának menedzsmentjével meghatalmazott személy, szolgáltatás létrehozását vagy megszüntetését nem kezdeményezheti.

[RFNY admin]

Az a személy, aki a KKSZB Rendszer Felhatalmazási Nyilvántartás rendszerében admin szerepkörrel meghatalmazott, a rendszerhez csatlakozó feleket kezeli, és a szolgáltatás kérelmeket elbírálja, felülvizsgálja, felfüggeszti amennyiben szükséges.

[Szolgáltatás]

(KKSZB szolgáltatás) Alkalmazás, amely a KKSZB rendszerben szolgáltatást nyújt HTTP protokollon keresztül. Szolgáltatásnak nevezünk minden KKSZB-n igénybe vehető, HTTP-n elérhető webes szolgáltatást.

[KKSZB felügyeleti szerv]

Az a szervezet, amely a KKSZB rendszer felügyeletét ellátja, rendelkezik felette és működésére

hatással lehet.

[KKSZB Gateway]

(röv.: gateway vagy GW) Olyan KKSZB architekturális elem, amely az adott kliens vagy szolgáltatás csatlakozását lehetővé teszi a KKSZB rendszerbe.

[Rendszer Felhatalmazási Nyilvántartás]

(röv. *RFNY*) Rendszer Felhatalmazási Nyilvántartás, olyan Webes felülettel rendelkező rendszer, amely a KKSZB rendszer kliens, szolgáltatás és gateway (átjáró) alkalmazásainak a hitelesítését és jogosultság kiosztását biztosítja az arra felhatalmazott személyek által.

[Szolgáltatás Katalógus]

Szolgáltatás Katalógus, amely a KKSZB rendszer összes elérhető szolgáltatásáról információt nyújt: *KKSZB végpont*, interfész leírás és egyéb adatok.

[Green-Page]

Green-Page, a KKSZB rendszerhez való csatlakozást segítő technikai, fejlesztői ismeretanyagok, cikkek megosztását, ismeretanyagok közzétételét szolgáló alkalmazás.

[Statisztika]

Statisztika, olyan Webes felülettel rendelkező rendszer, amely a KKSZB RFNY adatbázisából, és a BigData log rendszeréből statisztikákat, kimutatásokat készít, az admin, a SZF és a KT felhasználók számára.

[Aszinkron Szolgáltatás]

Aszinkron Szolgáltatás, amely a KKSZB rendszerben biztosítja a HTTP protokollon történő aszinkron üzenet küldés lehetőségét.

[Echo Szolgáltatás]

Az Echo szolgáltatás a beüzemelés és az üzemi állapot ellenőrzésére szolgál a Kliensek irányából, a beküldött üzenetet visszaküldi.

[Próba kliens]

Olyan kliens, amely a KKSZB rendszert felügyelő szervezet saját, kliensként csatlakozó alkalmazása, amely segíti a valós kliens és szolgáltatás kapcsolat ellenőrzését a Kliens irányából.

[szolgáltatás létrehozási kérelem]

A szolgáltatók, ha létre akarnak hozni egy új szolgáltatást, szolgáltatás létrehozási kérelmet hoznak létre, melyet a KKSZB felügyeleti szerv megfelelő jogosultsággal rendelkező felhasználója elbírál.

[szolgáltatási kérelem]

A szolgáltatók, ha létre akarnak hozni egy új szolgáltatást, vagy meg akar szüntetni, akkor a szolgáltatásra vonatkozó kérelmet tölt ki és küld be, melyet a KKSZB felügyeleti szerv megfelelő jogosultsággal rendelkező felhasználója elbírál.

[szolgáltatás elérési jogosultság kérelem]

Amennyiben egy kapcsolódó fél el kíván érni egy szolgáltatást, a szolgáltatás elérési jogosultság

megszerzéséhez egy szolgáltatás elérési jogosultság kérelmet kell létrehoznia, melyet az adott szolgáltatást nyújtó Szolgáltatás felelőse elbírál.

[szolgáltatás elérési jogosultság]

A kliensnek rendelkeznie kell az adott KKSZB szolgáltatás eléréséhez szükséges jogosultsággal. Nem keverendő össze az egyéb, *szakrendszeri jogosultságokkal.*

[kliens autentikációs token]

Olyan Json Web Token (JWT), amely az RFNY által kiállított, kizárólag egy adott kapcsolódó fél, egy adott *Szolgáltatás elérési jogosultságához* tartozó, KKSZB rendszerben történő hitelesítésére szolgál. Érvényessége a kiállítástól számított maximum 12 hónap. Biztonságos, megbízható módon kezelendő, titkos információ, csak a KKSZB és a kliens ismerheti.

[kliens access token]

Olyan Json Web Token (JWT), amely az RFNY által kiállított, rövid lejáratú (néhány perc), kizárólag egy adott kliens által a KKSZB szolgáltatások elérésére szolgál. Csak a kliens autentikációs token birtokában szerezhető meg a KKSZB kliens gateway által. A szolgáltatások számára megismerhető információ.

[KKSZB szolgáltatás végpont]

Olyan URL, amely a KKSZB rendszerben értelmezett HTTP szolgáltatási végpont, például: https://gw.kkszb.gov.hu/jarmu/leksz/rsz/v1 Ez az adott szolgáltatás előtt álló gatewayen értelmezett elérési pont. A szolgáltatás azonosítóval felírva a *KKSZB szolgáltatás végpont:* `https://gw.kkszb.gov.hu/szolgáltatás-azonosító`

[valós szolgáltatási végpont]

Olyan URL, amely az adott szolgáltató eszköz környezetében értelmezett, a szolgáltatás oldali KKSZB gatewayről elérhető.

[szolgáltatás névtér]

(KKSZB szolgáltatás névtér) A KKSZB-n elérhető szolgáltatások névterekbe vannak szervezve. A névteret a szolgáltatás végpont URL útvonal részének első tagja azonosítja. Ez egyben a névtér egyedi azonosítója is, pl.: a *jarmu* névtér a következőképpen jelenik meg a szolgáltatás végpont URL-ben: *https://gw.kkszb.gov.hu/jarmu*

[szolgáltatás azonosító]

A KKSZB rendszerben értelmezett szolgáltatás azonosítója, például: **/jarmu/leksz/rsz/v1** vagy **/jarmu/private/leksz/eucaris/rsz/v1** A KKSZB szolgáltatás végpont és a **szolgáltatás azonosító** a protokoll (https) és a domain név (kkszb.gov.hu) kivételével megegyezik.

[KKSZB azonosítási rendszer]

A KKSZB átvállalja az alkalmazás szintű azonosítást a szolgáltatástól. A szakrendszeri felhasználók azonosítása és a szakrendszeri jogosultság kezelése továbbra is a szolgáltatás felelőssége marad.

[kliens alkalmazás]

Olyan alkalmazás, amely a KKSZB rendszerre, mint kliens csatlakozik és Szolgáltatást vesz igénybe.

[állapot információ végpont]

Olyan szolgáltatás végpont, amely 200 OK HTTP státus kóddal válaszol, amennyiben a szolgáltatás üzemszerűen használható.

[polimorf szolgáltatás]

Azokat a szolgáltatásokat, amelyek **ugyanazon lekérdezésre, ugyanazon végponton más választ adnak** attól függően, hogy mely rendszertől (vagy felhasználótól) érkezett a kérés, polimorf szolgáltatásoknak nevezzük.

Hivatkozások

- [kkszb-magas-szintu-szoftverarchitektura] **KKSZB magas szintű szoftverarchitektúra** dokumentum: kkszb_magas_szintu_szoftverarchitektura.pdf
- [kkszb-logikai-rendszerterv] **KKSZB logikai rendszerterv** dokumentum: kkszb_logikai_rendszerterv.pdf
- [kkszb-fizikai-gepigeny] **KKSZB fizikai gépigény** dokumentum: kkszb-fizikai-gepigeny.pdf (hely: 03_rendszerterv/04_fizikai/)
- [kkszb-fizikai-kapcsolat] **KKSZB fizikai kapcsolatok** dokumentum: kkszb-fizikai-kapcsolat.ods (hely: kkszb-docs/adoc/architektura/)
- [kkszb-fizikai-architektura] **KKSZB fizikai architektúra** dokumentum: kkszb-fizikai-architektura.pdf
- [UUID] **A Universally Unique IDentifier (UUID) URN Namespace,** https://www.ietf.org/rfc/rfc4122.txt
- [JWT] **JSON Web Token (JWT),** https://www.ietf.org/rfc/rfc7519.txt
- [Base64url] **The Base16, Base32, and Base64 Data Encodings,** https://www.ietf.org/rfc/rfc4648.txt
- [it-biztonsag] **IT biztonság:** NEIH előadás: biztonsági osztályba sorolás, NEIH: "Útmutató a 77/2013. NFM Rendelet Szerinti Biztonsági Felmérést Támogató Segédlethez", NbSZ: "A Kormányzati Eseménykezelő Központ működésének tapasztalatai", Magyar Közlöny: 77/2013. (XII. 19.) NFM rendelet
- [cloud-message-delivery] **Megbízható és sorrendtartó üzenet kezelési minták,** http://www.cloudcomputingpatterns.org, https://www.infoq.com/articles/no-reliable-messaging

függelék A: Kliens autentikációs token

A kiadott tokent kezelje bizalmasan. Azt csak a kliens szoftver és a KKSZB kliens gateway ismerheti meg az RFNY rendszeren kívül.

A kliens autentikációs token azonosít egy kliensként kapcsolódó rendszert (alkalmazást) a KKSZB rendszerben. A token kibocsájtója az RFNY, amely JWT formájú, és amelyet az RFNY digitálisan aláír.

A tokent csak az a kliens használhatja, amely részére kiadták, és csak a KKSZB rendszerben szabad azt felhasználni.

Minden jogosult kliens példány ugyanazt a tokent használja, nem szükséges alkalmazás példányonként újabb tokent igényelni.

Token kiadása

A token létrehozását és letöltését **a kapcsolódó fél kliens kapcsolattartó szereppel rendelkező felhasználói végzik** az RFNY webes felületén keresztül.

Token verziók

2023-ban, az RFNY CouchDB-ről MongoDB-re való átállásakor, a token egyes verzióját a kettes verzió váltotta fel. Az új verzió bevezetését az entitások azonosítójának a korábbi CouchDB-sről MongoDB-sre való módosulása indokolta.

Az adatbázis migrációjáig a token verziója egyes, CouchDB-ben generált ID-kal töltött serviceId, sapId, legalbasisId mezőkkel, az után kiállítottak verziója kettes, MongoDB ID-kal generált.

Ez a változás, mivel az újonnan generált tokenek értékeit megváltoztatta, a token ellenőrzési eljárásának módosítását követelte.

Az első verzió használata, a tokenek éves lejáratása miatt 2024-től folyamatosan megszűnik.

Token szerkezete

Minta kliens autentikációs token - version 1

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjEifQ.eyJqdGkiOiIyZDgyNWY2ZC0xZmFiLTRmY2E
tOGIxMi0wZWYxNDU4ZTBkMDAiLCJpc3MiOiJ1cm46c3lzOmtrc3piOmZueSIsInN1YiI6InVybjpwaWQ6a2tze
mI6cGVlcjEiLCJhdWQiOiJ1cm46c3lzOmtrc3piOmdhdGV3YXkiLCJ0eXBlIjoidXJuOnRva2VuOmtrc3piOmN
saWVudDphdXRoIiwiaWF0IjoxNDkzMTEzNjUzLCJuYmYiOjE0OTMxMTM2NTMsImV4cCI6MTUyNDY0OTY1Mywic
2VydmljZUlkIjoiTDJwaGNtMTFMM05sY25acFkyVTFMM1l4Iiwic2VydmljZVVyaSI6Ii9qYXJtdS9zZXJ2aWN
lNS92MSIsInNhcElkIjoiNDFmNzVlOTYtZWI0Ni00NTBjLTk0NmUtNTE3YmVjZjVlYzlhIiwic2FwTmFtZSI6I
mRlZmF1bHQiLCJsZWdhbEJhc2lzSWQiOiJkY2RmMWU4MiIsIm5hbWUiOiJUb2tlbjEiLCJsZWdhbEJhc2lzQ29
kZSI6IkpBUjEyMDJBIiwic2VjdXJpdHlDbGFzcyI6NCwidmVyc2lvbiI6MX0.i6NZ3G4zik1V8RJvdMphSMbwj
```

```
dGxSwR3Bim4X608DtXWjWQKie-ymrwaFstvn9lGv1o1jLwpDgc-
Ub9gOYzrEbl4WMxFVo5Bz9uWA0vJMDwBcKO1QhH0u3sBhYqtVuQD_TchUt61cLBobESVaa1a9QSGRAB-
UNVlnVXEbJBJvF3nXxVNB7RZ-dUix4r2jooGo8pqaaBQocyUJgIHUqGqJuyJeGVV0LtrIoMWxNkmHNSPaIpL-
ns_XrumzVZZqYOhGbuODMrmCWJRzYl0CsPZo8BtUnRzEhK1kgrBkaSwlKlPEKxv-
S9fKxOnhiVp6KBbNZhLj75Y0rdzSy3K5eol3Q
```

Minta kliens autentikációs token - version 2

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjMifQ.eyJqdGkiOiIyZDgyNWY2ZC0xZmFiLTRmY2E
tOGIxMi0wZWYxNDU4ZTBkMDAiLCJpc3MiOiJ1cm46c3lzOmtrc3piOmZueSIsInN1YiI6InVybjpwaWQ6a2tze
mI6cGVlcjEiLCJhdWQiOiJ1cm46c3lzOmtrc3piOmdhdGV3YXkiLCJ0eXBlIjoidXJuOnRva2VuOmtrc3piOmN
saWVudDphdXRoIiwiaWF0IjoxNDkzMTEzNjUzLCJuYmYiOjE0OTMxMTM2NTMsImV4cCI6MjIyNDY0OTk5OSwic
2VydmljZUlkIjoiNjM5YjZhNDIzNmQ2NWMwNmY2ODg4YTBlIiwic2VydmljZVVyaSI6Ii9qYXJtdS9zZXJ2aWN
lNS92MSIsInNhcElkIjoiNjM5YjZhNDIzNmQ2NWMwNmY2ODg4YTBmIiwic2FwTmFtZSI6ImRlZmF1bHQiLCJsZ
WdhbEJhc2lzSWQiOiI2MzliNmE0MjM2ZDY1YzA2ZjY4ODhhMGciLCJuYW1lIjoiVG9rZW4xIiwibGVnYWxCYXN
pc0NvZGUiOiJKQVIxMjAyQSIsInNlY3VyaXR5Q2xhc3MiOjQsInZlcnNpb24iOjJ9.CcxpOhnShARbIq3QnTJV
Rjgt1rmynx3J6R5IXKC77Lw3Tk_lEOcGoW9Kz09pC4riAK3yCMC0BtOomhlG3OGZCvlZSHSjG7v2EmrTeS8qs-
iNIfPPArew0U2eNCOoOoPvGuuyBQYxQ9mg5i3rPdM8nxzNnyC-
xVSbIyxkdQ803eONqLRuObgvOCubqh20rLiyZUqzZ83yuCDsWR_WYYB3nikoIiKJLWqfTkUPbrtlhVEBAEd68n
sWGVT0K0Ie3oEqNs03Z_eXrBJLnwJnReZaEhZ5gqhNztXtWDFIdAdfNyW3V2PXpz5MmyDpu9rSRspYlK_A0Qvk
2j6n9mVL1ToSFQ
```

Az autentikációs token egy [JWT], amely az RFNY által alá van írva. Kódolása [Base64url], amely olvasható, tartalma JSON.

Minta kliens autentikációs token header Base64url kódolás nélkül (JWT header)

```
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "3"
}
```

Minta kliens autentikációs token body Base64url kódolás nélkül (JWT claim) - version 1

```
{
  "jti": "2d825f6d-1fab-4fca-8b12-0ef1458e0d00",
  "iss": "urn:sys:kkszb:fny",
  "sub": "urn:pid:kkszb:peer1",
  "aud": "urn:sys:kkszb:gateway",
  "type": "urn:token:kkszb:client:auth",
  "iat": 1493113653,
  "nbf": 1493113653,
  "exp": 2224649999,
  "serviceId": "L2phcm11L3NlcnZpY2U1L3Yx",
  "serviceUri": "/jarmu/service5/v1",
  "sapId": "41f75e96-eb46-450c-946e-517becf5ec9a",
  "sapName": "default",
  "legalBasisId": "dcdf1e82",
```

```
  "name": "Token1",
  "legalBasisCode": "JAR1202A",
  "securityClass": 4,
  "version": 1
}
```

Minta kliens autentikációs token body Base64url kódolás nélkül (JWT claim) - version 2

```
{
  "jti": "2d825f6d-1fab-4fca-8b12-0ef1458e0d00",
  "iss": "urn:sys:kkszb:fny",
  "sub": "urn:pid:kkszb:peer1",
  "aud": "urn:sys:kkszb:gateway",
  "type": "urn:token:kkszb:client:auth",
  "iat": 1493113653,
  "nbf": 1493113653,
  "exp": 2224649999,
  "serviceId": "639b6a4236d65c06f6888a0e",
  "serviceUri": "/jarmu/service5/v1",
  "sapId": "639b6a4236d65c06f6888a0f",
  "sapName": "default",
  "legalBasisId": "639b6a4236d65c06f6888a0g",
  "name": "Token1",
  "legalBasisCode": "JAR1202A",
  "securityClass": 4,
  "version": 2
}
```

Táblázat 4. Az autentikációs token header részében szereplő kulcsok és leírásuk

Kulcs	Leírás
alg	A token aláírásához használt algoritmus (JWA). Jelenleg két algoritmus támogatott: ***RS256*** és ***ES256***
typ	A token típusa. ***JWT***
kid	Az aláíráshoz használt kulcs azonosítója, pl.: **1**

Táblázat 5. Az autentikációs token body részében szereplő kulcsok és leírásuk

Kulcs	Leírás
jti	Az autentikációs token egyedi azonosítója.
iss	A kibocsájtó egyedi azonosítója. ***[urn:sys:kkszb:fny]***
sub	A kapcsolódó fél (kliens alkalmazás) egyedi azonosítója, aki a szolgáltatást igénybe veszi, pl.: **urn:pid:kkszb:peer1**
type	A token típusa ***[urn:token:kkszb:client:auth]***
aud	A token célközönségének azonosítója ***[urn:sys:kkszb:gateway]***
iat[1]	Kibocsájtás időpontja (POSIX time).

Kulcs	Leírás
nbf[1]	Ezen időponttól érvényes a token (POSIX time).
exp[1]	Ebben az időpontban jár le a token. Megjegyzés: előfordulhat az az eset, hogy az *exp* időpont a szolgáltatáshoz történő beérkezéskor már elmúlt. Ez adódhat abból, hogy a szolgáltatás szerveren és az RFNY szerver időpontja eltér, valamint abból, hogy a KKSZB rendszeren történő áthaladás időt vett igénybe, és az ellenőrzéskor még nem érte el az *exp* időpontot Az előzőek miatt a szolgáltatáson ezt az értéket nem kell ellenőrizni, arról a KKSZB gondoskodik.
serviceId	Az autentikációs tokenhez tartozó szolgáltatás egyedi azonosítója. (v1: CouchDB ID, v2: MongoDB ID)
serviceUri	Az autentikációs tokenhez tartozó szolgáltatás URI-ja.
sapId	A szolgáltatás elérési jogosultság egyedi azonosítója. (v1: CouchDB ID, v2: MongoDB ID)
sapName	A szolgáltatás elérési jogosultság neve, amely a könnyebb azonosíthatóságot és nyomonkövethetőséget segíti.
name	Az autentikációs token neve, például: az alkalmazás címkéje, amelyben felhasználásra kerül.
legalBasisId	Jogalap azonosító. (v1: CouchDB ID, v2: MongoDB ID)
legalBasisCode	Jogalapkód értéke, amelyet a Szolgáltatás felelős adott meg az RFNY felületén ehhez a jogalaphoz.
securityClass	A szolgáltatást hívó fél kérésének IBTV biztonsági osztály értéke, numerikus érték 1-5 között (implicit). Értéke a Szolgáltatás Elérési Jogosultság Kérelem kitöltésekor kerül megadásra a Kapcsolódó fél által.
version	A token verziója.

[1] Az iat, nbf és exp értéke 'POSIX time', másodperc pontossággal.

függelék B: Kliens access token

A kliens access token felhasználója a KKSZB gateway, a kliens oldalon nem használt, ugyanakkor a kliens access token megismerhető a szolgáltatás számára is, így az ott igény szerint felhasználható.

A kliens access token tartalmazza a kliensről szóló információkat egy adott kérésben. A token kibocsájtója az RFNY, [JWT] formájú, amelyet az RFNY digitálisan aláír.

Token kiadása

A token kiadását az RFNY automatikusan végzi a Kliens autentikációs token alapján. Csak a KKSZB Gateway képes a token megszerzésére.

A token a HTTP fejlécben az **x-kk-access-token** mezőben elérhető a szolgáltatás oldalon.

Token verziók

2023-ban, az RFNY CouchDB-ről MongoDB-re való átállásakor, a token egyes verzióját a kettes verzió váltotta fel. Az új verzió bevezetését az entitások azonosítójának a korábbi CouchDB-sről MongoDB-sre való módosulása indokolta.

Az adatbázis migrációjáig a token verziója egyes, CouchDB-ben generált ID-kal töltött serviceId, sapId, legalbasisId mezőkkel, az az után kiállítottak verziója kettes, MongoDB ID-kal generált.

Ez a változás, mivel a generált token értékeit megváltoztatta, a token ellenőrzési eljárásának módosítását követelte.

Token szerkezete

Minta kliens access token - version 1


```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjEifQ.eyJqdGkiOiJkNzNlYzBmYy1lZTU2LTQwZDk
tYjVhYS1iZDE3NDZlMmJhM2MiLCJpc3MiOiJ1cm46c3lzOmtrc3piOmZueSIsInN1YiI6InVybjpwaWQ6a2tze
mI6cGVlcjEiLCJ0eXBlIjoidXJuOnRva2VuOmtrc3piOmNsaWVudDphY2Nlc3MiLCJpYXQiOjE0OTMxMTM2NTM
sIm5iZiI6MTQ5MzExMzY1MywiZXhwIjoxNDkzMTE0MjUzLCJzZXJ2aWNlSWQiOiJMMnBoY20xMUwzTmxjblpwW
TJVMUwzWXgiLCJzZXJ2aWNlVXJpIjoiL2phcm11L3NlcnZpY2U1L3YxIiwiYXV0aHRva2VuSnRpIjoiMmQ4MjV
mNmQtMWZhYi00ZmNhLThiMTItMGVmMTQ1OGUwZDAwIiwic2FwSWQiOiI0MWY3NWU5Ni1lYjQ2LTQ1MGMtOTQ2Z
S01MTdiZWNmNWVjOWEiLCJzYXBOYW1lIjoiZGVmYXVsdCIsImF1dGh0b2tlbk5hbWUiOiJUb2tlbjEiLCJsZWd
hbEJhc2lzSWQiOiJkY2RmMWU4MiIsImxlZ2FsQmFzaXNDb2RlIjoiSkFSMTIwMkEiLCJzZWN1cml0eUNsYXNzI
jo0LCJ2ZXJzaW9uIjoxfQ.UUnMFj2oyG3tsDRHQCKB5yFL2rSX-
SZpHsAVie522FSAhAUXePZmJJNCYq8wVqkzrBvQ_VmlM3X5jyKuDR_TrKeNAOWVrCG4B0VwsRc9U0hBNuaxYLe
jktH_z-8SFdgIF7nlBUoQ2JnFOxZQLlpKQY7-OHObUj6aJBDbnqt8O731Jga2QjY-
1HM9e9dbONLXBY_FrhXYpQZWDvokRNndAqZwNQ2YeIy8ONds6cUhQTh_8-
wRjOBvkqvQAjXvpraJMG5_WtWYWY2n_wMQtZgdaszxjdnT191KbQoLqoO-P-
YVAmR7FrF6yXdJvvXZrnNZM4kIs4gCcXn_xH3njomB-Q
```

Minta kliens access token - version 2

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjMifQ.eyJqdGkiOiJkNzNlYzBmYy1lZTU2LTQwZDk
tYjVhYS1iZDE3NDZlMmJhM2MiLCJpc3MiOiJ1cm46c3lzOmtrc3piOmZueSIsInN1YiI6InVybjpwaWQ6a2tze
mI6cGVlcjEiLCJ0eXBlIjoidXJuOnRva2VuOmtrc3piOmNsaWVudDphY2Nlc3MiLCJpYXQiOjE0OTMxMTM2NTM
sIm5iZiI6MTQ5MzExMzY1MywiZXhwIjoxNDkzMTE0MjUzLCJzZXJ2aWNlSWQiOiI2MzliNmE0MjM2ZDY1YzA2Z
jY4ODhhMGUiLCJzZXJ2aWNlVXJpIjoiL2phcm11L3NlcnZpY2U1L3YxIiwiYXV0aHRva2VuSnRpIjoiMmQ4MjV
mNmQtMWZhYi00ZmNhLThiMTItMGVmMTQ1OGUwZDAwIiwic2FwSWQiOiI2MzliNmE0MjM2ZDY1YzA2ZjY4ODhhM
GYiLCJzYXBOYW1lIjoiZGVmYXVsdCIsImF1dGh0b2tlbk5hbWUiOiJUb2tlbjEiLCJsZWdhbEJhc2lzSWQiOiI
2MzliNmE0MjM2ZDY1YzA2ZjY4ODhhMGciLCJsZWdhbEJhc2lzQ29kZSI6IkpBUjEyMDJBIiwic2VjdXJpdHlDb
GFzcyI6NCwidmVyc2lvbiI6Mn0.Ms1KnXmSkCcMkD_eLyuqGV-
GS6LKMQz6DPP0T2LCpHOUPjBJxHAglqmo3bznvv7vtjeOQf7yW3rQeJs1QZ6A0hY_5QLbtNzZOLgH6sTR0_Zde
jQtS1V3qnccubIf_QOhpTh2NHnNSI5Tci7_kLGrfy-
SQH4DQDhkq_IEtixwJuxPvNWHAmbazENnY_1lxeecjC3euZHszPohKoJx0AhY39SqTfjgao__-
w_p_eILmEYE2n4yEWLv8JNdzpPbUK1sbhQf7hiBz7e20LIrdNWiDH7_yYXmmSAzWoQM6R4b4OPV8Bv5r20Gs9A
sVlVSk2pv6dAW_obCxXhA25hAEhylRA
```

Az access token egy [JWT], amely az RFNY által alá van írva. Kódolása [Base64url], amely olvasható, tartalma JSON.

Minta kliens access token header Base64url kódolás nélkül (JWT header)

```
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "3"
}
```

Minta kliens access token body Base64url kódolás nélkül (JWT claim) - version 1

```
{
  "jti": "d73ec0fc-ee56-40d9-b5aa-bd1746e2ba3c",
  "iss": "urn:sys:kkszb:fny",
  "sub": "urn:pid:kkszb:peer1",
  "type": "urn:token:kkszb:client:access",
  "iat": 1493113653,
  "nbf": 1493113653,
  "exp": 2224649999,
  "serviceId": "L2phcm11L3NlcnZpY2U1L3Yx",
  "serviceUri": "/jarmu/service5/v1",
  "authtokenJti": "2d825f6d-1fab-4fca-8b12-0ef1458e0d00",
  "sapId": "41f75e96-eb46-450c-946e-517becf5ec9a",
  "sapName": "default",
  "authtokenName": "Token1",
  "legalBasisId": "dcdf1e82",
  "legalBasisCode": "JAR1202A",
  "securityClass": 4,
  "version": 1
}
```

Minta kliens access token body Base64url kódolás nélkül (JWT claim) - version 2

```
{
  "jti": "d73ec0fc-ee56-40d9-b5aa-bd1746e2ba3c",
  "iss": "urn:sys:kkszb:fny",
  "sub": "urn:pid:kkszb:peer1",
  "type": "urn:token:kkszb:client:access",
  "iat": 1493113653,
  "nbf": 1493113653,
  "exp": 1493114253,
  "serviceId": "639b6a4236d65c06f6888a0e",
  "serviceUri": "/jarmu/service5/v1",
  "authtokenJti": "2d825f6d-1fab-4fca-8b12-0ef1458e0d00",
  "sapId": "639b6a4236d65c06f6888a0f",
  "sapName": "default",
  "authtokenName": "Token1",
  "legalBasisId": "639b6a4236d65c06f6888a0g",
  "legalBasisCode": "JAR1202A",
  "securityClass": 4,
  "version": 2
}
```

Táblázat 6. Az access token header részében szereplő kulcsok és leírásuk

Kulcs	Leírás
alg	A token aláírásához használt algoritmus (JWA). Jelenleg két algoritmus támogatott: ***RS256*** és ***ES256***
typ	A token típusa. ***JWT***
kid	Az aláíráshoz használt kulcs azonosítója, pl.: **1**

Táblázat 7. Access tokenben szereplő kulcsok és leírásuk

Kulcs	Leírás
jti	Az access token egyedi azonosítója, UUIDv4.
iss	A kibocsájtó egyedi azonosítója, , sztring, max. 100 karakter, URN formában, pl.: urn:sys:kkszb:fny
sub	A kérést küldő egyedi azonosítója, sztring, max. 100 karakter, URN formában, pl.: urn:pid:kkszb:xxxx.
type	A token típusa, , sztring, max. 100 karakter, URN formában, pl.: urn:token:kkszb:client:access
iat[1]	Kibocsájtás időpontja (POSIX time), numerikus, pl.: 1504703686
nbf[1]	Ezen időponttól érvényes a token (POSIX time), numerikus, pl.: 1504703686

Kulcs	Leírás
exp[1]	Ebben az időpontban jár le a token (POSIX time), numerikus, pl.: 1504703686 Megjegyzés: előfordulhat az az eset, hogy az *exp* időpont a szolgáltatáshoz történő beérkezéskor már elmúlt. Ez adódhat abból, hogy a szolgáltatás szerveren és az RFNY szerver időpontja eltér, valamint abból, hogy a KKSZB rendszeren történő áthaladás időt vett igénybe, és az ellenőrzéskor még nem érte el az *exp* időpontot. Az előzőek miatt a szolgáltatáson ezt az értéket nem kell ellenőrizni, arról a KKSZB gondoskodik.
serviceId	Az access token által elérhető szolgáltatás egyedi azonosítója (hash), string, max. 500 karkater. (v1: CouchDB ID, v2: MongoDB ID)
serviceUri	Az access token által elérhető szolgáltatás URI-ja, sztring, max. 201 karakter.
authtokenJti	Annak a kliens autentikációs tokennek az egyedi azonosítója, ami alapján ez az access token ki lett adva, UUIDv4.
sapId	A szolgáltatás elérési jogosultság egyedi azonosítója, sztring. (v1: CouchDB ID, v2: MongoDB ID)
sapName	A szolgáltatás elérési jogosultság neve, amely a könnyebb azonosíthatóságot és nyomon követhetőséget segíti, sztring, max. 30 karakter.
authtokenName	Az autentikációs token neve, szabadszöveges string, max. 20 karakter, például: az alkalmazás címkéje, amelyben felhasználásra kerül.
legalBasisId	Jogalap azonosító, mely v1 (CouchDB) esetén 8 karakter hosszú hexadecimális sztring, amelyet a KKSZB generál, sztring, hexadecimális érték, pontosan 8 karakter, pl.: a70d2dfd, v2 (MongoDB) esetén, Mongo által generált Id.
legalBasisCode	Jogalapkód, opcionális (ha nincs értéke, akkor a fejlécben nem szerepel), **amelyet az RFNY webes felületén a Szolgáltatás felelős ad meg, így közvetlenül használható a Szolgáltatást nyújtó alkalmazásban**, a következő karaktereket tartalmazhatja: a-z, A-Z, 0-9, -, _, / és . (pont) karakterek, max. 20 karakter.
securityClass	A szolgáltatást hívó fél kérésének IBTV biztonsági osztály értéke, numerikus érték 2-5 között (implicit), az AccessToken securityClass mezőjével egyezik meg, a Szolgáltatás fel tudja használni arra, hogy a kérést a biztonsági szintnek megfelelő útvonalra terelje. Értéke a Szolgáltatás Elérési Jogosultság Kérelem kitöltésekor kerül megadásra a Kapcsolódó fél által.
version	A token verziója, numerikus érték.

[1] Az iat, nbf és exp értéke 'POSIX time', másodperc pontossággal.

A HTTP fejlécek közül az **x-kk-sap-name** (sapName) és az **x-kk-token-name** (authtokenName) értékét a HTTP szabvány követelménye szerint *escape-elni* kell,

amelyre a KKSZB a javascript **encodeURI()** függvényét használja. A megadott karakterhosszúságok a dekódolt értékre vonatkoznak, a **kódolt értékek hosszabbak lehetnek.**

függelék C: x-kk-client-id szerkezete

A Szolgáltatások a HTTP kérés fejlécben megkapják az kliens access tokenben található **sub** mező értékét.

Ez a mező az alábbi formában épül fel:

Táblázat 8. Példa: x-kk-client-id = urn:pid:kkszb:bm szerkezete

urn:pid:kkszb	A KKSZB rendszerben a *peer-id* típus jelzője.
bm	A kapcsolódó fél egyedi azonosítója a KKSZB rendszerben, jelen esetben Belügyminisztérium.

függelék D: HTTP státusz kódok

A KKSZB kizárólag a HTTP státusz kódot használja a hiba típusának jelzésére.

Azt hogy a státuszkódot ki küldte - a szolgáltatás, vagy a KKSZB - a HTTP fejlécbe elhelyezett **x-kk-gw-status-message** alapján lehet eldönteni.

Amennyiben szerepel a **x-kk-gw-status-message** fejléc a válaszban, úgy a KKSZB-től származik az üzenet. Amennyiben ilyen nincs a fejlécben, úgy a szolgáltatás küldte azt.

Táblázat 9. KKSZB rendszer által visszaadott HTTP státusz kódok

HTTP státusz kód	x-kk-gw-status-message
200	Az üzenet feldolgozása sikeresen megtörtént.
A kérés nem tartalmazza a kötelező HTTP fejléc adatokat, ellenőrizze a programot	
400	"x-kk-authentication http header required"
400	"x-request-id http header required"
A beérkező HTTP kérés nem értelmezhető, nem felel meg a HTTP v1.1 szabvány követelményeinek	
400	"Invalid HTTP request"
Autentikációs hiba, ellenőrizze a kliens azonosító tokent	
401	"Authentication token refused"
KKSZB jogosultság hiba: a kért szolgáltatás elérésére nincs joga, vagy a szolgáltatás még nem elérhető	
403	"Permission denied"
429	"Too Many Requests", a Szolgáltatás Felelős korlátozást vezetett be, amelyben beállított a maximális kérés/perc értéket, és ezt a kérés számok túllépték.
KKSZB rendszer hiba	
500	Belső alkalmazás problémára vonatkozó üzenet, a kérést meg kell ismételni.
502	A Gateway nem tudja a kérést továbbítani a Szolgáltatás felé vagy az nem válaszol: a kérést meg kell ismételni, vagy az interfész leírás szerint kell eljárni. Tipikusan akkor fordul elő, ha a Szolgáltatás nem érhető el rosszul megadott belső URL cím miatt, vagy tűzfal beállítás miatt.[1]
503	A Szolgáltatlás vagy a KKSZB **átmeneti** elérhetetlenségére vonatkozó üzenet, a kérést meg kell ismételni.
Nincs válasz (timeout)	Timeout esetén a kliens nem kap választ (empty response, socket hang up, a hibaüzenet a Kliens implementációtól függ), így a HTTP státusz kód nem értelmezett.

[1] Rosszul megadott Szolgáltatás URL vagy tűzfal beállítás hiba miatt az alábbi választ látja

```
HTTP/1.1 502 Bad Gateway
Content-Type: text/plain
Content-Length: 14
x-kk-gw-status-message: Bad Gateway
Date: Mon, 16 Oct 2017 12:46:59 GMT

socket hang up
```

Megoldásként az RFNY webes felületén megadott belső URL címet javítani kell és ellenőrizze a Szolgáltatás oldalon a tűzfal beállításokat!

függelék E: Miért nem biztosít az ASZ tradicionális tranzakció menedzsmentet?

Az alábbi programrész egy - az erős konzisztenciára gyakran használt - *két fázisú kommit (2PC)* tranzakció példáján keresztül mutatja be a potenciális inkonzisztencia előállásának lehetőségét a saját és a távoli rendszer között. Amennyiben nincs kapcsolati hiba (hálózati hiba, timeout, stb), úgy a program hibátlanul működik, ugyanakkor ha ilyen előfordul, akkor az inkonzisztencia előállhat két rendszer között.

Az alábbi programban:

- előkészítjük a ASZ-szel a tranzakciót [1], majd
- meghívjuk a *prepare()* [2] funkciót, amely a tranzakció azonosítóval az *ASZ adatbázisba menti* az üzenetet (de még nem kézbesíti), a következő lépésben
- az ASZ tranzakción meghívjuk a *commit()* műveletet [3], amely azt jelenti, hogy az adott tranzakcióhoz tartozó (előzőleg adatbázisba mentett) *üzenetet kézbesítse.*

Adatbázis művelet és ASZ üzenetküldés 2 fázisú kommittal

```
ASZ_tr = tranzakcio_inditas; [1]
adatbazis_tr = tranzakcio_inditas;
try {
  uzenet_eloallitasa();
  adatbazis_tr.rekord_adatbazisba_irasa();
  ASZ_tr.statisztika_uzenet_kuldese();
  ASZ_tr.prepare() [2]
  adatbazis_tr.prepare()
  ASZ_tr.commit() [3]
  adatbazis_tr.commit();
} catch (hiba) {
  ASZ_tr.rollback() [4]
  adatbazis_tr.rollback();
}
```

Az inkonzisztencia nagy valószínűséggel előáll a távoli és a mi rendszerünk között, amennyiben a [3] pontban a *commit()* művelet eredménye ismeretlenné válik egy *kapcsolati hiba miatt,* akkor nem tudjuk, hogy a távoli ASZ kézbesítette-e az üzenetet, vagy nem kézbesítette, tehát **azt sem tudjuk eldönteni, hogy a saját rendszerünkben végrehajtsuk a tranzakciót, vagy ne hajtsuk végre.**

Tételezzük fel, hogy ezt a programrészt egy webes felületről egy felhasználó indította. Amennyiben az előbb említett *kapcsolati hiba* fennáll és a felhasználó egy hibajelzést kap vissza, akkor két dolgot tehet: ismétli a kérést és *esetlegesen* sikerül jól végrehajtania a műveletet, vagy nem ismétli többet, ekkor marad az inkonzisztens állapot.

A távoli inkonzisztencia nagyon veszélyes eset, mert nem derül ki a saját rendszerünkön belül - hiszen a mi rendszerünk önmagában konzisztens -,

> valamint a távoli rendszer is úgy gondolja, hogy ő konzisztens állapotban van a mi rendszerünkkel (kapott üzenetet, amit megfelelően feldolgozott, vagy nem kapott üzenetet).

A jobb érthetőség kedvéért a fenti esetre egy példát mutatunk be.

Példaként tegyük fel, hogy a járművünket akarjuk átíratni, amelynek végső eredménye, hogy megkapjuk a Törzskönyvet postán. A jármű rendszer és a törzskönyv gyártó rendszer két független rendszer, amelyek HTTP felületen webes szolgáltatásokkal kommunikálnak.

Kapcsolati hiba esetén a jármű rendszerbe rögzített adatokat *rollback()* művelettel visszagörgetjük, így a jármű átírása nem történik meg, ugyanakkor a Törzskönyv rendszer felé az *üzenetküldésünk eredményét nem ismerjük* (lehet hogy sikeres, lehet hogy nem) - jelen esetben tegyük fel, hogy a Törzskönyv rendszer azt *átvette és feldolgozta* az üzenetünket, a Törzskönyvet előállította.

- A Törzskönyvet postán megkapjuk, *szerintünk az autó átírás sikeres.*
- A Törzskönyv rendszer nem tapasztalt semmilyen hibát, minden üzemszerűen zajlott, *Törzskönyvet előállította.*
- A jármű rendszer érzékelte a kapcsolati hibát és a hiba miatt *rollback()* művelettel visszagörgette az adatbázist előző állapotba, a *jármű nem került átírásra.*

Mivel a fenti helyzetben senki sem gondolja úgy, hogy bármit is tennie kellene, az inkonzisztencia "örökre" fennmarad.

Mi történne, ha *kapcsolati hiba* esetén is *commit()* kerülne végrehajtásra? Ekkor a jármű átírás megtörténne a fenti esetben és nem lenne inkonzisztencia.

Ugyanakkor mi van akkor, ha a *commit()* művelet mellett, *valóban nem dolgozta fel sikeresen* a Törzskönyv rendszer az üzenetünket és nem gyártott Törzskönyvet. (Ne feledjük, hogy nem ismerjük az üzenet kézbesítés eredményét.)

Ekkor az alábbi helyzet áll elő:

- A Törzskönyvet sohasem kapjuk meg, *nem tudjuk mi történt, és felkeressük újra az ügyintézési pontot.*
- A Törzskönyv rendszert el sem érte az üzenet, esetleg elérte, de válaszában jelezte hogy nem megfelelő (ez sohasem érte el a Jármű rendszert), minden üzemszerűen zajlott, *Törzskönyvet nem állította elő.*
- A jármű rendszer érzékelte a kapcsolati hibát és ennek ellenére végrehajtotta a *commit()* műveletet, így a *jármű átírásra került.*

A fenti esetben a Jármű és a Törzskönyv rendszer szerint is minden a legnagyobb rendben van, mindaddig amíg az állampolgár nem reklamál.

A reklamáció után csak úgy lehet helyreállítani a problémát, ha a Jármű rendszer meggyőződik arról, hogy valóban nem történt Törzskönyv előállítás - ezt saját rendszeréből nem tudja, be kell kérdeznie a Törzskönyv rendszerbe - majd ezután *ismétli az üzenetküldést és csak azt, mert még egyszer nem hajtja végre az üzleti tranzakciót (átírást).*

Mint látható, a tradicionális tranzakció menedzsment megoldások nem alkalmasak a rendszerek közötti konzisztencia megtartására, vagy helyre állítására, ugyanakkor a garantált üzenetküldés megvalósításával biztosítható a konzisztencia, valamint az esetleges inkonzisztencia megszüntethető az üzenet újraküldésével (és megfelelő feldolgozásával).

függelék F: Aszinkron Szolgáltatás válasz

Maximális üzenet hossz elérésekor az átvitel megszakad, és **413 Payload Too Large** választ küld vissza a szerver, PAYLOAD_TOO_LARGE üzenetkóddal.

Amennyiben **nem** HTTP POST üzenet lett küldve, úgy a **405 Method Not Allowed** üzenettel válaszol az ASZ.

Az üzenet küldéskor vagy újraküldésekor, sikeres üzenet küldés esetén mindig **202 Accepted** választ kap a *küldő fél*, melynek jelentése: üzenet befogadva ASZ által, de nincs kézbesítve.

A feladható maximális üzenet hossz 10MB.

Minden ASZ oldalon befogadott kérésre az alábbi választ adja az Aszinkron Szolgáltatás.

Minden egyéb HTTP státuszkód a HTTP státusz kódok függelék szerint kezelendő a *küldő fél* oldalán.

Táblázat 10. ASZ üzenet állapot válasz tartalma

Mezőnév	Típus	Leírás
messageId	string, UUIDv4	Az üzenet egyedi azonosítója, amelyet a KKSZB ASZ képez.
requestId	string, UUIDv4	Az üzenet küldéskor használt egyedi azonosító (x-request-id).
created_ts[1]	number	Üzenet befogadás időpontja ASZ oldalon.
producerPeerId	string	Az üzenet küldőjének azonosítója.
consumerPeerId	string	Az üzenet fogadójának azonosítója.
serviceId	string	A szolgáltatás egyedi azonosítója.
serviceUri	string	A szolgáltatás URI-ja.

[1] Az értéke 'POSIX time', ezredmásodperc pontossággal.