**ASP adattárház**

**Interfész specifikáció**

Készült az ASP2 Adattárház alprojekthez

Betöltési prototípus I.

**Dokumentum kontroll**

| Verzió | Dátum | Státusz | A MEGVÁLTOZOTT RÉSZEK, a módosítás leírása | Készítő, Módosító |
| --- | --- | --- | --- | --- |
| 0.1 | 2018.03.07. | Dokumentum vázlat | Első verzió | Erdey Levente, Frunza Zsolt, Kurdi Márió, Szijártó András |
| 0.2 | 2018.03.09. | Dokumentum vázlat | Első jegyzetek | Kurdi Márió |
| 0.3 | 2018.03.14 | Dokumentum vázlat | Fájlformátumok, Névkonvenciók, Karakterkészlet, Szeparátorok, Naplófájl tartalma, Tömörítés | Mészáros Tamás |
| 0.4 | 2018.03.21 | Dokumentum vázlat | 2., 3., 4. fejezetek bizonyos részeinek feltöltése | Kurdi Márió, Mészáros Tamás |
| 0.5 | 2018.04.03 | Első átadandó verzió | GAZD űrlapok, KTÖRZS adatok véglegesítése, további fejezetek | Erdey Levente, Kurdi Márió, Mészáros Tamás |
| 1.0 | 2018.04.01 | Átadásra kész | Több fejezet | Kurdi Márió |
| 1.1 | 2018.04.17 | Átadásra kész | Észrevételek alapján történt pontosítások, több fejezet | Kurdi Márió |
| 1.2 | 2018.05.04 | Adott verzió draft-ja | IPARKER szálláshely adatok köre | Kurdi Márió, Szijártó András |
| 1.2.1 | 2018.05.10 | Adott verzió draft-ja | GAZD kötelezettségvállalás adatok köre | Kurdi Márió, Szijártó András |
| 1.3 | 2015.05.17 | Átadásra kész | Észrevételek alapján történt pontosítások | Kurdi Márió, Szijártó András |
| 1.4 | 2018.05.18 | Javított | Apróbb pontosítások | Kurdi Márió |
| 1.5 | 2018.07.05 | Átadásra kész | 3.2.4 fejezet új meta3.2.4 fejezet pontosítása | Szijártó András |
| 1.5.1 | 2018.08.01 | Adott verzió draft-ja | ADÓ szakrendszerrel bővítve | Szijártó András |
| 1.5.2 | 2018.08.11 | Adott verzió draft-ja | ADÓ szakrendszerrel módosítása | Szijártó András |
| 1.6 | 2018.08.14 | Átadásra kész | Adó szakrendszer | Szijártó András |
| 2.0 | 2018.09.10 | Átadásra kész | Rendszerspecifikus részek külön dokumentumba áthelyezése | Szijártó András |
| 2.1 | 2018.09.13 | Javított | Apróbb pontosítások | Kurdi Márió |
| 2.2 | 2018.09.20 | Javított | Adó részekkel kiegészítve, pontosítások | Szijártó András, Kurdi Márió |

Tartalom

[1. Bevezetés 5](#_Toc528740735)

[1.1 Dokumentum célja 5](#_Toc528740736)

[1.2 Használt rövidítések 5](#_Toc528740737)

[1.3 Fogalomtár 5](#_Toc528740738)

[2. Adatátadás logikai folyamata 6](#_Toc528740739)

[3. Általános követelmények 9](#_Toc528740740)

[3.1 Formai követelmények 9](#_Toc528740741)

[3.1.1 Fájlformátumok 9](#_Toc528740742)

[3.1.2 Névkonvenciók 10](#_Toc528740743)

[3.1.2.1 Csomag neve 10](#_Toc528740744)

[3.1.2.2 Állományok 13](#_Toc528740745)

[3.1.2.3 Naplófájlok 14](#_Toc528740746)

[3.1.3 Csomagverziók 15](#_Toc528740747)

[3.1.4 Karakterkészlet 15](#_Toc528740748)

[3.1.5 Szeparátorok 15](#_Toc528740749)

[3.2 Tartalmi követelmények 16](#_Toc528740750)

[3.2.1 Rendezés 16](#_Toc528740751)

[3.2.2 Deperszonalizáció 16](#_Toc528740752)

[3.2.3 Ellenőrzések 17](#_Toc528740753)

[3.2.4 Naplófájl és kísérő metaadatok 18](#_Toc528740754)

[3.2.5 Checksum képzése 20](#_Toc528740755)

[3.2.6 Üres csomagok feladása 22](#_Toc528740756)

[3.3 Tömörítés 22](#_Toc528740757)

[3.4 Titkosítás 22](#_Toc528740758)

[3.5 Vírusellenőrzés 22](#_Toc528740759)

[3.6 Küldés 23](#_Toc528740760)

[3.6.1 UploadDocument 23](#_Toc528740761)

[3.6.2 GetDocumentStatus 24](#_Toc528740762)

[3.6.3 GetDocumentStatusByFilename 24](#_Toc528740763)

[3.6.4 GetDocumentErrorLog 24](#_Toc528740764)

[3.7 Hibakezelés 24](#_Toc528740765)

[3.8 Visszajelzés 24](#_Toc528740766)

[4. Tesztelés és próbaüzem 25](#_Toc528740767)

[4.1 Csatlakozási (smoke) teszt 25](#_Toc528740768)

[4.2 Betöltési prototípus tesztek 26](#_Toc528740769)

[5. Ütemezés 27](#_Toc528740770)

[6. Mellékletek 28](#_Toc528740771)

[6.1 Üzleti specifikáció 28](#_Toc528740772)

[6.2 Egyszerűsített „E” űrlap koncepció 29](#_Toc528740773)

[6.3 Mintaállományok 29](#_Toc528740774)

[6.3.1 Gazdálkodási szakrendszer - Űrlap adatok adatköre mintaállomány 29](#_Toc528740775)

[6.3.2 Ipar- és kereskedelmi rendszer - Szálláshely adatok adatköre mintaállomány 29](#_Toc528740776)

[6.4 Az állományok feltöltéséhez szükséges szolgáltatás definíciója 29](#_Toc528740777)

[6.5 „A” űrlap 29](#_Toc528740778)

# Bevezetés

## Dokumentum célja

A dokumentum célja a Magyar Államkincstár KÖFOP-1.0.0.-VEKOP-15-2016-00008 azonosító számú „Az önkormányzati ASP rendszer továbbfejlesztése és országos kiterjesztése (ASP 2.0)” projekt önkormányzati ASP Adattárház alprojektjén belül megvalósuló adattárházhoz csatlakozó interfészek specifikálása.

Jelen specifikációban megfogalmazott követelmények vonatkoznak mind az ASP rendszer szakrendszereire, mind pedig az ASP használata alól felmentést kapó, interfészen keresztül csatlakozó önkormányzatok lokális szakrendszereire.

A dokumentáció végleges verziója minden interfész specifikációjának általános leírását tartalmazza. A köztes verziók ezek egy egyre bővülő részhalmazát képezik. A dokumentumhoz tartozik minden szakrendszer esetében egy rendszerspecifikus kiegészítés, amely az adott szakrendszerben lefejlesztendő konkrét interfész pontos leírását tartalmazza.

## Használt rövidítések

|  |  |
| --- | --- |
| **Rövidítés** | **Jelentés** |
| ASP | Application Service Provider |
| COFOG | Classification of Functions of Government (Kormányzati funkciók, államháztartási szakfeladatok és szakágazatok osztályozási rendje) |
| EIP | Egységes Integrációs Platform |
| ITB | Információtechnológiai Biztonság |
| IVK | Ingatlan-vagyonkataszter |
| KGR | Költségvetési Gazdálkodási Rendszer |
| NISZ | Nemzeti Infokommunikációs Szolgáltató Zrt. |
| WSDL | Web Services Description Language (webszolgáltatás-leíró nyelv) |
| XML | Extensible Markup Language (kiterjeszthető jelölő nyelv) |

## Fogalomtár

|  |  |
| --- | --- |
| **Rövidítés** | **Jelentés** |
| metaadat | Adat leíró adat. |
| deperszonalizáció | Személytelenítés, az egyedi azonosításra alkalmas adatok maszkolása. |
| data lake | Adattárolási módszer  |
| deltaképzés | A legutóbbi leválogatás óta rögzült, változott, törölt adatok leválogatása. |

# Adatátadás logikai folyamata

Az adattárház számára szolgáltatandó adatok előállításának, küldésének, fogadásának és betöltésre előkészítésének egy jól megtervezett, szigorúan követendő folyamat mentén kell megtörténnie. Az alábbi ábra az ASP szakrendszer és az egyes beszállítók, valamint az adattárház közötti adatátadást, továbbá az érkező adatok ellenőrzési folyamatának lépéseit mutatja be az adatok rendszerenkénti leválogatásától az EIP (Egységes Integrációs Platform) feladásig szállító oldalról, továbbá az EIP fogadástól a beérkező adatok feldolgozásáig az ASP adattárház oldalán.

Az egyes ellenőrzéseknél a vírusellenőrzés kivételével az ’Y’ jelenti az ellenőrzés pozitív kimenetelét, az ’N’ pedig a sikertelen ellenőrzést. Amennyiben egy ellenőrzés sikertelen eredménnyel zárul, úgy a folyamat egy korábbi pontjától kell újra elvégezni a lépéseket. A vírusellenőrzés esetében az ’Y’ kimenetel vírus felfedezését jelenti, az ’N’ pedig azt, hogy a keresés nem talált kártékony elemet.



1. ábra – Adatátadás logikai folyamat

Az adatátadás és az átadott adatok ellenőrzésének és betöltésre előkészítésének folyamata két részre bontható. Az első felében a feladó (szállító) oldali lépések, a második felében a fogadó (adattárház) oldali teendők sorozata található.

A folyamat első lépéseként szakrendszeri oldalon megtörténik a szakrendszerenként külön tárgyalt adatok rendszerenkénti leválogatása. Ezt követően az adatok a szakrendszerenként meghatározott ellenőrzéseken kell, hogy keresztülmenjenek.

A sikeres ellenőrzések után történik az adatok kiöntése, azaz azok állományokba kiírása. Ezek az állományok lesznek a deperszonalizációs eljárás inputjai.

A sikeres kiöntést követi az adatok deperszonalizációja. Az ASP adattárház adatvagyona nem tartalmazhat személyes adatnak vagy adótitoknak minősülő adatot. Ennek érdekében a 3.2.2 fejezetben kifejtett deperszonalizációnak kell alávetni a beérkező adatok egy részét. Deperszonalizálni csak az előre definiált adatokat kell, de léteznek olyan adatkörök is, melyek nem érintettek ezzel. A deperszonalizáló programot az ASP projekt központilag kifejleszti és átadja az önkormányzatok részére.

A központilag fejlesztett deperszonalizációs eljárás az adatokon bizonyos formai követelményeket ellenőriz annak érdekében, hogy apró formai különbségek ne rontsák a deperszonalizált adatok felhasználhatóságát. Az algoritmus formai ellenőrzésein fennakadt állományok nem deperszonalizálhatóak.

Az adatfájlok deperszonalizálását követi azok formátumának ellenőrzése. Az adatfájlok .dat kiterjesztésű fájlok, melyek soronként tartalmaznak egy-egy rekordot, soron belül pedig az egyes adatokat | jel szeparálja. Nem lehet az állományban az adatmezőn belül | szeparátor vagy sortörés, hiszen ez az oszlopok és/vagy sorok elcsúszását eredményezné. Az adatfájlokkal szembeni formátumkövetelményeket a 3.1 fejezetben olvashatóak.

A formátumellenőrzést egy .log kiterjesztésű naplófájl előállítása követi. Ez a fájl az adatfájlokra vonatkozó adatokat tartalmaz. A naplófájl tartalmára vonatkozó elvárások a 3.2.4 fejezetben részletesen kifejtésre kerülnek.

Miután az adatfájlok és a naplófájl sikeresen elkészült, ezek egy tar.gz kiterjesztésű tömörített állományba kerülnek összecsomagolásra. A tömörített fájl nem tartalmazhat könyvtárat, tartalma a .dat kiterjesztésű adatfájlok és az adatfájlok tartalmára vonatkozó .log kiterjesztésű naplófájl. A tömörített fájllal kapcsolatos elvárások a 3.3 fejezetben olvashatóak.

A tömörített fájl előállítását követően a létrehozott csomag metaadatai kerülnek előállításra. Ezek az archívumra vonatkozóan tartalmaznak információkat (pl.: fejlesztő, leválogatás időpontja, checksum stb.). A metaadatokkal szembeni tartalmi és formai követelmények a 3.2.4 fejezetben kerülnek részletesen tárgyalásra.

A tömörített állomány és a metaadatok sikeres létrehozása után megtörténhet a feladás az ASP adattárház felé. A feladás a 3.6 fejezetben leírt szolgáltatás fájl feltöltés műveletének meghívásával történik. A hívást a feladó oldalon fejlesztett interfész kezdeményezi.

A folyamat második felében az ASP adattárház fogadja a tömörített állományt a szállítói oldalról.

Első lépésben a metaadatok alapján az adattárház ellenőrzi a tömörített állományt. A metaadatokhoz kapcsolódó ellenőrzésekkel kapcsolatos elvárásokat a 3.2.4 fejezet tartalmazza. A metaadat alapú ellenőrzés tartalmazza többek között a duplikátumellenőrzést és a feladó a feladott adatok elküldésére való jogosultságának ellenőrzését. Ezen ellenőrzések részleteit szintén a 3.2.4 fejezet tartalmazza. Sikertelen metaadat ellenőrzés esetén az adatokat tartalmazó állomány létrehozását a tömörítéstől kell újrakezdeni. A metaadat alapú ellenőrzés sikeressége esetén a csomag mentésre kerül a DR site-on is.

Következő lépésként az adattárház egy vírusellenőrzést hajt végre a bejövő állományon. A vírus ellenőrzés folyamata a 3.5 fejezetben részletesen kifejtésre kerül. Abban az esetben, ha a vírusellenőrző kártékony szoftvert vagy komponenst talál, úgy a kiadó szerver vírus ellenőrzése szükséges, majd a kiöntés fázisától kell újra elkezdeni az adatokat tartalmazó csomag előállítását.

Amennyiben a vírusellenőrzés nem talált kártékony kódot, vagy komponenst, ellenőrzésre kerül a tömörített fájl formátuma, valamint a checksum értéke. Ha nem megfelelő (.tar.gz) formátumot vagy a checksum-ot találunk, úgy a tömörítési fázistól szükséges újrakezdeni a fájlok előállítását. Ha az ellenőrzés sikeres, a tömörített állományt az adattárház kicsomagolja.

A kicsomagolást követően előáll egy vagy több .dat kiterjesztésű adatfájl és egy darab .log kiterjesztésű naplófájl az adatfájlokra vonatkozó információkkal. Az adatfájlok és naplófájlok formátumai szintén ellenőrzésre kerülnek. Amennyiben nem felelnek meg a 3.2.4 fejezetben foglalt formátumkövetelményeknek, úgy a kiöntési fázistól kell újrakezdeni a fájlok előállítását. Sikeres formátum ellenőrzés után következik a naplófájl alapú ellenőrzés.

A kicsomagolt adatfájlokon az adattárház végrehajtja a naplófájl alapú ellenőrzéseket a naplófájlban tartalmazott metaadatok alapján. Minden adatfájlon történik egy sorszám ellenőrzés, valamint további ellenőrzések, melyek a 3.2.3 fejezetben kerülnek részletes kifejtésre. Sikertelen ellenőrzés esetén az adattárház visszajelez a forrás oldalnak a hiba pontos leírásával, a fájlok pedig nem kerülnek betöltésre. Sikeres naplófájl alapú ellenőrzés után megtörténik az adatok betöltése a Hadoop keretrendszerbe. A betöltés során az adatok először változatlan formában kerülnek be az adattárház data lake részébe. A data lake-ben a bejövő adatokon további ellenőrzések futnak. Itt történik annak vizsgálata, hogy az adatok üzleti szempontból megfelelnek-e a tartalmi követelményeknek. A rendszerspecifikus követelményeket szakrendszerenként külön dokumentum tárgyalja részletesen. A data lake ellenőrzések sikertelensége esetén az adatok előállítását, a szállító oldalán, a rendszerenkénti leválogatástól szükséges újrakezdeni.

A data lake rétegbeli ellenőrzések sikeres végkimenetele után megkezdődhet az adatok az ASP adattárház strukturált rétegeiben történő feldolgozása.

# Általános követelmények

Elsőként az adattárház interfészeire vonatkozó általános követelmények kerülnek részletezésre. Jelen fejezet tartalma vonatkozik minden interfészre, kivéve, ha a rendszerspecifikus követelményekben leírtak erről külön másképp rendelkeznek. Ebben az esetben a vonatkozó rendszerspecifikus követelményei az elsődlegesnek tekintendők.

Az ASP adattárházhoz kapcsolódó minden forrásrendszernek és interfésznek meg kell felelnie a hatályos az önkormányzati ASP rendszerről szóló 257/2016. (VIII. 31) Korm. rendelet (a továbbiakban: ASP rendelet) elvárásainak minden szempontból.

A lokális szakrendszereknek képesnek kell lenniük delta állományok előállítására az interfészekben definiált, deltaképzéssel érintett adatkörök esetében. A szakrendszeri leválogatónak biztosítani kell ennek teljességét, zártságát, azaz azt, hogy adott alkalommal a leválogató eljárások pontosan azokat a rekordokat válogassák le, melyek a legutóbbi adatátadás óta keletkeztek vagy módosultak. Bizonyos adatkörök leválogatás alapelve nem a deltaképzés, erről a rendszerspecifikus dokumentáció tartalmaz pontos információkat.

Az interfészekkel szemben elvárás, hogy azok kialakítása a lehető legrugalmasabb legyen annak érdekében, hogy a későbbiekben felmerülő módosítások, bővítések gyorsan és gördülékenyen kivitelezhetőek legyenek. Ennek biztosítása érdekében az interfészeket paraméterezhető módon kell kialakítani. Ez azt jelenti, hogy lehetőség szerint a leválogató eljárások ne adatkör-, tábla-, mezőspecifikusan készüljenek, hanem egy általános eljárás készüljön, mely egy vezérlőtábla alapján készíti el az aktuálisan szükséges leválogatásokat. Ebben a vezérlőtáblában kell szerepelnie a leválogatás minden paraméterének (pl.: csomag, tábla/nézet, oszlopok, a deltaképzés alapjául szolgáló dátum stb.). Ezzel a megoldással a módosítások, bővítések esetén elegendő egy leválogató nézet elkészítése, valamint a vezérlőtáblába a szükséges új paraméterek felvitele, amivel az eljárástól elvárt rugalmasság biztosítható.

Az interfészeket úgy kell kialakítani, hogy üzemszerű működésük alatt teljesen automatizáltan, emberi beavatkozás nélkül fussanak a meghatározott időpontokban, beavatkozásra csak rendellenesség esetén legyen szükség.

Formai követelmények

Az alábbi fejezet a küldendő állományok általános, formai követelményeit írja le.

### Fájlformátumok

Az adatátadáshoz használt tömörített fájlok elvárt formátuma:

*fájlnév*.**tar.gz**

A tömörített állomány által tartalmazott, adatfájlok alapvetően text állományok, melyek elvárt formátuma:

*fájlnév*.**dat**

A .dat fájlok mellé szükséges egy kísérőfájl. A kísérőfájl egy text állomány, elvárt formátuma:

*fájlnév*.**log**

### Névkonvenciók

Az interfészen érkező állományok csomagokba történő rendezése 5 jellemző szerint történik:

* szakrendszer (pl.: GAZD, IPARKER, ADO, IVK stb.),
* adatkör (pl.: űrlap adatok, kötelezettségvállalások adatai, szálláshely szolgáltatók adatai stb.),
* PIR-szám (azé az önkormányzaté vagy bizonyos ASP-s esetekben intézményé, amelynek az adatai (és intézményeinek adatai) szerepelnek a csomagban)
* fejlesztő (ASP vagy adott interfészes önkormányzat, illetve szakrendszerének szállítója/fejlesztője, aki a leválogató programot kidolgozza),
* időszak (amely időszakra a csomagban lévő adatok vonatkoznak).

Ez a tagolás jelenik meg az egyes csomagok elnevezésében is a beazonosíthatóság érdekében. A csomag nevének egyértelműen definiálni kell a tartalmát és fordítva, a nevezéktan kialakításának ez az egyik alapvető szempontja. Ezen felül a fejlesztő kivételével a többi információ minden állomány első 5 oszlopában is meg kell, hogy jelenjen, lásd a 6.1-es üzleti specifikációt tartalmazó mellékletben.

A csomagokon belüli állományok egy-egy logikai entitásnak (adott esetben akár fizikai táblának) felelnek meg. Így egy csomag az adott adatkörhöz tartozó entitások állományait, valamint egy naplófájlt tartalmaz.

Az egyes fájlnevek számokat, alulvonást (underscore ASCII CODE: 95), az angol abc betűit és **kizárólag kisbetűket** tartalmazhatnak. Ezek alapján az alábbi névkonvenciók alkalmazandók.

#### Csomag neve

A forrásrendszerek által feladandó, tömörített állományok nevénél az alábbi névkonvenció alkalmazandó. Állomány neve:

[szakrendszer]\_[adatkör]\_[pir]\_[fejlesztő]\_[időszak\_tól]\_[időszak\_ig].tar.gz

ahol,

**Szakrendszer**:

| **Szakrendszer kódja** | **Szakrendszer neve** |
| --- | --- |
| keret | ASP szakrendszeri portfólió keret alkalmazása |
| gazd | Gazdálkodási rendszer |
| ado | Adóügyi rendszer |
| ivk | Ingatlan-vagyonkataszter |
| irat | Iratkezelő rendszer |
| iparker | Ipar- és kereskedelmi rendszer |
| ktorzs | Kincstár Közhiteles Törzskönyvi Nyilvántartása |
| onegm | Önkormányzati Előirányzat Modul (Kincstár) |

1. táblázat – Szakrendszer kódok

**Adatkör** (bővítés alatt):

|  |  |  |
| --- | --- | --- |
| **Adatkör kódja** | **Adatkör neve** | **Szakrendszer** |
| urlapheti | KGR-K11 űrlapok szabályai szerint képzett heti aggregátumok | gazd |
| urlaphavi | KGR-K11 űrlapok szabályai szerint képzett havi aggregátumok | gazd |
| kotval | Kötelezettségvállalások, más fizetési kötelezettségek | gazd |
| koveteles | Követelések adatai | gazd |
| penzforg | Pénzforgalmi adatok | gazd |
| eszkoz | Eszköz adatok | gazd |
| kozos | Közös, több gazdálkodási adatkört érintő táblák | gazd |
| szallas | IPARKER rendszer szálláshelyek adatai | iparker |
| piac | IPARKER rendszer piac adatai | iparker |
| rendezveny | IPARKER rendszer rendezvény adatai | iparker |
| telep | IPARKER rendszer telephely adatai | iparker |
| uzlet | IPARKER rendszer üzlet adatai | iparker |
| torzs | Adó szakrendszer adóalany nyilvántartása | ado |
| bevallas | Adó adatkör bevallás adatai | ado |
| irat | Irat adatok | ado |
| penzugy | Pénzügyi adatok | ado |
| osszesito | Összesítő adatok | ado |

2. táblázat – Adatkör kódok

**PIR**: annak a szervezetnek (önkormányzat, intézmény, nemzetiségi önkormányzat, társulás, költségvetési szerv stb.) a PIR-száma, amely rendszerének az adatai szerepelnek a csomagban. Adott esetben egy csomagban több szervezet adatai is szerepelhetnek (az egyes csomagokra vonatkozó tartalmi elvárásokat lásd külön dokumentációban). Ilyenkor:

* az ASP esetében az adott tenantért felelős szervezet (önkormányzat, intézmény, nemzetiségi önkormányzat, társulás, költségvetési szerv stb.) PIR-száma,
* interfészes önkormányzat esetében az adott (1254-es szektorkódú) önkormányzat PIR-száma, aki a csomag feladásáért felelős, abban az esetben is, ha a csomag az önállóan gazdálkodó intézményre vonatkozó adatokat is tartalmaz.

**Fejlesztő**: ASP esetén ’asp’, interfészes önkormányzat esetén annak PIR-száma vagy szállítója esetén a szállító nevéből képzett azonosító az alábbiak szerint.

| **Fejlesztő kódja** | **Fejlesztő neve** |
| --- | --- |
| asp | ASP szakrendszerek esetén |
| PIR szám (a fejlesztő interfészes önkormányzaté) | Interfészes önkormányzat közvetlen fejlesztése esetén |
| citybyte | CityByte Zrt. |
| computrend | CompuTREND Zrt. |
| eszoftver | E-Szoftverfejlesztő Kft. |
| griffsoft | GriffSoft Informatikai Zrt. |
| komunal | Komunáldata Számítástechnikai Fejlesztő és Szolgáltató Kft. |
| korend | Korend Rendszerház Kft. |
| mak | Magyar Államkincstár |
| kozig | Magyar Közigazgatásfejlesztési Zrt. |
| misoftware | MI Software Kft. |
| nyirinfo | Nyírinfó Nonprofit Kft. |
| prof | Professzionál Informatikai Zrt. |
| rudaskarig | Rudas & Karig Számítástechnikai Kft. |
| saldo | SALDO Pénzügyi Tanácsadó és Informatikai Zrt. |
| tsystems | T-Systems Magyarország Zrt. |
| visualaid | VisualAid Szoftver Kft |
| zalaszam | Zalaszám Kft. |
| ulyssys | ULYSSYS Számitástechnikai Fejlesztő és Tanácsadó Kft. |

3. táblázat – Fejlesztő kódok

**Időszak\_tól**: az állományokban lévő adatok időszakának kezdő dátuma a csomag nevében YYYYMMDD formátumban.

**Időszak\_ig**: az állományokban lévő adatok időszakának vég dátuma a csomag nevében YYYYMMDD formátumban.

Amennyiben egy napi delta állományt kapunk, az időszak\_tól dátum megegyezik az időszak\_ig dátummal.

**Példák**:

Az ASP által a KGR-K11 űrlapok alapjául szolgáló aggregátumok 2018. szeptemberi előállításának állománya Bodajk önkormányzatának esetében az alábbi névvel kell, hogy érkezzen:

gazd\_urlaphavi\_362159\_asp\_20180101\_20180831.tar.gz

Az ASP által a KGR-K11 űrlapok alapjául szolgáló aggregátumok 2018. szeptemberi előállításának állománya az Ágfalvi Napsugár Óvoda, mint külön tenantban szereplő önállóan gazdálkodó szervezet speciális esetében:

gazd\_urlaphavi\_642211\_asp\_20180101\_20180831.tar.gz

Ugyanez az adatkör győri önkormányzat által feladva:

gazd\_urlaphavi\_727705\_zalaszam\_20180101\_20180831.tar.gz

Az ASP által a kötelezettségvállalások adatköreinek tételes adatai Bodajk esetében 2018.09.02-én:

gazd\_kotval\_362159\_asp\_20180902\_20180902.tar.gz

Ugyanerre a napra és önkormányzatra az ASP IPARKER szakrendszer szálláshelyszolgáltatókat tartalmazó adatköre:

iparker\_szallas\_362159\_asp\_20180902\_20180902.tar.gz

Ugyanerre a napra és önkormányzatra az ASP ADO szakrendszer törzsadatokat tartalmazó adatköre:

ado\_torzs\_362159\_asp\_20180902\_20180902.tar.gz

#### Állományok

A forrásrendszerek által feladandó, tömörített csomagokban található, adatokat tartalmazó állományok nevénél az alábbi névkonvenció alkalmazandó.

Állomány neve:

[szakrendszer]\_[adatkör]\_[pir]\_[állomány]\_[fejlesztő]\_[időszak\_tól]\_[időszak\_ig].dat

ahol,

**Szakrendszer**: lásd 3.1.2.1 fejezet.

**Adatkör**: lásd 3.1.2.1 fejezet.

**PIR**: A közvetlenül az adatkör neve után szereplő PIR-szám jelentése és képzése megegyezik a 3.1.2.1 fejezetben leírtakkal. (Adott esetben előfordulhat, hogy ez után, mint állománynév újabb PIR-szám szerepel, lásd 4. táblázat).

**Állomány**: lásd a rendszerspecifikus dokumentációk üzleti mellékletében.

**Fejlesztő**: lásd 3.1.2.1 fejezet.

**Időszak\_tól**: az állományokban lévő adatok időszakának kezdő dátuma a fájl nevében YYYYMMDD formátumban.

**Időszak\_ig**: az állományokban lévő adatok időszakának vég dátuma a fájl nevében YYYYMMDD formátumban.

**Példák**:

Az ASP által a KGR-K11 űrlapok alapjául szolgáló aggregátumok 2018. szeptemberi előállításának állományában a Bodajk önkormányzat adatai alábbiak névvel kell, hogy érkezzen:

gazd\_urlaphavi\_362159\_362159\_asp\_20180101\_20180831.dat

A két PIR-szám azért szükséges, mert jelen esetben az első azt jelzi, hogy a bodajki önkormányzat tenantjának adatairól van szó, a második pedig azt, hogy ezen belül is a magának az önkormányzatnak az űrlap adatairól. Az ugyan ehhez a tenanthoz tartozó, így ugyanebben a csomagban érkező Bodajki Zengő Óvoda (PIR: 834247) állományának neve:

gazd\_urlaphavi\_ 362159\_834247\_asp\_20180101\_20180831.dat

Ugyanez az adatkör a győri önkormányzat által feladva a saját adataira:

gazd\_urlaphavi\_727705\_727705\_zalaszam\_20180101\_20180831.dat

Ugyanez az adatkör a Győri Nemzeti Színházra (PIR: 467012) győri önkormányzat által feladva:

gazd\_urlaphavi\_727705\_467012\_zalaszam\_20180101\_20180831.dat

Az ASP IPARKER szakrendszere általa a szálláshely szolgáltatók adatkörének szálláshelyeit, üzemeltetőit, eseményeit, valamint forgalmi adatait tartalmazó állományai esetében szintén Bodajkra, egy példa napra az alábbi fájlokat kell előállítani:

iparker\_szallas\_362159\_szhely\_asp\_20180909\_20180909.dat

iparker\_szallas\_362159\_uzem\_asp\_20180909\_20180909.dat

iparker\_szallas\_362159\_esemeny\_asp\_20180909\_20180909.dat

iparker\_szallas\_362159\_forg\_asp\_20180909\_20180909.dat

Ugyanezek az adatkörök győri önkormányzat adataira a Komunáldata által feladva:

iparker\_szallas\_727705\_szhely\_komunal\_20180909\_20180909.dat

iparker\_szallas\_727705\_uzem\_komunal\_20180909\_20180909.dat

iparker\_szallas\_727705\_esemeny\_komunal\_20180909\_20180909.dat

iparker\_szallas\_727705\_forg\_komunal\_20180909\_20180909.dat

#### Naplófájlok

A forrásrendszerek által feladandó, tömörített csomagokban található, adatokat tartalmazó fájlokhoz kapcsolódó naplóállomány nevénél az alábbi névkonvenció alkalmazandó.

Naplóállomány neve:

[szakrendszer]\_[adatkör]\_[pir]\_naplo\_[fejlesztő]\_[időszak\_tól]\_[időszak\_ig].**log**

A naplófájlok elnevezésénél a szakrendszer, adatkör, pir, időszak\_tól, időszak\_ig részek megegyeznek a fentebb leírtakkal. A ’naplo’ fix érték.

### Csomagverziók

Esetenként előfordulhat, hogy ugyanazt a csomagot több alkalommal is el kell küldenie a feladónak (pl.: sérült, hibás csomag vagy állomány(ok) esetében). Ebben az esetben a teljes csomagot (minden benne lévő állományt) újra kell küldeni úgy, hogy az újraküldött csomag nevének végére tett sorszámmal kell megkülönböztetni őket a következők szerint.

| **Csomag verziója** | **Csomag neve** |
| --- | --- |
| Eredeti csomag | [eredeti\_csomagnév].tar.gz |
| 1. javítás | [eredeti\_csomagnév]\_1.tar.gz |
| 2. javítás | [eredeti\_csomagnév]\_2.tar.gz |
| … | … |
| n. javítás | [eredeti\_csomagnév]\_[n].tar.gz |

4. táblázat – Csomagverziók

A csomagon belüli állományok esetében a verziószám alkalmazása nem szükséges.

Hibás csomag esetében a javított csomagnak a lehetőségekhez képest a leghamarabb meg kell érkeznie, ugyanis a következő időszakra vonatkozó csomag addig nem kerül elfogadásra, amíg a megelőző javítása be nem töltődik.

### Karakterkészlet

Az átadandó állományok kizárólagos megengedett karakterkészlete az **UTF-8**. Betöltés előtt adattárház oldalon ellenőrizzük a karakterkészletet, és a nem megfelelő csomagokat nem töltjük be az adattárházba, ezek hibaüzenettel térnek vissza.

### Szeparátorok

A forrásrendszerek által feladandó, tömörített csomagokban található, adatokat tartalmazó állományok (fájlok) egyes soraiban a következő szeparátorok használata a megengedett:

Az egyes rekordok mezőit a **|** (függőleges vonal (vertical bar), ASCII CODE 124) karakter választja el egymástól, példa:

adat1|adat2|adat3

Üres értéket pedig úgy kell feladni, hogy a két szeparátor jel közé nem kell semmit sem tenni (nem kell sem szóköz, sem idézőjelek, sem a null szöveges kiírása). Példa:

||

Fontos, hogy ez a karakter megmaradjon az állományokban egyedi, csak szeparálásra használt speciális karakternek, így amennyiben valamely, az állományokban lévő szabad szöveges (vagy bármilyen más) mezőben előfordulnak ilyenek, akkor azokat az állományok előállítása előtt le kell cserélni valamilyen másik karakterre (például pontosvesszőre). Továbbá fontos, hogy a szabadszöveges mezőkben ne forduljon elő sortörés nem, az csak a sorok végén szerepelhet. Az ilyen típusú problémáknak az állományok formátumellenőrzésénél ki kell derülnie.

A rekordokon belüli numerikus adatokban szereplő esetleges tizedesjegyeket ’**.**’ (pont, ASCII CODE 46) jelöli (a magyar tizedesvesszős jelöléstől való eltérésre a Hadoop sajátosságai miatt van szükség), példa:

adat1|3.1415|1.33|adat4

A rekordokon belüli adatokban szereplő esetleges felsorolásoknál az egyes elemek elválasztására a ’**,**’ (vessző, ASCII CODE 44) használandó, példa:

adat1|adat21,adat22,adat23|adat3

adat1|2.1,2.2,2.3|adat3,adat4

A forrásrendszerek által feladandó állományokkal kapcsolatos naplófájlok tartalmi elemeire ugyanezek a szeparátorokkal kapcsolatos elvárások vonatkoznak.

A sorok végét jelző speciális karakter a \n (soremelés, ASCII CODE 10).

Tartalmi követelmények

Az alábbi alfejezet a küldendő állományok általános, tartalmi követelményeit írja le.

Minden adatállomány esetében (.dat fájlok) az állománynak egy header sorral kell kezdődnie, melyben az állomány oszlopainak neve található. Ezek az elnevezések jelen dokumentum üzleti tartalmat specifikáló xls mellékletében találhatóak. Úgynevezett tailer rekordot az állományoknak nem kell tartalmaznia. Az állományok naplófájljai (.log fájlok) esetében erre nincs szükség.

### Rendezés

Az adatállományok esetében elvárás egy természetes rendezés az adattárház oldali indexelések megkönnyítése érdekében. Delta állományok esetében ez a változás időpontja szerinti növekvő rendezést jelenti. Tranzakció jellegű adatok leválogatása esetén (például könyvelési adatok) ez a rekord létrejöttének időpontja a rendezés szempontja. Egyéb, speciális adatkörök esetében az adott adatkör fejezetében leírtak szerint kell rendezni.

### Deperszonalizáció

A személyes adatok deperszonalizálása az adattárházba történő betöltés előtt jogszabályi elvárás (257/2016. (VIII. 31.) Korm. rendelet az önkormányzati ASP rendszerről), így a deperszonalizáció az input állományok adattárháznak történő átadása előtt meg kell, hogy történjen függetlenül a küldő szakrendszertől és intézménytől.

A deperszonalizáció megvalósítása központosítottan történik. E megoldás lényege, hogy ne minden interfészen csatlakozó önkormányzat és annak szállítója végezze a fejlesztést külön-külön, hanem egy központi megoldás valósuljon meg, melyet az önkormányzatok és a szállítók lokálisan, a saját környezetükben tudnak majd implementálni és használni. Az egységes megoldás mellett szól továbbá az, hogy a közös fejlesztés, tesztelés, dokumentálás csökkenti a ráfordításokat, valamint az egyetemleges verziókezelés miatt egyazon módon fog folyni a deperszonalizáció minden önkormányzatnál.

A deperszonalizációs alkalmazás, mind önkormányzati, mind szakrendszeri szinten csak egyszer kerül lefejlesztésre és központilag kerül menedzselésre, ezáltal csökkenti a fejlesztés/tesztelés ráfordítását az Önkormányzati oldalon, és redukálja a hibás, vagy nem teljes körű deperszonalizáció esélyét.

Reperszonalizáció alatt a jelenlegi technológiai lehetőségek alapján a tárolt adat és hash-elt adat párok kiolvasását érjük. Erre a lokálisan elvégzendő tesztek, valamint a majdani éles üzemi használat során az adattárházban készített elemzések lokális, önkormányzati szinten történő hasznosítása miatt van szükség.

A reperszonalizáció akkor működik megfelelően, ha a felhasználó önkormányzat minden adatkörére (pl.: GAZD, ADÓ, IPARKER stb.) képes elvégezni a hash-elt érték visszakeresését. Mivel az adattárház támogatni fogja az adatkörökön átívelő lekérdezéseket, ezért a reperszonalizációs adatbázist önkormányzatonként célszerű kialakítani.

Minden forrásrendszer minden deperszonalizálandó adata esetében elvárás, hogy az üres (*null*) értéket ne deperszonalizálja. Ennek oka a deperszonalizációhoz használandó salt érték biztonságos kezelése, valamint az, hogy az adattárházban az üres értékekre szűrés könnyen elvégezhető legyen. Ez a követelmény a központi megoldásban lesz kezelve.

A deperszonalizálandó adatok a 6.1-es üzleti specifikációt tartalmazó mellékletben kerülnek meghatározásra. Ezen adatok esetében kiemelten fontos a mellékletben definiált formai elvárások betartása, ellenkező esetben a deperszonalizáció nem fut le, az algoritmus hibaüzenettel tér vissza és a csomag nem küldhető az adattárház felé.

### Ellenőrzések

Ahogyan az az 1. ábrán szerepelt, az átadandó csomagokkal kapcsolatban több ellenőrzés elvégzése is szükséges. Ezek egy része a feladó oldalon, másik része a fogadó oldalon implementálandó.

Az átadandó állományok előállítása után közvetlenül el kell végezni bizonyos számszaki ellenőrzéseket, melyek a leválogatott adatok tartalmi helyességét hivatottak biztosítani. Ezek az ellenőrzések adatkörönként eltérnek, így ezek bővebben a rendszerspecifikus dokumentumokban kerülnek tárgyalásra.

Mind feladói, mind fogadói oldalon a tömörítési eljárás előtt és után is szükséges egy formátumellenőrzés annak érdekében, hogy az állományok a specifikációnak megfelelő formában érkezzenek meg az adattárházba, ezzel elősegítve a feldolgozás zökkenőmentes futását.

Az adattárház bizonyos funkciói miatt betöltéskor ellenőrizzük a dátumokat, és a (vélhetően adathibából fakadó) 1400.01.01 előtti dátumokat 1400.01.01-re cseréljük. Természetesen a kialakítandó adatminőségellenőrzési riportokon megjelennek majd ezek az 1400.01.01 előtti dátumok is. Minden csomag elküldésével együtt a csomaghoz tartozó metaadatok is feladásra kerülnek (részletesebben lásd a 3.2.4 fejezetben). Ezen metaadatok alapján még a csomag kicsomagolása előtt történik két ellenőrzés. Egyrészt az adattárház ellenőrzi, hogy kitől és mi érkezett, megvizsgálja, hogy az adott feladó küldhetett-e adott tartalmú állományt (pl.: adott feladó rendszer jogosult-e adott PIR-számú adatok küldésére, vagy adott adatkört valóban attól a szakrendszertől várja-e az adattárház). Ezek után lefut egy checksum (MD5) ellenőrzés, ami a feladott fájl sértetlenségét hivatott megvizsgálni, gyakorlatilag a küldés sikerességét ellenőrzi.

A leválogatások, állományelőállítások után a feladói oldalon minden esetben kell készülnie egy naplófájlnak az adott átadandó csomaghoz (a naplófájl tartalmát részletesebben lásd a 3.2.4 fejezetben). Ennek a naplófájlnak is ellenőrzési funkcionalitásokat kell ellátnia. Fogadó oldalon ez alapján kerül ellenőrzésre, hogy a csomagban megfelelő számú állomány érkezett-e, azok időszaka megfelel-e az aktuálisan várt időszaknak (nem korábbi és nem későbbi), valamint az, hogy az egyes állományok a megfelelő számú sort tartalmazzák-e, nem sérült-e esetleg meg valamelyik fájl az előállításkor vagy a továbbításkor. Többszörös ellenőrző funkciója miatt ez az egyik legfontosabb ellenőrző lépés, amit fogadó oldalon el lehet végezni, ezért a naplófájl megléte és megfelelő tartalma kiemelt fontosságú.

A forrás oldali számszaki ellenőrzésekhez hasonlóan a data lake-ben is történnek adattartalmi ellenőrzések. Ezek az ellenőrzések a forrásoldali ellenőrzésekhez hasonlóan adatkörspecifikus szabályrendszeren alapulnak, így ezek is bővebben a rendszerspecifikus dokumentumokban kerülnek tárgyalásra.

### Naplófájl és kísérő metaadatok

A forrásrendszerek által feladandó csomag állományaihoz tartozik egy naplófájl. Ez a leválogatást naplózza, azaz azt, hogy az adott csomagban milyen fájlok, ezeknek milyen tartalma van. Ebben az alábbiaknak kell szerepelnie.

|  |  |  |
| --- | --- | --- |
| **Naplófájl oszlop** | **Leírás** | **Naplófájl mintasor** |
| fajlnev | Adott fájl neve | gazd\_urlaphavi\_330684\_asp\_20180101\_20180228 |
| idoszak\_tol | Adott fájl adattartalma időszakának kezdete | 2018-01-01 |
| idoszak\_ig | Adott fájl adattartalma időszakának vége | 2018-02-28 |
| sor\_db | Adott fájl sorainak száma (fejléc nélkül) | 120 |

5. táblázat – Naplófájl tartalma

Minden, a csomagban lévő fájlhoz kell tartoznia egy rekordnak ebben a naplóban.

Ezen felül a csomagok küldésének meghívásakor szükséges bizonyos metaadatok elküldése a csomag mellett. Ezek az alábbiak.

| **Metaadat** | **Leírás** | **Metaadat mintasor** |
| --- | --- | --- |
| fajlNev | Adott fájl neve | gazd\_urlaphavi\_330684\_asp\_20180101\_20180228.tar.gz |
| pirCode | Csomaghoz tartozó PIR szám | 330684 |
| adatkorAzon | Adatkör azonosító (lásd 2. táblázat – Adatkör kódok) | urlaphavi |
| fajlVerzio | Fájl verziója | 1.0 |
| deperszonVerzio | Deperszonalizáció verziója | 1.0 |
| feladasAzonosito | Feladás azonosítója (UUID[[1]](#footnote-1)) | c8b78d18-534c-4502-b5c3-7ed37d426acf |
| feladasIdopontja | Feladás időpontja | 2018-03-31T15:00:00 |
| levalogatasIdopontja | Leválogatás időpontja | 2018-03-31T15:00:00 |
| idoszakTol | Adattartalom időszakának kezdete | 2018-01-01 |
| idoszakIg | Adattartalom időszakának vége | 2018-02-28 |
| gyakorisag | Adatküldés gyakorisága (napi/heti/havi/eves/os) | havi |
| szakrendszer | Szakrendszer azonosítója (lásd 1. táblázat – Szakrendszer kódok) | gazd |
| checksum | checksum (MD5) | ba411cafee2f0f702572369da0b765e1 |
| felado | Rendszer szállítója (lásd 3. táblázat – Feladó kódok) | asp |
| ures | Fájl nélküli feladás jele (lásd lentebb) | 0 |
| forras | ASP esetében a forrás tenant neve,interfészes önkormányzat esetében "felho" vagy "lokalis" annak függvényében, hogy felhő alapú szolgáltatást használ vagy lokálisan telepített (úgynevezett on premise) megoldást. | lippo |

6. táblázat – Metaadatok

### Checksum képzése

Az adatsorok folytonosságának ellenőrizhetőségét biztosítandó a metaadatok között szereplő checksum mezőt úgynevezett hash-láncként kell előállítani. Ez azt jelenti, hogy adott értéket az előző sikeres küldés checksum értékéből és az aktuálisan küldendő tar.gz formátumú csomagból kell MD5 algoritmussal előállítani. Feladónként, csomag PIR számonként, szakrendszerenként, adatkörönként és gyakoriság (ős és az azt követő napi delták ilyen szempontból egy láncot képeznek) szerint bontva kell a hashláncot előállítani. (Például az asp feladó 330684 PIR számra hash láncot épít az egymást követő sikeres heti űrlapokból. Egy másik láncot épít a sikeres havi űrlapokból, és egy harmadikat a kötvál adatokból, ahol a lánc egy ősfeltöltéssel indult, majd a napi sikeres delta állományokkal folytatódik.) Az első adatállomány előző checksum értéke: d41d8cd98f00b204e9800998ecf8427e (MD5("") vagy MD5(NULL)). Amennyiben nem áll rendelkezésre feladó oldalon az előző sikeres csomag checksum értéke, lehetőség van újraindítani a láncot. Ekkor ugyanúgy kell eljárni, mint a lánc első eleménél, vagyis előző checksum értékként az alábbit kell használni: d41d8cd98f00b204e9800998ecf8427e (MD5("") vagy MD5(NULL)).

Fontos, hogy a sikeresen feldolgozott csomagokból kell láncot fűzni. Üres feladások a láncba nem számítanak bele, hiszen ott checksum-ot sem várunk.

A feladási láncokat SZAKRENDSZER, ADATKOR, PIR, FELADO és GYAKORISAG (ős és az azt követő napi delták egy láncba tartoznak) tulajdonságok alapján képezhetünk. Tehát azok a csomagok, ahol ezek a paraméterek megegyeznek, ott kell feladási láncot létrehozni. Egy új lánc indításánál felhasználunk egy „üres” checksum-ot. Ez a „d41d8cd98f00b204e9800998ecf8427e” (null állomány MD5 hash kódja).

/tmp>md5sum </dev/null

d41d8cd98f00b204e9800998ecf8427e -

Ezt az értéket ki kell íratni egy temporális állományba, majd ebbe az állományba bele kell íratni a csomag állomány (tar.gz állomány) tartalmát.

/tmp/chktmp>echo -n $(md5sum </dev/null | cut -d " " -f1) >checksumtmp.txt

/tmp/chktmp>cat gazd\_kotval\_735694\_saldo\_20180101\_20180210.tar.gz >> checksumtmp.txt

/tmp/chktmp>md5sum checksumtmp.txt

830fe2a437096600862a3ea8bc56ec81 checksumtmp.txt

A példa alapján a feladási lánc első csomagjának checksum-ja „830fe2a437096600862a3ea8bc56ec81” lesz.

Ezután a temporális fájlt törölni lehet, a generált checksum-ot pedig el kell tárolni, hiszen a következő csomag checksum előállításához szükség lesz rá.

A következő csomag checksum-ja az alábbi módon készül:

/tmp/chktmp>echo -n "830fe2a437096600862a3ea8bc56ec81" >checksumtmp.txt

/tmp/chktmp>cat gazd\_kotval\_735694\_saldo\_20180101\_20180210\_1.tar.gz >>checksumtmp.txt

/tmp/chktmp>md5sum checksumtmp.txt

4c1d7c284ea7f1710bad938804bf19c9 checksumtmp.txt

/tmp/chktmp>

Tehát az előző csomag checksum-ját kiírjuk egy temporális állományba. Fontos, hogy a kiíratott kód után ne legyen semmilyen sorvégjel. Ezután az aktuális tar.gz állományt hozzámásoljuk az első lépésben ismertetett módszerrel, majd kiszámítjuk a hash kódot.

Ennek a csomagnak a checksumja: „4c1d7c284ea7f1710bad938804bf19c9”

Checksum számítás Windowsban Powershell-ben elvégezve:

PS C:\Users\chktmp> Out-File -FilePath checksumtmp.tmp -Encoding ascii -InputObject d41d8cd98f00b204e9800998ecf8427e -NoNewline

PS C:\Users\chktmp> cmd /c copy /b .\checksumtmp.tmp + .\gazd\_kotval\_735694\_saldo\_20180101\_20180210.tar.gz .\checksumtmp.tmp

.\checksumtmp.tmp

.\gazd\_kotval\_735694\_saldo\_20180101\_20180210.tar.gz

 1 file(s) copied.

PS C:\Users\chktmp> Write-Output (certutil.exe -hashfile .\checksumtmp.tmp MD5 | select -First 2 | select -Last 1).replace(' ','')

830fe2a437096600862a3ea8bc56ec81

Tehát a feladási lánc első elemének az MD5 hash kódja „830fe2a437096600862a3ea8bc56ec81” lesz a linux eljárással megegyezően. Itt is fontos, hogy a checksum kiíratása a –NoNewline opcióval kerüljön végrehajtásra.

A következő csomag checksum-ja az alábbi módon készül:

PS C:\Users\chktmp> Out-File -FilePath checksumtmp.tmp -Encoding ascii -InputObject 830fe2a437096600862a3ea8bc56ec81 -NoNewline

PS C:\Users\chktmp> cmd /c copy /b .\checksumtmp.tmp + .\gazd\_kotval\_735694\_saldo\_20180101\_20180210\_1.tar.gz .\checksumtmp.tmp

.\checksumtmp.tmp

.\gazd\_kotval\_735694\_saldo\_20180101\_20180210\_1.tar.gz

 1 file(s) copied.

PS C:\Users\chktmp> Write-Output (certutil.exe -hashfile .\checksumtmp.tmp MD5 | select -First 2 | select -Last 1).replace(' ','')

4c1d7c284ea7f1710bad938804bf19c9

A kapott checksum ebben az esetben is „4c1d7c284ea7f1710bad938804bf19c9”.

A fenti eljárások csak egy-egy mintaként szolgálnak, természetesen más módszerekkel is elvégezhetők a lépések. Az MD5 módszer biztosítja, hogy jól elvégezve a műveleteket azonos eredményt kapunk minden esetben.

### Üres csomagok feladása

Bizonyos esetekben előfordulhat, hogy adott időszaki leválogatás nem tartalmaz rekordokat. A specifikáció alapján ilyenkor csak header sorokat tartalmazó állományok állnának elő, melyek mellett egy csupa nulla sorszámot leíró napló kerülne a csomagban. Ezek feladása és feldolgozása az adatkörök és ASP-s tenantok számosság miatt napi akár több száz felesleges adatátadást és feldolgozást generálna, ami kerülendő. Ezért ilyen esetében a fájlküldés fájl nélküli, de kitöltött metaadat tartalommal történő meghívása a teendő úgy, hogy a metaadatok között szereplő „*ures*” jel 1-es értéket kap, és a checksum érték is üresen marad. Így az adattárház értesül róla, hogy a leválogatás sikeresen megtörtént, de adatot nem adott vissza, ugyanakkor az üres állomány feladásának hiánya miatt felesleges feldolgozási folyamatok nem indulnak el.

Ilyen feladás esetében a következő nem üres feladás időszakának a legutóbbi időszak plusz egy napnak kell lennie, megőrizve ezzel az időszakok folytonosságát. Például, ha egy napi interfésznél 09.02-án van adat, de 09.03-án és 09.04-én nincs, akkor két fájl nélküli feladás történik ezeken a napokon. Ha 09.05-én már van adat, akkor az a fájl 09.05-09.05 időszaki megjelöléssel kell, hogy érkezzen. A hash lánc megelőző eleme ebben az esetben a 09.02-ai, azaz a legutolsó nem üres csomag kell, hogy legyen.

Tömörítés

A forrásrendszerek által feladandó állományok tömörítésére tar.gz formátumú archívumot előállító tömörítési eljárás alkalmazandó. Ez jelent először egy tar (Tape Archive) archiválási metódust, ami az állományokat egy fájllá fűzi össze. Majd ezt követően egy gzip tömörítési eljárás futtatását, ami tömöríti az első lépés után előállt .tar fájlt.

A tömörített csomag nem tartalmazhat könyvtárat, kizárólag a megfelelő számú .dat kiterjesztésű fájlokat, azok mellett pedig egy darab .log kiterjesztésű naplófájlt.

A csomagokban lévő egyes fájloknak 0440 oktális kódolású jogosultságokkal kell rendelkezniük, azaz a tulajdonosnak és a csoportnak kell rendelkeznie olvasási jogosultsággal.

Titkosítás

Az interfészhez meghatározott biztonságos kommunikációs csatorna (EIP), valamint a szenzitív adatok deperszonalizálása miatt külön titkosítási eljárás alkalmazása nem elvárás. Az EIP és a deperszonalizáció megfelelő annak biztosítására, hogy szenzitív adatok ne kerülhessenek illetéktelenek kezébe.

Vírusellenőrzés

A vírusellenőrzést a NISZ által üzemeltetett, a “Kormányzati hitelesítésszolgáltatás kiterjesztése” c. KÖFOP projekt keretében megvalósult víruskereső alkalmazás végzi. Az alkalmazás részére a tömörített állományok továbbításra kerülnek, majd a külső víruskereső szoftverek felhasználásával elvégzi az ellenőrzést a beérkező állományokon. Az alkalmazás és az ASP 2.0 adattárház közötti kommunikáció REST szabványnak megfelelő formában történik, az üzenetek pontos specifikációja megtalálható a víruskereső alkalmazás dokumentációjában.

A folyamat során az adattárház oldal kezdeményezi a vírusdetektálási kérést az alkalmazás felé. Sikeres befogadás esetén a küldött állomány bekerül a végrehajtási sorba alkalmazás oldalon. Az adattárház üzenetet kap egy ID-val, melynek segítségével lekérdezheti a végrehajtás állapotát a víruskereső alkalmazás oldalán.

Küldés

A szállítói oldal a smoke teszt időtartama alatt SOAP UI request-eken keresztül kommunikál az adattárházzal. A smoke teszt lezárulta után a SOAP UI által használt WSDL alapján szállítói oldalon szükséges egy, a későbbiek során használandó EIP interfész fejlesztése.

A betöltési prototípus ideje alatt és később, az üzemszerű működés során az adattárház oldalán az állományokat egy SOAP[[2]](#footnote-2) protokoll szerint működő szolgáltatás fogadja. A SOAP protokoll miatt a szolgáltatással történő kommunikáció XML dokumentumok küldéséből és fogadásából fog állni. A szolgáltatás által elfogadott és küldött üzenetek (XML dokumentumok) szerkezetéről pontos leírást az DWHFileService.wsdl melléklet biztosít.

A szolgáltatással történő kommunikációt minden esetben a hívó fél kezdeményezi, az állomány küldőjének ezt a kommunikációt elvégző interfészt le kell fejlesztenie. Az adattárház szolgáltatás a következőkben leírt műveleteket támogatja, a kifejlesztett interfésznek képesnek kell lennie az összes meghívására, és a kapott válasz értelmezésére.

A leválogatásoknak és a küldéseknek értelem szerűen minden csomag esetében a csomag időszaka után kell megtörténnie. Például egy napi leválogatásnak a nap vége, azaz éjfél után kell megtörténnie, ellenkező esetben nem biztosítható a küldött adatok teljessége, konzisztenciája.

### UploadDocument

A kérés tartalma a feltöltött állomány metaadatai, és maga az állomány binárisan. A kérés a következő metaadatokat tartalmazza:

* fajlNev,
* pirCode,
* adatkorAzon,
* fajlVerzio,
* deperszonVerzio
* feladasAzonosito,
* feladasIdopontja,
* levalogatasIdopontja,
* idoszakTol,
* idoszakIg,
* gyakorisag,
* feladoRendszer,
* checksum,
* szallito,
* ures,
* forras

A kérésre érkező válasz tartalmaz egy documentId nevű azonosítót, mely később (a feldolgozási folyamat státuszának lekérdezésekor) használandó, valamint egy varhatoAllapotValtozasSec nevű értéket, mely másodpercben adja meg, hogy mennyi idő múlva érdemes lekérdezni a feldolgozás állapotát annak érdekében, hogy már egy új állapotban találjuk azt.

### GetDocumentStatus

A kérés tartalmazza a documentId-t, melyet az UploadDocument kérésre kapott válaszban küldött a beküldő részére az adattárház. A kérésre a válasz a dokumentum feldolgozás státusza.

### GetDocumentStatusByFilename

A kérés során a documentId ismerete nélkül is le tudjuk kérni a csomag státuszát a fajlNev megadásával. A kérésre a válasz a feldolgozás státusza.

### GetDocumentErrorLog

Amennyiben a GetDocumentStatus által visszaadott státuszinformáció arra enged következtetni, hogy a feldolgozás során hiba történt, úgy a GetDocumentErrorLog kérés és a documentId segítségével lekérdezhetők a hibával kapcsolatos információk. A kérésre adott válasz tartalma a hibákkal kapcsolatos log állomány neve (fileName) és annak tartalma (fileContent).

Hibakezelés

Az ASP 2.0 adattárház oldalán a feldolgozás során hibakezelés nem történik. A folyamat ellenőrzéseket tartalmazó lépéseiben amennyiben az ellenőrzések sikertelenek, vagy a víruskeresés kártékony összetevőt talál, minden esetben a folyamat valamely szállítói oldalon található lépésétől kell újrakezdeni a folyamatot.

Az ellenőrzéseket tartalmazó lépések és az indokolt esetben szállítói oldalon való újrakezdés pontjai a következők:

|  |  |
| --- | --- |
| **Adattárház oldali lépés** | **Újrakezdési pont szállítói oldalon** |
| Metaadat alapú ellenőrzés | Tömörítés |
| Vírus ellenőrzés | Kiöntés |
| Fájltípus ellenőrzés | Tömörítés |
| Formátum ellenőrzés | Kiöntés |
| Naplófájl alapú ellenőrzés | Naplófájl előállítása |
| Data lake ellenőrzés | Szakrendszeri leválogatás |

7. táblázat – Ellenőrzések és visszajelzések

Visszajelzés

A szakrendszeri oldalon, miután a tömörített állomány feltöltésre került és az UploadDocument kérésre kapott válaszból a szakrendszeri oldalon rendelkezésre áll a documentId, a 3.6.2 és 3.6.3 fejezetekben ismertetett kérések használatával van lehetőség a feldolgozással kapcsolatos visszajelzések kérésére az adattárház részéről. GetDocumentErrorLog tartalmát képező logfájlbejegyzés tartalmaz minden, az esetleges problémákkal kapcsolatos információt, amely szállítói oldalon a hibakeresést és a hiba megoldását segítheti, valamint magukat a hibás rekordokat is.

A beérkezett állományok feldolgozása után egy e-mail formájú visszajelzésre is sor kerül az előzőleg megadott kapcsolattartók részére. Ez az e-mail a feldolgozás eredményével kapcsolatos információkat tartalmazza. Sikeres feldolgozás esetén ennek tényét, sikertelen feldolgozás esetén egy hibaüzenetet is tartalmaz, a hiba jellegével kapcsolatos információkkal. Ez alapvetően egy tájékoztató e-mail, melynek tartalma üzleti szempontból megegyezik a metódushívások által kinyerhető eredménnyel (azaz a felmerült hibatípusokat tartalmazza).

# Tesztelés és próbaüzem

Jelen fejezet célja, hogy bemutassa az ASP2 adattárház alprojekt során elvégzendő teszteket, és a tesztelési feladatok bontását megadja.

A tesztelésben a következő résztvevőkkel szükséges számolni:

1. Kincstár – Az ASP2 és az ASP2 adattárház rendszerért felelős kormányzati szerv, aki osztja és elszámoltatja a feladatokat, döntéseket hoz, valamint konzultációs, koordinációs szerepet lát el.
2. Kincstár ITB – Kincstár, mint megrendelő IT Biztonsági osztálya, a rendszert érintő biztonsági kérdésekben dönt, valamint konzultációs pont.
3. T-Systems - ASP2 adattárházért és az adattárház integrációért felelős vállalkozó.
4. Szakrendszerek – Jelen dokumentum nem tesz különbséget az interfészen keresztül csatlakozó önkormányzatok és az ASP szakrendszereiből feladott állományok tesztelése között.
5. Önkormányzatok – A szakrendszerek és a szakrendszerekben tárolt adatok gazdái.
6. ASP Keret – de- és reperszonalizációs megoldás szállítója
	1. Csatlakozási (smoke) teszt

A csatlakozási teszt során a szakrendszerek csatlakozási potenciáját szükséges felmérni. A teszt során az interfészt használó egy előre generált állományt ad fel egy SoapUI projektben. A teszt sikerkritériuma, hogy a file megérkezzen a data lake integrációért felelős Edge szerverére. A folyamat lépéseit és a hozzá tartozó felelősségi köröket az alábbi táblázat szemlélteti.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feladat** | **Felelős****(R)** | **Elszámoltató (A)** | **Konzultáció****(C)** | **Informálandó (I)** |
| Smoke teszthez szükséges SoapUI projekt és ennek dokumentálása | T-Systems | Kincstár | Szakrendszerek,Kincstár ITB | Szakrendszerek,NISZ |
| SoapUI telepítése  | Szakrendszerek | Kincstár | T-Systems | Kincstár ITB |
| File feladás | Szakrendszerek | Kincstár | T-Systems | Kincstár ITB |
| Beérkező file-ok fogadása | T-Systems | Kincstár |  | Szakrendszerek |
| Teszt kiértékelése | T-Systems | Kincstár | NISZ | Szakrendszerek |

8. táblázat – Smoke teszt RACI mátrix

* 1. Betöltési prototípus tesztek

A betöltési prototípus tesztek célja, hogy a data lake töltési eljárást néhány szakrendszeren (GAZD, KTÖRZS, IPARKER), kisebb adatmennyiségen tesztelje az alábbi aspektusok szerint:

* a szakrendszeri leválogatók működése,
* az előállt állományok deperszonalizálása,
* a szakrendszerből automatikusan történő file-feladásokat és a válaszállományok lekérdezése,
* a beérkezett file-ok feldolgozása az adattárház oldalán,
* hibakezelések tesztelése (lásd 3.7 fejezet),
* ősfeltöltési képesség,
* delta töltési képesség,
* riportok előállítása
* riportok kiküldése az önkormányzatok számára,
* riportok reperszonalizációja,
* riportok adathelyességének ellenőrzése.

A felhasználói tesztelések elkezdése előtt minden szállítónak lehetősége van fejlesztői tesztek keretében tesztállományokat küldeni. Ezek elkülönítése érdekében alkalmazandó egy *test\_* prefix a korábbi fejezetekben definiált névkonvenciók előtt.

A felhasználói teszt során az interfészt használó egyszeri ősfeltöltési állományt ad fel, majd időszakonkénti változás (delta) állományt. Azon szakrendszereknél, ahol nem értelmezhető az ősfeltöltés és a deltaképzés (pl.: űrlap szerinti aggregátumok), ott az adott adatkör esetében a rendszerspecifikus dokumentációban definiált állományt szükséges feladni. Az állományok feldolgozása a data lake-ben történik meg. Az itt elkészült riportokat az önkormányzatok lokális (azaz a saját környezetükben elvégzett) reperszonalizáció után ellenőrzik.

A teszt sikerkritériuma az, hogy az interfész kommunikáció automatikusan működjön, és a szakrendszerek által szolgáltatott – szükség esetén deperszonalizált - adatok (egyszeri, folyamatos és delta töltések) helyes formátumban, helyes adattartalommal érkezzenek be az adattárházba.

A folyamat lépéseit, és a hozzá tartozó felelősségi köröket a lentebbi táblázat szemlélteti

| **Feladat** | **Felelős****(R)** | **Elszámoltató (A)** | **Konzultáció****(C)** | **Informálandó****(I)** |
| --- | --- | --- | --- | --- |
| Szakrendszeri leválogató tesztelése | Szakrendszerek | Kincstár | T-Systems,NISZ |  |
| Deperszonalizációs megoldás Szakrendszeri leválogatóba való illesztésének tesztelése | Szakrendszerek | Kincstár | ASP2 Keret | T-Systems |
| Automatikus file feladás Szakrendszeri leválogatóból, és a válasz állományok lekérdezése | Szakrendszerek | Kincstár | T-Systems,NISZ |  |
| Ősfeltöltési állományok feladása | Szakrendszerek | Kincstár | T-Systems,NISZ |  |
| Delta állományok feladása | Szakrendszerek | Kincstár | T-Systems,NISZ |  |
| Egyszeri állományok feladása | Szakrendszerek | Kincstár | T-Systems,NISZ |  |
| Beérkezett állományok feldolgozása | T-Systems | Kincstár | Szakrendszerek,Kincstár |  |
| Riportok előállítása, és eljuttatása az Önkormányzatok számára  | T-Systems | Kincstár | Szakrendszerek,Önkormányzatok |  |
| Elkészült riportok reperszonalizációja | Szakrendszerek | Kincstár | ASP2 Keret,Önkormányzatok | T-Systems |
| Riportok ellenőrzése | Önkormányzatok | Kincstár | T-Systems,Szakrendszerek |  |

9. táblázat – Betöltési prototípus teszt RACI mátrix

# Ütemezés

A betöltési prototípus elkészítésének ideje alatt a feladatok elvégzése tekintetében az alábbi ütemezés követendő.

| **Tevékenység** | **Leírás, megjegyzés** | **Kezdete** | **Vége** |
| --- | --- | --- | --- |
| **Csatlakozási teszt** | Küldő és fogadó szerverek közötti kommunikáció ellenőrzése | 2018.04.09 | 2018.04.16 |
| **Specifikáció** |  | 2018.03.19 | 2018.05.15 |
|  | SOAP UI projekt átadása |  | 2018.04.03 |
|  | Első verzió: GAZD aggregátumok, KTÖRZS, adatellenőrzési elvárások | 2018.02.15 | 2018.04.03 |
|  | Teljes kidolgozás a következőkre: GAZD prototípus elemi adatok és IPARKER prototípus elemi adatok, deperszonalizációs elvárások kidolgozása és átadása | 2018.04.02 | 2018.05.15 |
| **Fejlesztés** | Fejlesztések: önkormányzati és ASP-s forrásrendszerekben és az adattárházban | 2018.04.02 | 2018.06.15 |
|  | Lehetőség fejlesztői teszt állományok küldésére (*t*e*st\_* prefix-szel). | 2018.04.23 | 2018.06.15 |
| **Tesztelés** | Interfész csatlakoztatási tesztek, ide értve a delta képzés tesztelését is | 2018.06.15 | 2018.07.02 |
| **Próbaüzem** | 6 hetes próbaüzemi időszak a további hibák feltárása és javítása érdekében | 2018.07.02 | 2018.08.17 |
| **BM döntés** | ASP rendelet 14.§ (3) bekezdése szerinti felülvizsgálat | 2018.08.17 | 2018.09.30 |
|  | Kincstár próbaüzemi tesztelési jegyzőkönyvek előállítása | 2018.08.17 | 2018.08.30 |
|  | Jegyzőkönyvek felülvizsgálata, BM döntések meghozatala | 2018.08.30 | 2018.09.30 |
| **Teljes specifikáció**  |  |  | 2018.10.31 |
|  | Interfész specifikációk átadása a GAZD űrlapok kiegészítése, IPARKER adatokra |  | 2018.08.31 |
|  | Interfész specifikációk átadása a további adatkörökre: Teljes GAZD, ADO, IRAT, IVK, KERET, ÖNEGM |  | 2018.10.31 |

10. táblázat – Betöltési prototípus ütemezése

# Mellékletek

* 1. Üzleti specifikáció

A 6.1-es melléklet a dokumentumban többször meghivatkozott táblázat, mely az interfészek üzleti tartalmát specifikálja. A specifikáció későbbi fázisaiban a táblázat folyamatosan bővülni fog. Az xls három fület tartalmaz.

Az első fül az adattárház által várt csomagokat és azok jellemzőit tartalmazza. Ezen a fülön külső szállító esetén 1 sor 1 csomagnak felel meg, ASP esetében minden tenanthoz tartozik 1 sor, azaz 1 sor nagyjából 3200 csomagot jelöl (azokat a paramétereket, melyek mentén több csomag áll elő szögletes zárójel, azaz [] jelöli, pl.: [pir]). Így a „csomag fejlesztője” oszlopra szűrve minden szállító megkapja, hogy hány interfész csomagot vár tőle az adattárház, valamint mi ezen csomagok csomagszintű jellemzői.

A második fül minden csomaghoz tartalmazza az abban a csomagban várt állományokat (adatállományok és a naplófájl), valamint ezen állományok oszlop szintű üzleti tartalmát. Bizonyos adatkörök esetében (például GAZD űrlapok) előfordulhat, hogy valamely paraméter mentén több állomány is előállítandó ugyanabban a csomagban, ezeket a paramétereket (hasonlóan az előző fülhöz) szögletes zárójel, azaz [] jelöli. Például, ha egy önkormányzat az önállóan gazdálkodó intézményeinek adatait is be szeretné küldeni az adattárházba, akkor ezeket ugyanazon csomagon belül, de külön adatállományban teheti meg.

A harmadik fül az állományokon belül előforduló kódok értékkészleteit tartalmazza.

* 1. Egyszerűsített „E” űrlap koncepció

A 01-R-E űrlap egyszerűsített verziójának koncepcióját a csatolt 6.2-es melléklet tartalmazza.

* 1. Mintaállományok

### Gazdálkodási szakrendszer - Űrlap adatok adatköre mintaállomány

gazd\_urlaphavi\_727244\_asp\_20170101\_20171231.tar.gz

### Ipar- és kereskedelmi rendszer - Szálláshely adatok adatköre mintaállomány

iparker\_szallas\_732923\_asp\_20180401\_20180401.tar.gz

## Az állományok feltöltéséhez szükséges szolgáltatás definíciója

DWHFileService.wsdl

## „A” űrlap

A 01-R-A űrlap beküldendő verzióját a csatolt 6.5-ös melléklet tartalmazza.

1. universal unique identifier (vagy globally unique identifier): globális egyedi azonosító, szoftveralkalmazások által használt, az RFC 4122 version 4 szerint képzett álvéletlen szám, mely a feladás egyedi azonosítását hivatott biztosítani. [↑](#footnote-ref-1)
2. http://www.w3.org/TR/soap/ [↑](#footnote-ref-2)