



AMIS*PACS FlexServer G2

verze 2.25.04-REL, vydáno 2024-05-30

Programming guide

ICZ a.s.

CE 1023

AMIS*PACS FlexServer G2: Programming guide

verze 2.25.04-REL, vydáno 2024-05-30

Copyright © 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024 ICZ a.s.

Žádná část tohoto dokumentu nesmí být kopírována žádným způsobem bez písemného souhlasu majitelů autorských práv.

Některé názvy produktů a společností uvedené v tomto dokumentu mohou být ochranné známky příslušných vlastníků.

Výrobce:

ICZ a.s.

Na hřebenech II 1718/10

140 00 Praha 4 - Nusle

Česká republika

Obsah

1. Úvod	1
1.1. Účel	1
1.2. Konvence	1
2. Rozhraní SOAP	2
2.1. Služba testiface	2
2.1.1. WSDL	2
2.1.2. Datové typy	2
2.1.3. Metody	2
2.1.3.1. getVersion	2
2.1.3.2. sayHello	2
2.1.3.3. sayHelloToMe	3
2.1.3.4. throwExceptionWithText	3
2.1.4. Příklady	3
2.2. Služba geniface	3
2.2.1. WSDL	4
2.2.2. Datové typy	4
2.2.2.1. PatientIdentifier	4
2.2.2.2. PatientInfo	5
2.2.2.3. StudyInfo	5
2.2.2.4. SeriesInfo	6
2.2.2.5. ImageInfo	6
2.2.2.6. AttachedFileInfo	6
2.2.2.7. EGenIface	7
2.2.3. Metody	7
2.2.3.1. getVersion	7
2.2.3.2. findPatients	7
2.2.3.3. findStudies	8
2.2.3.4. getPatientInfo	8
2.2.3.5. getStudyInfo	9
2.2.3.6. getSeriesInfo	9
2.2.3.7. getImageInfo	9
2.2.3.8. getImageJpeg	9
2.2.3.9. prefetchStudyToViewer	9
2.2.3.10. Zastaralé	9
2.2.4. Další vlastnosti	10
2.2.4.1. Autentizace	10
2.2.4.2. Kontrola IP adresy	10
2.2.5. Příklady	10
2.2.5.1. Pomocná knihovna	10
2.2.5.2. Chybějící autentizační údaje	11
2.2.5.3. Špatné heslo	11
2.2.5.4. Úspěšná autentizace	11
2.2.5.5. Zachycení výjimky EGenIface	12
2.2.5.6. Průchod hierarchickou strukturou	12
2.3. Služba docarchiface	14
2.3.1. WSDL	14
2.3.2. Datové typy	14
2.3.2.1. PatientIdentifier	14
2.3.2.2. PatientInfo	15
2.3.2.3. Diagnosis	16
2.3.2.4. CaseInfo	16
2.3.2.5. EventInfo	16

2.3.2.6. AttachedFileInfo	17
2.3.2.7. AttachedFileVersionInfo	18
2.3.2.8. DesaRDProcessChangesInfo	19
2.3.2.9. DesaRDProcessStateInfo	20
2.3.2.10. EDocArchIface	20
2.3.3. Metody	20
2.3.3.1. getVersion	20
2.3.3.2. putAttachedFileVersionData	20
2.3.3.3. getAttachedFileVersionData	21
2.3.3.4. getAttachedFileMaxVersionData	21
2.3.3.5. getAttachedFileVersionInfo	21
2.3.3.6. getAttachedFileVersionStatus	21
2.3.3.7. getAttachedFileInfo	21
2.3.3.8. getCaseInfoOfAF	22
2.3.3.9. getEventInfoOfAF	22
2.3.3.10. getPatientInfo	22
2.3.3.11. findAttachedFiles	22
2.3.3.12. findAttachedFilesVersions	24
2.3.3.13. getRDProcessChanges	25
2.3.3.14. getRDProcessState	25
2.3.3.15. savePatient	26
2.3.3.16. saveAttachedFileInfo	26
2.3.3.17. updateExternalEvent	26
2.3.3.18. deprecateAttachedFile	26
2.3.4. Další vlastnosti	27
2.3.5. Příklady	27
2.3.5.1. Uložení AttachedFileVersion	27
2.4. Služba docrepoiface	29
2.4.1. ITI-41, RAD-68 DocumentRepository_ProvideAndRegisterDocumentSet-b	29
2.5. Služba docregiface	29
2.5.1. ITI-57 documentRegistryUpdateDocumentSet	29
3. Rozhraní STA/LTA REST API	30
3.1. Služba sendChangesFromSTA	30
3.2. Služba acceptChangesOnLTA	30
3.3. URL	31
4. Rozhraní shred/manage documents REST API	32
4.1. disposal REST API	32
4.1.1. Execute final disposal operation (shred or tag with final state)	32
4.1.2. Get package metadata	32
4.1.3. Get package metadata and components	33
4.1.4. Get codelist and classification	33
4.1.5. Get user credentials	33
4.1.6. Put document into archive system	33
4.1.7. Run disposal process action.	34
4.1.8. Create disposal process instance	34
4.1.9. Get disposal process detail	34
4.1.10. Get list of disposal processes	34
4.1.11. Get disposal process action progress	35
4.1.12. Get current state of disposal process	35
4.1.13. Disposal process candidate	35
4.1.14. Register packages into disposal process	35
4.1.15. Remove packages from disposal process	36
4.1.16. Disposal process registered packages	36
4.1.17. Change parameters of packages in disposal process	36

4.2. manage document REST API	37
4.2.1. Get document content	37
4.2.2. Get document metadata	37
4.2.3. Save document with metadata	37
4.2.4. Deprecate document	37
4.2.5. Save NSESSS document	38
4.2.6. Get NSESSS document	38
4.2.7. Save ISSD document	38
4.2.8. Get document STATUS	38
4.2.9. Get document metadata by producer and document ident	39
4.2.10. Get document metadata collection by document id and producer collection	39
4.2.11. Deprecate document by producer and document ident	39
4.2.12. Deprecate documents by collection of document id and producer	39
4.2.13. Get document content by producer and document ident	40
4.2.14. Get document identifiers by document producer and patient namespace and identifier	40
A. Příloha: Příklad SOAP klienta v PHP	41
A.1. Instalace a konfigurace (CentOS Linux 5.x)	41
A.2. Vytvoření klienta	41
A.2.1. WS-Security	41
A.2.2. Příklady	42
A.2.2.1. Úspěšná autentizace	42
A.2.2.2. Odchycení a zpracování výjimky	43
A.2.2.3. Průchod hierarchickou strukturou	43

Kapitola 1. Úvod

1.1. Účel

Tento dokument popisuje programové rozhraní (API) produktu *AMIS*PACS FlexServer G2*.

1.2. Konvence

Produkt *AMIS*PACS FlexServer G2* bude ve zbytku textu označován zkráceně jako APFS.

Kapitola 2. Rozhraní SOAP

Programové rozhraní APFS je založeno na dominantním prostředku pro integraci systémů - protokolu *SOAP*. Podpora SOAPu je dostupná pro všechny běžné programovací jazyky, které se využívají pro tvorbu nebo integraci podnikových aplikací (například Java, C#, C++, C, Perl, PHP, Python). Popis API bude ilustrován příklady v jazyce Python. Příklady pro některé další jazyky jsou pak uvedeny v přílohách.

Jako transportní protokoly pro SOAP jsou použity HTTP a HTTPS.

Rozhraní obsahuje tři služby;

- *testiface* je testovací API
- *geniface* je obecné API (orientované především na PACS aplikace)
- *docarchiface* je API dokumentového archivu

Každá služba pak poskytuje svoji vlastní sadu metod.

2.1. Služba testiface

Služba testiface představuje jednoduché testovací rozhraní.

2.1.1. WSDL

WSDL této služby je dostupné buď z běžícího APFS na URL:

```
http[s]://IP_adresa:TCP_port/testiface?wsdl
```

kde:

- *IP_adresa* je IP adresa nebo jméno počítače, na kterém běží APFS
- *TCP_port* je TCP port, na kterém APFS vystavuje SOAP rozhraní; obvykle 9090 pro HTTP a 9091 pro HTTPS

Příklad:

```
http://flex1:9090/testiface?wsdl
```

WSDL je k dispozici i offline. Je uloženo v souboru `conf/wsTestIface.wsdl` v instalačním adresáři APFS.

2.1.2. Datové typy

Služba nedefinuje vlastní datové typy kromě výjimky `ETestIface`. Ta obsahuje jediný atribut - řetězec text.

2.1.3. Metody

2.1.3.1. getVersion

Metoda nemá žádné parametry a vrací řetězec, který představuje verzi APFS. Metoda nevyhazuje výjimku.

2.1.3.2. sayHello

Metoda nemá žádné parametry a vrací řetězec "Hello!". Metoda nevyhazuje výjimku.

2.1.3.3. sayHelloToMe

Metoda má jediný parametr, řetězec *myName* a vrací řetězec obsahující "Hello ", zadaný parametr a "!". Metoda nevyhazuje výjimku.

2.1.3.4. throwExceptionWithText

Metoda má jediný parametr, řetězec *text* a vždy vyhodí výjimku `ETestIface`. V jejím atributu `text` je pak řetězec zadaný do metody jako parametr.

2.1.4. Příklady

Následuje příklad klientské aplikace, která postupně provolá všechny metody. Jde o skript v jazyce Python s využitím knihovny `Suds`:

```
import logging
from suds import WebFault
from suds.client import Client

logging.basicConfig(level=logging.FATAL)
client = Client('http://192.168.242.38:9090/testiface?wsdl')

print "\n--- getVersion(): ---"
print client.service.getVersion()

print "\n--- sayHello(): ---"
print client.service.sayHello()

print "\n--- sayHelloToMe(): ---"
print client.service.sayHelloToMe(myName="world")

print "\n--- throwExceptionWithText(): ---"
try:
    client.service.throwExceptionWithText(text="Some catastrophic message")
except WebFault, e:
    if (hasattr(e.fault, 'detail') and hasattr(e.fault.detail, 'ETestIface')):
        print "ETestIface caught, text: %s" % e.fault.detail.ETestIface.text
    else:
        print e
```

Výstup z této aplikace vypadá takto:

```
--- getVersion(): ---
1.02.03-REL

--- sayHello(): ---
Hello!

--- sayHelloToMe(): ---
Hello world!

--- throwExceptionWithText(): ---
ETestIface caught, text: Some catastrophic message
```

2.2. Služba geniface

Tato služba zpřístupňuje obecné rozhraní pro práci s APFS jako PACS archivem.

2.2.1. WSDL

WSDL této služby je dostupné buď z běžícího APFS na URL:

```
http[s]://IP_adresa:TCP_port/geniface?wsdl
```

kde:

- *IP_adresa* je IP adresa nebo jméno počítače, na kterém běží APFS
- *TCP_port* je TCP port, na kterém APFS vystavuje SOAP rozhraní; obvykle 9090 pro HTTP a 9091 pro HTTPS

Příklad:

```
http://flex1:9090/geniface?wsdl
```

WSDL je k dispozici i offline. Je uloženo v souboru `conf/wsGenIface.wsdl` v instalačním adresáři APFS.

2.2.2. Datové typy

Služba definuje několik složených datových typů. Formální definice typů je součástí WSDL, tady je popíšeme méně formálním způsobem s důrazem jejich sémantiku.

2.2.2.1. PatientIdentifier

APFS jednoznačně určuje všechny entity, a to i pacienta, unikátní identifikátorem UID. U některých entit, zejména DICOMových (studie, série, instance) toto UID vzniká obvykle na přístrojích a APFS je přebírá. U některých entit (například pacient, modalita, ...) žádný vnější zdroj UID není a tak si je APFS vytváří sám. Entita pacient je ale výjimečná tím, že má nebo může mít několik dalších identifikátorů z vnějších zdrojů (například rodné číslo, Social Security number, interní identifikátor přidělený klinickým systémem apod.) s různou mírou jednoznačnosti a trvanlivosti. APFS tyto identifikátory uchovává a umožňuje podle nich vyhledávat.

Vnější zdroj patientského identifikátoru nazveme *jmenný prostor* (*namespace*). Jeden jmenný prostor a identifikátor v něm pak představuje datový typ `PatientIdentifier`:

jméno	typ	význam
namespace	řetězec **	jmenný prostor identifikátoru
id	řetězec	vlastní identifikátor

**) povinné s podmínkou. pacient musí mít identifikátor, který je v systému nastaven jako default. viz dále

Pacient může mít více identifikátorů `PatientIdentifier`. Proto je položka `identifiers` v typu `PatientInfo` vícečetná.

Pacientské identifikátory mají tyto vlastnosti:

- Vždy platí, že pacient má v rámci jednoho namespace nejvýše jeden identifikátor.
- V rámci jednoho namespace může mít více pacientů tentýž identifikátor, neboli APFS připouští duplicity v identifikátorech. Některé komponenty APFS sice mohou v některých situacích a ze specifických důvodů znemožňovat vytvoření duplicity, přesto obecně platí, že duplicity jsou přípustné.
- Jeden namespace je označen jako default a je vždy specifický pro konkrétní instalaci APFS. V Česku to často bývají rodná čísla ("RC"), ale mohou to být i jiné jmenné prostory. Například pokud je APFS integrován s klinickým systémem, který

generuje vlastní identifikátory, pak se obvykle jako default namespace použije prostor primárního identifikátoru klinické aplikace. Default namespace se uplatňuje při ukládání nebo hledání, když k identifikátoru není uveden jeho namespace.

- nastavení default namespace v apfs.xml v sekci core:

```
patient.defaultnamespace=RC
```

Jak už jsme řekli v úvodu této kapitoly, APFS generuje vlastní řetězec UID, kterým jednoznačně identifikuje pacienta. Toto UID se strukturou `PatientIdentifier` nijak nespojuje. Je součástí typu `PatientInfo` jako položka `patientUid`.

2.2.2.2. PatientInfo

popisuje pacienta v APFS. Má následující položky:

jméno	typ	význam
patientUid	řetězec (64)	UID pacienta (nejde o DICOM atribut, ale o interní UID, které přiděluje APFS)
lastName	řetězec (32) *	příjmení
firstName	řetězec (32)	jméno
middleName	řetězec (32)	prostřední jméno
namePrefix	řetězec (16)	předpona jména
nameSuffix	řetězec (16)	přípona jména
identifiers	seznam objektů typu <code>PatientIdentifier</code>	identifikátory pacienta; více vizte u datového typu <code>PatientIdentifier</code>
birthDate	řetězec (8)	datum narození, neboli DICOM atribut (0010,0030); formát je RRRRMMDD
birthTime	řetězec (6)	čas narození, neboli DICOM atribut (0010,0032); formát je hhmmss
death	xsd:datetime	datum a čas úmrtí
sex	řetězec (1)	nejvýše jednoznakový řetězec vyjadřující pohlaví pacienta podle číselníku DICOM (M,F,O), neboli DICOM atribut (0010,0040)
studiesUids	seznam řetězců	UID všech DICOM studií, které pacientovi náleží
attachedFilesUids	seznam řetězců	tento seznam je vždy prázdný
ordersUids	seznam řetězců	UID všech žádanek, které pacientovi náleží (nejde o nějaké DICOM atributy, ale o interní UID, která přiděluje APFS)

*) povinná hodnota, musí být vyplněno

2.2.2.3. StudyInfo

nese informace o DICOM studii uložené v APFS. Položky tohoto datového typu jsou:

jméno	typ	význam
studyUid	řetězec (64)	Study Instance UID (UID samotné studie)
patientUid	řetězec (64)	UID nadřazeného pacienta (nejde o DICOM atribut, ale o interní UID, které přiděluje APFS)

jméno	typ	význam
accessionNumber	řetězec	číslo žádanky, neboli DICOM atribut (0008,0050)
referringPhysicianName	řetězec	přebírající lékař, neboli DICOM atribut (0008,0090)
studyID	řetězec	ID studie, neboli DICOM atribut (0020,0010)
studyDescription	řetězec	popis studie, neboli DICOM atribut (0008,1030)
seriesDate	řetězec	datum studie, neboli DICOM atribut (0008,0020); formát je RRRRMMDD
seriesTime	řetězec	čas studie, neboli DICOM atribut (0008,0030); formát je hhmmss
seriesUids	seznam řetězců	UID všech DICOM sérií, které patří do studie

2.2.2.4. SeriesInfo

popisuje DICOM série uložené v APFS. Obsahuje položky:

jméno	typ	význam
seriesUid	řetězec	Series Instance UID (UID samotné série)
studyUid	řetězec	Study Instance UID (UID nadřazené studie)
modality	řetězec	typ přístroje, neboli DICOM atribut (0008,0060)
bodyPartExamined	řetězec	vyšetřovaná část těla, neboli DICOM atribut (0018,0015)
seriesNumber	řetězec	číslo série, neboli DICOM atribut (0020,0011)
performingPhysicianName	řetězec	provádějící lékař, neboli DICOM atribut (0008,1050)
seriesDate	řetězec	datum série, neboli DICOM atribut (0008,0021); formát je RRRRMMDD
seriesTime	řetězec	čas série, neboli DICOM atribut (0008,0031); formát je hhmmss
imagesUids	seznam řetězců	UID všech DICOM objektů, které patří do série

2.2.2.5. ImageInfo

slouží jako popis DICOM objekty uložené v APFS. Má následující položky:

jméno	typ	význam
imageUid	řetězec	SOP Instance UID (UID samotného objektu)
seriesUid	řetězec	Series Instance UID (UID nadřazené série)
imageType	řetězec	Image Type, neboli DICOM atribut (0008,0008)
sopClassUid	řetězec	SOP Class UID, neboli DICOM třída objektu
sopClassName	řetězec	název DICOM třídy
numberOfFrames	řetězec	počet frejmů v objektu, tak jak je uveden v DICOM atributu (0028,0008); často bývá nastaven na null, i když objekt (obrázek) má přesně jeden frejm

2.2.2.6. AttachedFileInfo

Tento datový typ přestal mít v rozhraní "geniface" význam.

2.2.2.7. EGenIface

je výjimka, kterou mohou vyházovat všechny metody služby "geniface". Má jediný atribut:

jméno	typ	význam
text	řetězec	detaily toho, proč k výjimce došlo

2.2.3. Metody

2.2.3.1. getVersion

Tato metoda nemá parametry. Vrací verzi APFS. Může vyhodit výjimku `EGenIface`, například pokud příslušný klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění.

2.2.3.2. findPatients

Metoda hledá pacienty v databázi APFS. Vrací seznam řetězců UID všech nalezených pacientů. Detaily o jednotlivých pacientech je pak možné zjistit metodou `getPatientInfo()`.

Jediným parametrem metody je objekt typu `PatientInfo`. Pro hledání pacienta budou použity jednotlivé neprázdné položky takto (neprázdnou položkou se myslí řetězec o délce aspoň jeden znak):

jméno	použití
patientUid	přesné porovnání
lastName	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
firstName	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
middleName	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
namePrefix	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
nameSuffix	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
identifiers	porovnání s podporou zástupných znaků; pokud není uveden namespace, použije se místo něj default namespace
birthDate	porovnání na datum narození
birthTime	porovnání na čas narození
sex	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
studiesUids	tato položka je ignorována
ordersUids	tato položka je ignorována
attachedFilesUids	tato položka je ignorována

Podpora zástupných znaků (wildcards) znamená, že lze zadat zástupné znaky:

- * - tento znak představuje řetězec libovolné délky (i nulové)
- ? - tento znak představuje jediný znak

Položky `birthDate` a `birthTime` mohou být zadány obecněji, než je popsáno u datového typu `PatientInfo`. Mohou obsahovat i intervaly tvaru `RRRRMMDD-RRRRMMDD`, resp. `hhmmss-hhmmss`, přičemž některé meze mohou být vynechány.

Stejnolehlé meze intervalů v datu a čas jsou přitom spojeny do jednoho časového bodu. Tyto vlastnosti nejlépe ukážeme na příkladech:

birthDate	birthTime	význam pro hledání
19130113		pacienti narození dne 13.1.1913
19130101-19130131		pacienti narození v lednu 1913
19130101	080000-120000	pacienti narození dne 13.1.1913 mezi 8:00 a 16:00
19130113-19130114	120000-120000	pacienti narození mezi polednem 13.1.1913 a polednem 14.1.1913
19130113-	120000-	pacienti narození v poledne 13.1.1913 a kdykoliv později
-19130113		pacienti narození 13.1.1913 a kdykoliv dříve

Metoda může vyhodit výjimku `EGenIface`, například pokud příslušný klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění nebo parametry nelze zpracovat.

2.2.3.3. findStudies

Metoda hledá studie v databázi APFS. Vrací seznam řetězců UID všech nalezených studií. Detaily o jednotlivých studiích je pak možné zjistit metodou `getStudyInfo()`.

Metoda má dva parametry. Prvním parametrem je objekt typu `PatientInfo` a má stejný význam jako u předchozí metody `findPatients()`. Druhým parametrem je objekt typu `StudyInfo` a jeho neprázdné položky se při hledání uplatní takto:

jméno	použití
studyUid	přesné porovnání
patientUid	tato položka je ignorována
accessionNumber	porovnání s podporou zástupných znaků
referringPhysicianName	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
studyID	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
studyDescription	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
studyDate	porovnání na datum studie
studyTime	porovnání na čas studie
seriesUids	tato položka je ignorována

Podpora expanzních znaků je stejná jako v metodě `findPatients()`.

Položky `studyDate` a `studyTime` připouštějí intervaly stejně jako `birthDate` a `birthTime`.

Metoda může vyhodit výjimku `EGenIface`, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění nebo parametry nelze zpracovat.

2.2.3.4. getPatientInfo

Metoda pro zadané UID vrátí detaily pacienta ve formě `PatientInfo`. Pokud žádný takový pacient není, vrátí `null`.

Metoda může vyhodit výjimku `EGenIface`, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění.

2.2.3.5. getStudyInfo

Metoda pro zadané UID vrátí detaily studie ve formě `StudyInfo`. Pokud žádná taková studie není, vrátí null.

Metoda může vyhodit výjimku `EGenIface`, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění.

2.2.3.6. getSeriesInfo

Metoda pro zadané UID vrátí detaily série ve formě `SeriesInfo`. Pokud žádná taková série není, vrátí null.

Metoda může vyhodit výjimku `EGenIface`, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění.

2.2.3.7. getImageInfo

Metoda pro zadané UID vrátí detaily DICOM instance ve formě `ImageInfo`. Pokud takové instance není, vrátí null.

Metoda může vyhodit výjimku `EGenIface`, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění.

2.2.3.8. getImageJpeg

Metoda vrací JPEG náhled (jako pole bajtů) na DICOM instanci zadanou jejím UID. Dalšími parametry jsou:

- `frame` - celé číslo udávající, kolikátý rámec instance se má použít (rámce se počítají od nuly); pokud má instance více rámců než jeden, je tento počet uveden v položce `numberOfFrames` typu `ImageInfo`
- `size` - celé číslo udávající, jak má být vytvořený náhled velký (kolik bodů má mít jeho delší hrana); pokud zadáte nulu, náhled nebude zmenšován, zůstane v přirozené velikosti instance
- `watermark` - pravdivostní příznak, zda se má do náhledu vložit vodoznak; vodoznak se konfiguruje pro celé APFS a předpokládá se, že jde o malý znak instituce, ve které APFS běží

Metoda vyhazuje výjimku `EGenIface`, pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění, instance s daným UID neexistuje nebo JPEG náhled nelze vytvořit.

2.2.3.9. prefetchStudyToViewer

Tato metoda odešle studii zadanou jejím UID na DICOM prohlížeč zadaný dvojicí dalších parametrů: AE Title a IP adresa.

Metoda nejdříve určí prohlížeč ze zadaných parametrů AE Title a IP adresa; aspoň jeden z nich nesmí být null. Pokud metoda žádný odpovídající prohlížeč nenalezne nebo jich nalezne víc, skončí výjimkou `EGenIface`. Pak na nalezený prohlížeč odešle zadanou studii (provede DICOM C-STORE).

Metoda vyhazuje výjimku `EGenIface`, pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění, nelze jednoznačně určit prohlížeč, studie s daným UID neexistuje nebo nejde odeslat.

2.2.3.10. Zastaralé

APFS verze 2.2 ruší v rozhraní "geniface" metody `putAttachedFileData()`, `findAttachedFiles()`, `getAttachedFileInfo()` a `getAttachedFileData()`. Místo nich je k dispozici nové a zpracovanější rozhraní "docarchiface".

2.2.4. Další vlastnosti

2.2.4.1. Autentizace

Služba geniface může být zkonfigurována tak, že vyžaduje autentizaci klienta formou *WS-Security UsernameToken Profile*. Klient pak musí ke svým požadavkům přidávat SOAP security header s přihlašovacím jménem a heslem, tak jak je příslušná SOAP aplikační entita vedena v APFS.

2.2.4.2. Kontrola IP adresy

Administrátor při zakládání SOAPAE entity adává její IP adresu nebo DNS jméno. Služba geniface může ověřovat IP adresu skutečného klienta proti tomuto záznamu. Toto ověřování není ve výchozí konfiguraci zapnuté.

2.2.5. Příklady

Následují příklady opět v jazyce Python s využitím knihovny Suds. Předpokládáme, že služba geniface je zkonfigurována tak, aby vyžadovala autentizaci.

2.2.5.1. Pomocná knihovna

Nejprve si uděláme pomocnou knihovnu, která se postará o úlohy, které by se opakují ve většině testů:

- handler pro výjimky
- nastavení typu hesla v UsernameToken (knihovna Suds v době psaní tohoto manuálu neměla pro toto přímou podporu)

Vytvořte si adresář `lib`, v něm:

- prázdný soubor `__init__.py`
- vlastní knihovnu `soap.py`:

```
from suds.plugin import MessagePlugin

# WebFault handler
def handleWebFault(e):
    if (hasattr(e.fault, 'detail') and hasattr(e.fault.detail, 'EGenIface')):
        print "EGenIface caught, text: %s" % e.fault.detail.EGenIface.text
    elif (hasattr(e.fault, 'detail') and hasattr(e.fault.detail, 'EDocArchIface')):
        print "EDocArchIface caught, text: %s" % e.fault.detail.DocArchIface.text
    else:
        print "Generic WebFault caught:"
        print e

# Suds does not provide the "type" attribute on the Password tag
# of the UsernameToken in the WS-Security SOAP headers. Create
# a plugin to workaround it.
class AddPasswordType(MessagePlugin):
    def marshalled(self, context):
        password = context.envelope.childAtPath('Header/Security/UsernameToken/Password')
        password.set('Type',
            'http://docs.oasis-open.org'\
            '/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText')
```

A teď už ukážeme vlastní příklady.

2.2.5.2. Chybějící autentizační údaje

Následující program používá službu, aniž by zadal autentizační údaje:

```
import logging
from suds import WebFault
from suds.client import Client
from suds.wsse import Security, UsernameToken
from lib import soap

# Init client
logging.basicConfig(level=logging.FATAL)
client = Client('http://192.168.235.102:9090/geniface?wsdl', plugins = [soap.AddPasswordType()])

# Fail because of missing authentication data
try:
    print client.service.getVersion()
except WebFault, e:
    soap.handleWebFault(e)
```

Proto má výsledek:

```
EGenIface caught, text: Client did not supply login and password
```

2.2.5.3. Špatné heslo

Následující program zadává autentizační údaje, které jsou ale chybné:

```
import logging
from suds import WebFault
from suds.client import Client
from suds.wsse import Security, UsernameToken
from lib import soap

# Init client
logging.basicConfig(level=logging.FATAL)
client = Client('http://192.168.235.102:9090/geniface?wsdl', plugins = [soap.AddPasswordType()])

# Fail because of wrong authentication data
security = Security()
security.tokens.append(UsernameToken('soapclient', 'wrongpasswd'))
client.set_options(wsse=security)
try:
    print client.service.getVersion()
except WebFault, e:
    soap.handleWebFault(e)
```

Proto proběhne takto:

```
Generic WebFault caught:
Server raised fault: 'The security token could not be authenticated or authorized'
```

2.2.5.4. Úspěšná autentizace

Následuje program, který použije správné autentizační údaje:

```
import logging
from suds import WebFault
from suds.client import Client
from suds.wsse import Security, UsernameToken
from lib import soap

# Init client
logging.basicConfig(level=logging.FATAL)
```



```

client = Client('http://192.168.235.102:9090/geniface?wsdl', plugins = [soap.AddPasswordType()])

# Set correct authentication data
security = Security()
security.tokens.append(UsernameToken('soapclient', 'secretpasswd'))
client.set_options(wsse=security)

# Succeed
try:
    print client.service.getVersion()
except WebFault, e:
    soap.handleWebFault(e)

```

a proto uspěje:

1.02.03-REL

2.2.5.5. Zachycení výjimky EGenIface

Následující program volá metodu prefetchStudyToViewer(), ale jako parametr zadá prohlížeč s neexistujícím AE Title REV_NOVAK:

```

import logging
from suds import WebFault
from suds.client import Client
from suds.wsse import Security, UsernameToken
from lib import soap

# Init client
logging.basicConfig(level=logging.FATAL)
client = Client('http://192.168.235.102:9090/geniface?wsdl', plugins = [soap.AddPasswordType()])

# Set correct authentication data
security = Security()
security.tokens.append(UsernameToken('soapclient', 'secretpasswd'))
client.set_options(wsse=security)

# Invoke EGenIface due to wrong paramater
try:
    print client.service.prefetchStudyToViewer(
        uid='1.2.392.200036.9125.0.199402091242.1',
        aeTitle='REV_NOVAK')
except WebFault, e:
    soap.handleWebFault(e)

```

Program zachytí výjimku EGenIface a pomocí handleru handleWebFault() vypíše její položku text:

EGenIface caught, text: No application entity found for AE Title [REV_NOVAK] and IP address [null]

2.2.5.6. Průchod hierarchickou strukturou

Tento příklad demonstruje průchod hierarchickou strukturou pacient-studie-série-objekt. Následuje záznam interaktivního sezení z interpretu Python:

```

$ python
Python 2.6.4 (r264:75706, Jun  4 2010, 18:20:31)
[GCC 4.4.4 20100503 (Red Hat 4.4.4-2)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import logging
>>> from suds import WebFault
>>> from suds.client import Client
>>> from suds.wsse import Security, UsernameToken
>>> from lib import soap
>>>
>>> logging.basicConfig(level=logging.FATAL)

```

```

>>> client = Client('http://192.168.235.102:9090/geniface?wsdl', plugins = [soap.AddPasswordType()])
>>>
>>> security = Security()
>>> security.tokens.append(UsernameToken('soapclient', 'secretpasswd'))
>>> client.set_options(wsse=security)
>>>
>>> p = {}
>>> p['lastName'] = '*O*'
>>> p['birthDate'] = '19000101-19491231'
>>> p['identifiers'] = { 'namespace': 'RC', 'id': '*O*' }
>>> s = { 'accessionNumber': 'FUJI95707' }
>>>
>>> print client.service.findStudies(patientPattern=p, studyPattern=s)
[1.2.392.200036.9125.0.199402091242.1]
>>> print client.service.getStudyInfo('1.2.392.200036.9125.0.199402091242.1')
(studyInfo){
  accessionNumber = "FUJI95707"
  patientUid = "1.3.6.1.4.1.20744.3.1.2.2.3100.1275989098883.3925"
  seriesUids[] =
    "1.2.392.200036.9125.0.199402091242.1",
  studyUid = "1.2.392.200036.9125.0.199402091242.1"
}
>>> print client.service.getPatientInfo('1.3.6.1.4.1.20744.3.1.2.2.3100.1275989098883.3925')
(patientInfo){
  birthDate = "19130324"
  firstName = "TOSHIAKI"
  identifiers[] =
    (patientIdentifier){
      id = "FUJI00007"
      namespace = "RC"
    },
  lastName = "ITO"
  patientUid = "1.3.6.1.4.1.20744.3.1.2.2.3100.1275989098883.3925"
  sex = "M"
  studiesUids[] =
    "1.2.392.200036.9125.0.199402091242.1",
}
>>> print client.service.getSeriesInfo('1.2.392.200036.9125.0.199402091242.1')
(seriesInfo){
  bodyPartExamined = "CHEST"
  imagesUids[] =
    "1.2.392.200036.9125.0.19950720105640",
  modality = "CR"
  seriesUid = "1.2.392.200036.9125.0.199402091242.1"
  studyUid = "1.2.392.200036.9125.0.199402091242.1"
}
>>> print client.service.getImageInfo('1.2.392.200036.9125.0.19950720105640')
(imageInfo){
  imageType = "ORIGINAL PRIMARY"
  imageUid = "1.2.392.200036.9125.0.19950720105640"
  seriesUid = "1.2.392.200036.9125.0.199402091242.1"
  sopClassName = "Computed Radiography Image Storage"
  sopClassUid = "1.2.840.10008.5.1.4.1.1.1"
}
>>> Ctrl-D
$

```

Sezení nejdříve vytvoří z WSDL objekt `client`, který bude dělat proxy ke službě "geniface". Pak v něm nastaví autentizační údaje; až sem je to stejný začátek jako v předchozích příkladech.

V dalším kroku si sezení nachystá dvě proměnné `p` a `s` (explicitně to není uvedeno, ale budou používány jako proměnné typu `PatientInfo` a `StudyInfo`) a naplní je vyhledávacími parametry pro hledání studie. Konkrétně:

- pacientovo příjmení má obsahovat písmeno O
- pacient se narodil mezi 1.1.1900 a 31.12.1949

- pacientův identifikátor ve jmenném prostoru "RC" obsahuje znak (číslici) 0
- číslo žádanky je "FUJ195707"

Následuje vlastní hledání studie metodou `findStudies()`. Metoda našla jedinou studii; její UID je 1.2.392.200036.9125.0.199402091242.1.

Dalším krokem je vypsaní podrobností o nalezené studii. Tyto detaily zajistí metoda `getStudyInfo()`. Ta mimo jiné dodá údaje pro pohyb v hierarchické struktuře, protože zjistí:

- UID nadřazeného objektu - pacienta: 1.3.6.1.4.1.20744.3.1.2.2.3100.1275989098883.3925
- UID přímo podřízených objektů - sérií: v tomto případě jde o jedinou sérii s UID 1.2.392.200036.9125.0.199402091242.1

Další postup je zřejmý - vypsaní podrobností o pacientovi (`getPatientInfo()`) a sérii (`getSeriesInfo()`). Z detailů série čteme UID podřízených objektů, DICOM instancí. Jde o jediný objekt a jeho podrobnosti vypíše závěrečné `getImageInfo()`.

2.3. Služba docarchiface

Tato služba zpřístupňuje dokumentový archiv. Rozšiřuje a nahrazuje rozhraní Geniface a používá některé společné objekty.

2.3.1. WSDL

WSDL této služby je dostupné buď z běžícího APFS na URL:

```
http[s]://IP_adresa:TCP_port/docarchiface?wsdl
```

kde:

- `IP_adresa` je IP adresa nebo jméno počítače, na kterém běží APFS
- `TCP_port` je TCP port, na kterém APFS vystavuje SOAP rozhraní; obvykle 9090 pro HTTP a 9091 pro HTTPS

Příklad:

```
http://flex1:9090/docarchiface?wsdl
```

WSDL je k dispozici i offline. Je uloženo v souboru `conf/wsDocArchIface.wsdl` v instalačním adresáři APFS.

2.3.2. Datové typy

Služba definuje několik složených datových typů. Formální definice typů je součástí WSDL, tady je popíšeme méně formálním způsobem s důrazem jejich sémantiku.

U položek objektů datových typů je vyznačena povinnost položek * a **, vždy následuje vysvětlení. Pokud nemají obsah objekty `Diagnosis`, `CaseInfo` a `EventInfo`, vynechají se celé, v datech zprávy se neuvádí. Povinnost platí při zařazení do zprávy.

2.3.2.1. PatientIdentifier

APFS jednoznačně určuje všechny entity, a to i pacienta, unikátní identifikátorem UID. U některých entit, zejména DICOMových (studie, série, instance) toto UID vzniká obvykle na přístrojích a APFS je přebírá. U některých entit (například pacient, modalita, ...) žádný vnější zdroj UID není a tak si je APFS vytváří sám. Entita pacient je ale vyjímečná tím, že má nebo může mít několik dalších identifikátorů z vnějších zdrojů (například rodné číslo, Social Security number, interní identifikátor přidělený klinickým systémem apod.) s různou mírou jednoznačnosti a trvanlivosti. APFS tyto identifikátory uchovává a umožňuje podle nich vyhledávat.

Vnější zdroj patientského identifikátoru nazveme *jmenný prostor (namespace)*. Jeden jmenný prostor a identifikátor v něm pak představuje datový typ `PatientIdentifier`:

jméno	typ	význam
namespace	řetězec (16) **	jmenný prostor identifikátoru
id	řetězec (255) **	vlastní identifikátor

**) povinné s podmínkou. Pacient musí mít identifikátor, který je v systému nastaven jako default. Viz dále.

Pacient může mít více identifikátorů `PatientIdentifier`. Proto je položka `identifiers` v typu `PatientInfo` vícečetná.

Patientské identifikátory mají tyto vlastnosti:

- Vždy platí, že pacient má v rámci jednoho namespace nejvýše jeden identifikátor.
- V rámci jednoho namespace může mít více pacientů tentýž identifikátor, neboli APFS připouští duplicity v identifikátorech. Některé komponenty APFS sice mohou v některých situacích a ze specifických důvodů znemožňovat vytvoření duplicity, přesto obecně platí, že duplicity jsou přípustné.
- Jeden namespace je označen jako default a je vždy specifický pro konkrétní instalaci APFS. V Česku to často bývají rodná čísla ("RC"), ale mohou to být i jiné jmenné prostory. Například pokud je APFS integrován s klinickým systémem, který generuje vlastní identifikátory, pak se obvykle jako default namespace použije prostor primárního identifikátoru klinické aplikace. Default namespace se uplatňuje při ukládání nebo hledání, když k identifikátoru není uveden jeho namespace.
- nastavení default namespace v `apfs.xml` v sekci `core`:

```
patient.defaultnamespace=RC
```

Jak už jsme řekli v úvodu této kapitoly, APFS generuje vlastní řetězec UID, kterým jednoznačně identifikuje pacienta. Toto UID se strukturou `PatientIdentifier` nijak nespojuje. Je součástí typu `PatientInfo` jako položka `patientUid`.

2.3.2.2. PatientInfo

popisuje pacienta v APFS. Má následující položky:

jméno	typ	význam
patientUid	řetězec (64)	UID pacienta (nejde o DICOM atribut, ale o interní UID, které přiděluje APFS)
lastName	řetězec (32) *	příjmení
firstName	řetězec (32) **	jméno
middleName	řetězec (32)	prostřední jméno
namePrefix	řetězec (16)	předpona jména
nameSuffix	řetězec (16)	přípona jména
identifiers	seznam objektů typu <code>PatientIdentifier</code>	identifikátory pacienta; více vizte u datového typu <code>PatientIdentifier</code>
birth	xsd:timestamp	datum a případně i čas narození
death	xsd:timestamp	datum a případně i čas úmrtí
sex	řetězec (1)	nejvýše jednoznakový řetězec vyjadřující pohlaví pacienta podle číselníku DICOM (M,F,O), neboli DICOM atribut (0010,0040)
addressStreet	řetězec (32)	adresa: ulice

jméno	typ	význam
addressCity	řetězec (32)	adresa: obec
addressPostalCode	řetězec (16)	adresa: PSČ
addressState	řetězec (32)	adresa: země (v ČR nemá význam)
addressCounty	řetězec (32)	adresa: stát
attachedFilesUids	seznam řetězců	seznam UID všech AttachedFiles připojených k pacientovi

*) položka musí být vyplněna

**) položka musí být vyplněna v případě integrace s archivem DESA

2.3.2.3. Diagnosis

je datová struktura popisující diagnózu. Obsahuje následující atributy:

jméno	typ	význam
code	řetězec (128)	kód diagnózy
codeSys	řetězec (256)	číselník diagnóz
priority	celé číslo	priorita
text	řetězec (1024)	text diagnózy

*) položka musí být vyplněna

2.3.2.4. CaseInfo

popisuje klinický případ:

jméno	typ	význam
idNamespace	řetězec (32) *	jméno systému, který přidělil číslo klinického případu (neboli jeho namespace)
idValue	řetězec (64) *	číslo klinického případu
type	řetězec (32) *	typ klinického případu
begin	timestamp	začátek klinického případu
end	timestamp	jeho konec

*) položka musí být vyplněna

2.3.2.5. EventInfo

popisuje klinickou událost:

jméno	typ	význam
idNamespace	řetězec (32) *	jméno systému, který přidělil číslo klinické události (neboli její namespace)
idValue	řetězec (64) *	číslo klinické události

jméno	typ	význam
type	řetězec (32) *	typ klinické události
begin	timestamp *	okamžik (resp. začátek) klinické události
departIdent	řetězec (16) *	zkratka oddělení, na kterém k události došlo
departName	řetězec (32) *	jméno oddělení
originatorFirstName	řetězec (32)	jméno osoby, která je původcem klinické události
originatorLastName	řetězec (32)	příjmení původce
originatorMiddleName	řetězec (32)	prostřední jméno původce
originatorPrefix	řetězec (16)	titul (před jménem) původce
originatorSuffix	řetězec (16)	titul (za jménem) původce
diagnoses	seznam Diagnosis	diagnózy

*) položka musí být vyplněna

2.3.2.6. AttachedFileInfo

popisuje dokument (všechny verze naráz) - obvykle lékařskou zprávu, která byla připojená k pacientovi.

jméno	typ	význam
attachedFileUid	řetězec (64)	UID lékařské zprávy (toto UID přiděluje APFS)
patientUid	řetězec (64)	UID nadřazeného pacienta (UID, které pacientům přiděluje APFS)
description	řetězec (128)	libovolný uživatelský popis zprávy
producer	řetězec (32) *	identifikace původce dat (systém, který zprávu vytvořil, nemocnice apod.)
ident	řetězec (64) *	jednoznačný identifikátor, který zprávě její <i>producer</i> přidělil
type	řetězec (32) *	typ zprávy podle default naplnění číselníku, obsah číselníku je možné změnit: <ul style="list-style-type: none"> • A - ambulantní zpráva • H - hospitalizační zpráva • R - radiologická zpráva • L - laboratorní zpráva • O - jiný typ
caseIdent	řetězec	identifikátor klinického případu, ke kterému zpráva náleží (zastaralá položka; namísto ní používejte CaseInfo.idValue)
physic	řetězec (64)	lékař, který zprávu vytvořil
depart	řetězec (64)	skupina (oddělení), které zprávu vytvořilo (jde o skupinu ve smyslu řízení přístupu k datům)
origin	timestamp	datum a čas vytvoření zprávy (pokud jej ukládající systém nezadá, APFS ho při ukládání doplní aktuálním časem)
destinationName	řetězec (64)	cílové zdrav. zařízení (v případě převozu pacienta)
versionsUids	seznam řetězců	seznam UID všech jednotlivých verzí dokumentu

jméno	typ	význam
rdAcr	řetězec (8) **	kód skartačního režimu
rdAction	řetězec (1) **	skartační znak - jeden ze znaků A/S/V
rdEventType	řetězec (32)	kód spouštěcí události, Hodnota odpovídá spisovému a skartačnímu plánu. Není-li vyplněno, znamená na základě data uzavření - AFTER_ENTITY_CLOSED. Pro externí událost je hodnota AFTER_EXTERNAL_EVENT.
rdFlag	řetězec (1) **	aplikace lhůty - hodnota 0 nebo 1. Obsahuje hodnotu 1, pokud se lhůta nepočítá na základě data uzavření, ale na základě jiné skutečnosti (AFTER_EXTERNAL_EVENT)
rdPeriod	int **	skartační lhůta v celých letech
szAcr	řetězec (8) **	kód spisového znaku, dle standardu jde o <nssess2:PlneUrcenySpisovyZnak> podle spisového plánu
reportType	řetězec (1)	

*) položka musí být vyplněna

**) položka musí být vyplněna, pokud je parametr v apfs.xml sekce digitalarchive catalog.compare.values nastaven na true. Potom se kontrolují hodnoty oproti číselníku recordclassificationcatitem a shredregimecatitem (spisový a skartační plán).

AttachedFile jako takový neobsahuje vlastní dokumenty, je to jenom "kontejner" pro jednotlivé verze. Teprve verze, neboli AttachedFileVersions, obsahují vlastní dokument.

2.3.2.7. AttachedFileVersionInfo

popisuje jednotlivou verzi dokumentu:

jméno	typ	význam
attachedFileVersionUid	řetězec (64)	UID verze (toto UID přiděluje APFS)
attachedFileUid	řetězec (64)	UID nadřazeného AttachedFile neboli "mateřského" dokumentu
description	řetězec (128)	libovolný uživatelský popis verze
docver	celé číslo	pořadové číslo verze
contentType	řetězec (255) *	identifikátor formátu
encoding	řetězec (64)	kódování dokumentu; upřesňuje contentType
stateCode	řetězec (16)	stav zpracování, v případě integrace s DESA stav zpracování včetně přijetí v DESA

*) položka musí být vyplněna

Tabulka stavů

Hodnoty stavů, kde zdrojem je archiv DESA jsou dosazeny pouze v případě aktivní integrace s DESA. Nastavení je popsáno v servisním manuálu.

kód stavu	zdroj	význam
AZD_OK	AZD (apfs)	Uloženo v apfs.
AZD_ERR	AZD (apfs)	Uloženo v apfs, při pokusu o předání do DESA vznikla chyba.
AI_RECEIVED	DESA	Obsah přijat pro zpracování

kód stavu	zdroj	význam
AI_AV_INP_OK	DESA	Vstupní antivirová kontrola v pořádku, čeká v karanténě
AI_FORM_OK	DESA	Kontrola formátu obsahu v pořádku
AI_INFECTED	DESA	Obsah neprošel antivirovou kontrolou
AI_QA_ERR	DESA	Interní chyba při kontrole formátu nebo antivirové kontrole
AI_IC_OK	DESA	Vstupní zpracování obsahu ukončeno
AI_EDIT	DESA	Neúplný nebo chybný obsah připraven pro úpravy archivářem / administrátorem
AI_REJECT	DESA	Uložení obsahu zamítnuto z důvodu neopravitelných chyb v obsahu, obsah vrácen původci k opravě
AI_INVALID	DESA	Uložení obsahu nemožné z důvodu chyb v obsahu, obsah může být doplněn archivářem nebo vrácen původci k opravě podle nastavení systému
AI_DM_OK	DESA	Uložení metadat do provozního systému
AI_AS_REP	DESA	Čeká na opakovaný pokus ukládání do archivního úložiště
AI_AS_OK	DESA	Uložení do archivního úložiště v pořádku
AI_ACC_REP	DESA	Čeká na opakovaný pokus o uložení do přístupového modulu
AI_ACC_OK	DESA	Uložení do přístupového modulu v pořádku, obsah zpracován
AI_ERROR	DESA	Interní chyba při zpracování nebo ukládání

Připojíme ještě několik detailů a komentářů:

- Jednoznačným "interním" identifikátorem dokumentu je UID a přiděluji ji APFS sám při ukládání. Dokumenty mají ale i jednoznačný "vnější" identifikátor - tím je dvojice (producer, ident). APFS odmítne uložit zprávu, pokud některý z atributů (producer, ident) chybí.
- Pokud ukládající systém pošle dokument s dvojicí (producer, ident) vícekrát, APFS je přijme jako nové verze téhož dokumentu (bez ohledu na to, jestli jde o binárně tatáž data). Každé verzi přidělí docver tak, že o jedničku zvýší dosavadní nejvyšší verzi.
- Jednoznačným "interním" identifikátorem verze dokumentu je UID, které APFS sám přidělí při ukládání, je jedinou návratovou hodnotou při úspěšném uložení nové verze. V aktivním režimu elektronického podpisu, kdy se provádí validace,

2.3.2.8. DesaRDProcessChangesInfo

Popisuje stav package (verze dokumentu) při procesu skartace v DESA:

jméno	typ	význam
acronym	řetězec (255)	acronym skartačního řízení
attachedFileVersionUid	řetězec (64)	shredrecord.reguuid - po skartaci zůstává už jen záznam o skartované položce (toto UID přiděluje APFS)
attachedFileUid	řetězec (64)	UID dokumentu (nadráženého verzi neboli "mateřského" dokumentu), pokud UID není vyplněno byl dokument v APFS již skartován
rdProcState	řetězec (16)	stav procesu skartace
rdState	řetězec (16)	stav dokumentu v procesu skartace
time	datetime	datum a čas provedení skartace (poslední změna stavu)

2.3.2.9. DesaRDProcessStateInfo

Popisuje stav procesu skartace v DESA.

jméno	typ	význam
acronym	řetězec (255)	akronym skartačního řízení
determinationDate	datetime	rozhodné datum skartace
rdProcState	řetězec (16)	stav procesu skartace
rdProcType	řetězec (16)	typ skartačního řízení
time	datetime	čas poslední změny stavu skartačního řízení

2.3.2.10. EDocArchIface

je výjimka, kterou mohou vyhazovat všechny metody služby "docarchiface". Má jediný atribut:

jméno	typ	význam
text	řetězec	detaily toho, proč k výjimce došlo

2.3.3. Metody

2.3.3.1. getVersion

Tato metoda nemá parametry. Vrací verzi APFS. Může vyhodit výjimku `EGenIface`, například pokud příslušný klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění.

2.3.3.2. putAttachedFileVersionData

Metoda uloží dokument.

Metoda má následující parametry:

- prvním parametrem je objekt typu `PatientInfo` a slouží k vyhledání nebo založení pacienta,
- druhým parametrem je objekt typu `CaseInfo` a nese popis klinického případu; může být null
- dalším parametrem je objekt typu `EventInfo` a nese popis klinické události; může být null
- následuje parametr typu `AttachedFileInfo`; nesmí být null a jeho hlavní "úkol" je přinést identifikaci dokumentu (producer a ident). Naproti tomu atributy `attachedFileUid`, `patientUid` a `versionsUids` budou ignorovány.
- následující parametr je typu `AttachedFileVersionInfo`; opět nesmí být null přenáší hlavně `contentType` a `encoding`. Naproti tomu atributy `attachedFileVersionUid` a `attachedFileUid` budou ignorovány. Atribut `docver` může (a typicky by měl) být null, v takovém případě APFS přidělí číslo verze sám.
- posledním parametrem je pole bajtů, což je vlastní ukládaný dokument

Metoda vrací jedinou hodnotu UID verze dokumentu.

Metoda může vyhodit výjimku `EGenIface`, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění, údaje nejsou dostatečné nebo jde o duplicitní uložení (ukládající systém zadal `docver`, ale hodnota již existuje).

2.3.3.3. getAttachedFileVersionData

Metoda vyzvedne verzi dokument.

Metoda má jediný parametr, UID verze dokumentu.

Vyhledání verze proběhne dle zadaného UID verze dokumentu. Pokud se nenajde, hledá se dle UUID (zadaný parametr služby je považován za UUID). Pokud se nenajde, hledá se verze dle UID v tabulce souborů, které byly rozšířeny podpisem nebo časovým razítkem (při této operaci dochází ke změně UID) a jsou vrácena data verze dokumentu po operaci.

Metoda může vyhodit výjimku EDocArchIface, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění nebo pokud dojde k vnitřní chybě.

2.3.3.4. getAttachedFileMaxVersionData

Metoda vyzvedne maximální verzi dokument.

Metoda má jediný parametr, UID dokumentu.

Metoda může vyhodit výjimku EDocArchIface, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění nebo pokud dojde k vnitřní chybě.

2.3.3.5. getAttachedFileVersionInfo

Metoda vyzvedne metadata o verzi dokumentu.

Metoda má jediný parametr, UID verze dokumentu.

Vyhledání verze proběhne dle zadaného UID verze dokumentu. Pokud se nenajde, hledá se dle UUID (zadaný parametr služby je považován za UUID). Pokud se nenajde, hledá se verze dle UID v tabulce souborů, které byly rozšířeny podpisem nebo časovým razítkem (při této operaci dochází ke změně UID) a jsou vrácena metadata verze dokumentu po operaci.

Metoda může vyhodit výjimku EDocArchIface, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění nebo pokud dojde k vnitřní chybě.

2.3.3.6. getAttachedFileVersionStatus

Metoda vrací stav zpracování verze dokumentu, Tabulka stavů je uvedena u objektu AttachedFileVersionInfo. Metoda je určena zejména pro režim spolupráce, tj. ukládání do externího archivu DESA.

Metoda má jediný parametr, UID verze dokumentu, návratová hodnota je řetězec v poli return.

Vyhledání verze proběhne dle zadaného UID verze dokumentu. Pokud se nenajde, hledá se dle UUID (zadaný parametr služby je považován za UUID). Pokud se nenajde, hledá se verze dle UID v tabulce souborů, které byly rozšířeny podpisem nebo časovým razítkem (při této operaci dochází ke změně UID) a je vrácen status verze dokumentu po operaci.

Metoda může vyhodit výjimku EDocArchIface, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění nebo pokud dojde k vnitřní chybě.

2.3.3.7. getAttachedFileInfo

Metoda vyzvedne metadata o dokumentu.

Metoda má jediný parametr, UID dokumentu.

Metoda může vyhodit výjimku `EDocArchIface`, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění nebo pokud dojde k vnitřní chybě.

2.3.3.8. `getCaseInfoOfAF`

Metoda vyzvedne informace o klinickém případě, které (pokud vůbec) byly uloženy spolu s dokumentem dokumentem.

Metoda má jediný parametr, UID dokumentu.

Metoda může vyhodit výjimku `EDocArchIface`, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění nebo pokud dojde k vnitřní chybě.

2.3.3.9. `getEventInfoOfAF`

Metoda vyzvedne informace o klinické události, které (pokud vůbec) byly uloženy spolu s dokumentem dokumentem.

Metoda má jediný parametr, UID dokumentu.

Metoda může vyhodit výjimku `EDocArchIface`, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění nebo pokud dojde k vnitřní chybě.

2.3.3.10. `getPatientInfo`

Metoda vyzvedne informace o pacientovi.

Metoda má jediný parametr, UID pacienta.

Metoda může vyhodit výjimku `EDocArchIface`, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění nebo pokud dojde k vnitřní chybě.

2.3.3.11. `findAttachedFiles`

Metoda vyhledá všechny dokumenty, které vyhovují vyhledávacím kritériím a jež je klient oprávněn číst. Vrací seznam řetězců UID všech nalezených dokumentů. Pokud hledáte určitou verzi dokumentu, použijte metodu `findAttachedFilesVersions`.

APFS interpretuje zadané hodnoty neboli vyhledávací kritéria:

- pokud je zadán parametr `patientPattern` typu `PatientInfo`, pak jeho jednotlivé položky (pokud budou různé od NULL) budou pro vyhledávání znamenat:

jméno	použití
<code>patientUid</code>	přesné porovnání
<code>lastName</code>	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
<code>firstName</code>	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
<code>middleName</code>	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
<code>namePrefix</code>	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
<code>nameSuffix</code>	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
<code>identifiers</code>	porovnání s podporou zástupných znaků; pokud není uveden namespace, použije se místo něj default namespace

jméno	použití
birth	porovnání na datum narození (ze zadané položky bude APFS ignorovat hodiny a další přesnější údaje, jinak řečeno, v položce je důležitá jenom datumová část)
sex	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
addressStreet	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
addressCity	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
addressPostalCode	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
addressState	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
addressCountry	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
attachedFilesUids	nemá pro hledání význam

Podpora zástupných znaků (wildcards) znamená, že lze zadat zástupné znaky:

- * - tento znak představuje řetězec libovolné délky (i nulové)
- ? - tento znak představuje jediný znak
- pokud je zadán parametr casePattern typu CaseInfo, pak jeho jednotlivé položky pak jeho jednotlivé položky (pokud budou různé od NULL) budou pro vyhledávání znamenat:

jméno	použití
idNamespace	porovnání s podporou zástupných znaků
idValue	porovnání s podporou zástupných znaků
type	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
begin	porovnání na datum začátku klinického případu (ze zadané položky bude APFS ignorovat hodiny další přesnější údaje, jinak řečeno, v položce je důležitá jenom datumová část)
end	porovnání na datum konce klinického případu (ze zadané položky bude APFS ignorovat hodiny další přesnější údaje, jinak řečeno, v položce je důležitá jenom datumová část)

- pokud je zadán parametr eventPattern typu EventInfo, pak jeho jednotlivé položky pak jeho jednotlivé položky (pokud budou různé od NULL) budou pro vyhledávání znamenat:

jméno	použití
idNamespace	porovnání s podporou zástupných znaků
idValue	porovnání s podporou zástupných znaků
type	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
begin	porovnání na datum klinické události (ze zadané položky bude APFS ignorovat hodiny další přesnější údaje, jinak řečeno, v položce je důležitá jenom datumová část)
departIdent	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
departName	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
originatorFirstName	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
originatorLastName	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
originatorMiddleName	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
originatorPrefix	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků

jméno	použití
originatorSuffix	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
diagnoses	nemá pro hledání význam

- pokud je zadán parametr attachedFilePattern typu AttachedFileInfo, pak jeho jednotlivé položky pak jeho jednotlivé položky (pokud budou různé od NULL) budou pro vyhledávání znamenat:

jméno	použití
attachedFileUid	přesné porovnání
patientUid	nemá pro hledání význam
description	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
producer	porovnání s podporou zástupných znaků
ident	porovnání s podporou zástupných znaků
type	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
caseIdent	zastaralá položka, nemá na vyhledávání vliv; místo ní používejte CaseInfo.idValue
begin	porovnání na datum klinické události (ze zadané položky bude APFS ignorovat hodiny další přesnější údaje, jinak řečeno, v položce je důležitá jenom datumová část)
physic	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
depart	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
origin	porovnání na datum vytvoření (ze zadané položky bude APFS ignorovat hodiny další přesnější údaje, jinak řečeno, v položce je důležitá jenom datumová část)
versionsUids	nemá pro hledání význam

Metoda může vyhodit výjimku EDocArchIface, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění nebo pokud dojde k vnitřní chybě.

2.3.3.12. findAttachedFilesVersions

Metoda vyhledá všechny verze dokumentů, které vyhovují vyhledávacím kritériím a jež je klient oprávněn číst. Vrací seznam řetězců UID všech nalezených verzí dokumentů.

Parametry metody neboli vyhledávací kritéria jsou stejná jak u metody findAttachedFiles, jejich výčet a postup interpretace najdete v předchozí kapitole.

Výše uvedené parametry jsou doplněny o údaje k verzi dokumentu:

- pokud je zadán parametr attachedFileVersionPattern typu AttachedFileVersionInfo, pak jeho jednotlivé položky pak jeho jednotlivé položky (pokud budou různé od NULL) budou pro vyhledávání znamenat:

Jméno	Použití
attachedFileVersionUid	přesné porovnání
attachedFileUid	nemá pro hledání význam
description	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
docver	přesné porovnání
contentType	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků
encoding	porovnání bez ohledu na malá/velká písmena a s podporou zástupných znaků

Metoda může vyhodit výjimku EDocArchIface, například pokud klient (SOAP aplikační entita) nemá v APFS dostatečná oprávnění nebo pokud dojde k vnitřní chybě.

2.3.3.13. getRDProcessChanges

Vyzvednutí seznamu změn skartačního řízení při integraci s DESA. Metoda vyzvedne seznam dokumentů, které byly skartovány, archivovány nebo zařazeny do skartačního řízení. Vstupem je označení původce a časové období, do něž spadá provedení operací nad dokumenty uvedenými na výstupu funkce. Výstupem je seznam identifikátorů dokumentů se stavem v rámci skartačního řízení a aktuálním stavem skartačního řízení, do něž byl dokument zahrnut. Současně je výstupem čas provedení archivace nebo skartace.

Metoda má 2 parametry - kód původce (producer) a určení období (datum).

Metoda vrací seznam objektů DesaRDProcessChangesInfo. Pokud již došlo ke skartaci v AZD (apfs) je vráceno pouze UID verze dokumentu, mateřský dokument již není k dispozici.

Tabulka stavy dokumentů ve skartačním řízení (RDState):

Zkratka stavu	Název stavu
RD_DISCARD	Dokument je navržen ke skartaci
RD_ARCHIVE	Dokument je navržen k archivaci
RD_DELAYED	Skartace nebo archivace odložena
RD_DISCARDED	Obsah byl skartován včetně metadat
RD_ARCHIVED	Dokument byl úspěšně předán do archívu
RD_A_DISCARDED	Dokument archivován - obsah není k dispozici
RD_D_AVAILABLE	Proběhla skartace - obsah dosud k dispozici
RD_DECIDE	Dokument s volbou skartační operace
RD_RELOCATE	Dokument navržen k rozluce
RD_RELOCATED	Rozluka byla provedena
RD_R_DISCARDED	Rozluka byla provedena, obsah není k dispozici

2.3.3.14. getRDProcessState

Stav skartačního řízení jako celku při integraci s DESA. Současně se vrací čas poslední změny stavu skartačního řízení a další jeho parametry nastavené při založení.

Metoda má 3 parametry - kód původce (producer), určení období (datum) a akronym - skartační znak, např "S5".

Metoda vrací seznam objektů DesaRDProcessStateInfo.

Tabulka stavy skartačního řízení (RDProcState):

Zkratka stavu	Název stavu
RDP_PROCEED	Probíhá skartace a archivace
RDP_FINISHED	Skartace a archivace ukončena

Typy skartačního řízení (RDProcType):

Zkratka stavu	Název stavu
RDT_INT	Interní skartace, u spisovny je možné až po proběhnutí řádného skartačního řízení, pokud jsou některé digitální dokumenty ponechány v kopii i u původce.
RDT_RD	Skartace a archivace ukončena
RDT_REL	Spisová rozluka - předání do jiné organizace.

2.3.3.15. savePatient

Zajišťuje změnu údajů pacienta.

Metoda má jediný parametr PatientInfo.

V případě aktivního režimu DESA zajistí zavolání metody updatePackageMetadata pro každý dokument tohoto pacienta.

Metoda vrací UID pacienta.

2.3.3.16. saveAttachedFileInfo

Zajišťuje změnu metadat u již uložených dokumentů, v rozsahu atributů:

```
SzAcr, RdAcr, RdPeriod, RdAction
```

Metoda má jediný parametr AttachedFileInfo, kde musí být vyplněné kromě změněných atributů také:

```
producer, ident, type
```

V případě aktivního režimu DESA zajistí zavolání metody updatePackageMetadata pro tento dokument.

Metoda vrací řetězec "Changes Saved".

2.3.3.17. updateExternalEvent

Zajišťuje doplnění nebo změnu údajů skartace pro dokumenty ve skartačních režimech EXTERNAL EVENT. Metoda se použije pro nastavení data externí spouštěcí události externalEventDate. Podle skartačního plánu se může jednat například o datum konce hospitalizace, datum podání přípravku, datum vyřazení z evidence nebo také datum úmrtí.

Nastavení má efekt pouze u dokumentů, jejichž skartační režim je nastavený jako externí (AFTER_EXTERNAL_EVENT). U ostatních se nepoužije. Údaj určuje, kdy začíná běžet skartační lhůta pro dokument, dokud není údaj nastaven, není vybrán do skartačního řízení. V případě aktivního režimu DESA je datum skartační události předáno k příslušnému SIP balíčku v DESA.

Metoda má parametry:

```
producer - kod původce dat
ident - interní identifikáto původce
attachedFileUid - uid attachedFile
externalEventDate - datum externí události YYYY-MM-DD
```

Metoda vrací řetězec "Changes Saved".

2.3.3.18. deprecateAttachedFile

Zajišťuje stornování dokumentu v archivu. Metoda se použije pro označení dokumentu (attachedfile) jako stornovaný (Deprecated), označení se týká všech uložených verzí v attachedfileversion. Pokud je zapnutá synchronizace tak je informace přenesena do registru dokumentů.

Metoda má jediný parametr:

```
String UID dokumentu, tj UID attachedfile
```

Metoda vrací řetězec "Deprecated".

2.3.4. Další vlastnosti

Autentizace a kontrola IP adresy mají stejné možnosti nastavení jako je v případě "geniface".

2.3.5. Příklady

Následují příklady opět v jazyce Python s využitím knihovny Suds. Nebudeme se věnovat příkladům, které se týkají autentizace, protože ty by byly stejné jako v případě rozhraní "geniface". Použijeme i stejnou pomocnou knihovnu `lib`.

2.3.5.1. Uložení `AttachedFileVersion`

Tento příklad předvádí "životní cyklus" lékařské zprávy (uložení, vyhledání, stažení). Opět půjde o záznam interaktivního sezení z interpretu Python:

```
Python 2.7 (r27:82500, Sep 16 2010, 18:02:00)
[GCC 4.5.1 20100907 (Red Hat 4.5.1-3)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import logging
>>> from suds import WebFault
>>> from suds.client import Client
>>> from suds.wsse import Security, UsernameToken
>>> from lib import soap
>>> import base64
>>> import filecmp
>>>
>>> logging.basicConfig(level=logging.FATAL)
>>> client = Client('http://192.168.235.102:9090/docarchiface?wsdl',
                    plugins = [soap.AddPasswordType()])
>>>
>>> security = Security()
>>> security.tokens.append(UsernameToken('soapclient', 'secretpasswd'))
>>> client.set_options(wsse=security)
>>>
>>> p = {}
>>> p['lastName'] = 'Novak'
>>> p['firstName'] = 'Josef'
>>> p['birth'] = '1968-08-21'
>>> p['identifiers'] = { 'namespace':'RC', 'id':'6808211111' }
>>>
>>> r = {}
>>> r['description'] = 'zvukova ambulantni zprava'
>>> r['producer'] = 'AMISH'
>>> r['ident'] = '345768123'
>>> r['type'] = 'A'
>>> r['physic'] = 'Milan Vavra'
>>> r['depart'] = 'NEUAMB'
>>>
>>> v = {}
>>> v['contentType'] = 'audio/mpeg'
>>> v['encoding'] = 'mp3'
>>>
>>> file = open("/temp/novakova_zprava.mp3", "rb")
>>> b = base64.b64encode(file.read())
>>> file.close()
```



```

>>>
>>> print client.service.putAttachedFileVersionData(p, None, None, r, v, b)
1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.177
>>> print client.service.putAttachedFileVersionData(p, None, None, r, v, b)
1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.184
>>>
>>> patientPattern = {}
>>> patientPattern['lastName'] = '*nov*'
>>> patientPattern['birth'] = '1968-08-21'
>>> print client.service.findAttachedFilesVersions(patientPattern, None, None, None, None)
[1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.184, 1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.177]
>>>
>>> print client.service.getAttachedFileVersionInfo('1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.184')
(attachedFileVersionInfo){
  attachedFileUid = "1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.175"
  attachedFileVersionUid = "1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.184"
  contentType = "audio/mpeg"
  docver = 2
  encoding = "mp3"
}
>>>
>>> print client.service.getAttachedFileInfo('1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.175')
(attachedFileInfo){
  attachedFileUid = "1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.175"
  description = "zvukova ambulantni zprava"
  ident = "345768123"
  origin = 2012-05-25 08:27:02.000864
  patientUid = "1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.172"
  physic = "Milan Vavra"
  producer = "AMISH"
  type = "A"
  versionsUids[] =
    "1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.177",
    "1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.184",
}
>>>
>>> print client.service.getPatientInfo('1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.172')
(patientInfo){
  attachedFilesUids[] =
    "1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.175",
  birth = 1968-08-21 00:00:00
  firstName = "Josef"
  identifiers[] =
    (patientIdentifier){
      id = "6808211111"
      namespace = "RC"
    },
  lastName = "Novak"
  patientUid = "1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.172"
}
>>>
>>> b = client.service.getAttachedFileVersionData('1.3.6.1.4.1.20744.3.1.2.2.33.1337930598589.184')
>>> b = base64.b64decode(b)
>>> file = open("/temp/stazena_zprava.mp3", "wb")
>>> file.write(b)
>>> file.close()
>>>
>>> print filecmp.cmp("/temp/novakova_zprava.mp3", "/temp/stazena_zprava.mp3")
True
>>> Ctrl-D
$

```

Sezení nejdříve vytvoří z WSDL objekt `client` jako proxy ke službě "docarchiface" a nastaví v něm autentizační údaje.

V dalším kroku si sezení nachystá tři proměnné `p`, `r` a `v` (explicitně to není uvedeno, ale budou používány jako proměnné typu `PatientInfo`, `AttachedFileInfo` a `AttachedFileVersionInfo`) a naplní je údaji pro založení pacienta, ambulantní zprávy a její verze ve formátu mp3. Do čtvrté proměnné si nachystá vlastní zprávu - zvukový soubor.

Řádek **print client.service.putAttachedFileVersionData(p, null, null, r, v, b)** tuto zvukovou ambulantní zprávu uloží do APFS a vypíše UID, které APFS zprávě (resp. této její verzi) přidělil. Bezprostředně poté soubor uložíme ještě jednou (s původními metadaty), v APFS tak vznikne další verze téhož dokumentu.

V sezení následuje hledání dokumentů (resp. jejich verzí) pro pacienty zadané datem narození a příjmením podle vzoru **"*nov*"**. Mezi výsledky jsou i UID, které jsme získali při ukládání dokumentů.

Pro jedno z nalezených UID pak sezení postupně vypisuje podrobnosti k verzi, dokumentu a pacientovi.

Posledním krokem je stažení vybrané verze metodou **getAttachedFileVersionData**, uložení do souboru a porovnání s původním dokumentem.

2.4. Služba docrepoiface

Tato služba implementuje IHE profil XDS.b v roli DocumentRepository.

<https://server:9091/docrepoiface?wsdl>

2.4.1. ITI-41, RAD-68

DocumentRepository_ProvideAndRegisterDocumentSet-b

Uložení dokumentu včetně metadat do archivu.

Poznámka: archiv po příslušném nastavení zajistí registraci dokumentů v document registry (ITI-42).

2.5. Služba docregiface

Tato služba implementuje IHE profil XDS.b,

<https://server:9091/docregiface?wsdl>

2.5.1. ITI-57 documentRegistryUpdateDocumentSet

Aktualizace metadat dokumentu, uložení nebo zneplatnění dokumentu.

Kapitola 3. Rozhraní STA/LTA REST API

Programové rozhraní APFS podporuje také prostředek pro komunikaci mezi servery krátkodobého archívu(STA) a serverem dlouhodobého archívu(LTA) - protokol REST. Jako transportní protokol pro REST je použito HTTP. Rozhraní obsahuje jednu službu : sendChangesFromSTA : zaslání provedených změn

3.1. Služba sendChangesFromSTA

Provádí odeslání záznamů o změnách na LTA. Parametry jsou autentizační řetězec(HeaderParam authorization) a objekt XmlSendEvent. Autentizační řetězec(Basic) obsahuje přihlašovací jméno a heslo. Heslo při zaslání aplikace upraví kódováním Base64. Jméno a heslo se nastavuje v konfiguračním souboru apfs.xml pod parametry rest.services.sendchanges.username a rest.services.sendchanges.password. Identické jméno a heslo musí být nastaveno i na LTA.

Objekt XmlSendEvent obsahuje :

```
sta      : aetitle krátkodobého archívu STA pro identifikaci odkud změna přišla.
eventType : typ provedené změny, hodnoty jsou :
    MERGE_PATIENTS : slučování 2 pacientů, vzniká přes GUI nebo zasláním ADT zprávy
    MOVE_STUDY     : přesouvání studie, vzniká přes GUI
    MOVE_SERIES    : přesouvání série, vzniká přes GUI
    MOVE_INSTANCES : přesouvání snímků, vzniká přes GUI
    CREATE_PATIENT : založení nového pacienta, vzniká přes GUI nebo přes ADT zprávu
    UPDATE_PATIENT : oprava základních dat a adresy pacienta (probíhá přes GUI i přes ADT zprávu). Pokud podle stáva
    SHRED_PATIENT  : skartace pacienta, zrušení patientských dat a všech jeho dotčených údajů z databáze a z úložiště
    SHRED_STUDY   : skartace studie, zrušení studie dat a všech jejích dotčených údajů z databáze a z úložiště, zazn
    SHRED_SERIES  : skartace série, zrušení série dat a všech jejích dotčených údajů z databáze a z úložiště, zaznam
    SHRED_IMAGE   : skartace snímku, zrušení snímku z databáze a z úložiště, zaznamená se do shred tabulek
patient       : JSON řetězec základních údajů o pacientovi u kterého se změna provádí. Protože není 100% zaručeno,
mergedpat     : JSON řetězec základních údajů o pacientovi, který se slučuje. Struktura stejná jako v údajích patient
eventData     : JSON řetězec odkazů na data strukturovaný dle typu události v eventType
createdUser   : kdo provedl změnu
created       : kdy byla změna provedena na STA
changeId      : Id záznamu tabulky change_events na STA
```

Změny se zasílají metodou POST.

Při změně na STA dle uvedených typů eventu dojde k založení záznamu do tabulky change_events se stavem change_status='A' a s aktuálním datem a časem zápisu.

Jednotlivé stavy, které mohou nastat při zaslání změn na LTA :

```
A = založení záznamu(jako aktivní, připraven k odeslání)
E = chyba při uložení, ve sloupci message je daná chyba zobrazena
S = změna zasláná na LTA
```

Pokud se nepodaří daná změna založit(stav jde do 'E') nebo pokud natrefí při dalším spuštění na záznam, který má stav 'E', pak to zapíše do logu a skončí, nejdříve se musí daná chyba vyřešit, protože další změny mohou být na předchozí nezaslané změně závislé.

Ale k tomu, že se provede daná změna a neprovede se záznam do tabulky change_events, by nemělo docházet. Pokud je záznam ve stavu 'S', pak je změna odeslána a LTA ji přijalo bez chyb (založilo do tabulky accept_changes). Pokud je nastaven parametr "sta.remove_after_success_send" na true, pak se daný záznam smaže z tabulky change_events.

3.2. Služba acceptChangesOnLTA

Změny z jednotlivých STA se zapisují do tabulky accept_changes na LTA ve stavu 'A' s datem a časem zápisu, kdy byly provedeny na STA. Aby se změny prováděly na LTA, musí být zpřístupněna služba acceptChangesOnLTA a jednotlivé eventy, které chce uživatel provádět, nastaveny na true.

Jednotlivé stavy zpracování :

```
A = změna se zapsala ke zpracování
P = změna se zpracovává
E = nastala chyba při provádění změny, zalogováno do logu i na daném db záznamu ve sloupci message
C = změna, která z nějakého důvodu nelze provést, stav se nastavuje ručně v db (Cancel) nebo
    přes GUI tlačítkem Zrušit na seznamu Neprovedeno-chyba
D = změna se promítla bez chyb
```

Nejdříve se v db hledá, zda neexistuje záznam ve stavu 'P', pokud ano, pak se služba ukončí a čeká na opakované použití dle časovače. Pak se načtou záznamy (počet závisí od nastavení parametru služby lta.batch.count=...) Pokud nastane při sekvenčním zpracování k chybě, označí se záznam stavem 'E' a zapíše se chyba do logu a do db(sloupec message). Musí se vyřešit, jinak další změny nebudou při opakovaném spouštění služby provádět.

Chybové řádky se dají zkoumat a řešit v GUI v menu Sledování a správa, kde je ikonka Provedené změny, kliknutím na radiobutton Neprovedeno-chyba se dané záznamy zobrazí. Pak po zkoumání případně logu administrátor rozhodne, zda chybový záznam zaktivovat (poté jej služba vykoná znovu) nebo zrušit(slужba ji nebere v potaz a pokračuje v provádění dalších změn). Pokud se změna promítne, záznam se označí jako 'D'

3.3. URL

Služba je dostupná z běžícího APFS na URL:

```
http://IP_adresa:TCP_port/services/rest/sendEvents
```

kde:

IP_adresa je IP adresa, na kterém běží APFS dlouhodobého archívu LTA

TCP_port je TCP port, na kterém APFS vystavuje REST rozhraní;

dle default nastavení v konfiguraci apfs.xml 8095 pro HTTP

Kapitola 4. Rozhraní shred/manage documents REST API

REST rozhraní určené pro

1. správu a řízení procesu skartace s modulem vyřazování ("ESKA") - disposal REST API
2. ukládání a správu dokumentů v archivu - manage document REST API

4.1. disposal REST API

Execute final disposal operation (shred or tag with final state)

```
@Path("/")
public interface ExecuteDisposalApi

@Path("/")
public interface ManageArchivesApi
```

4.1.1. Execute final disposal operation (shred or tag with final state)

Execute final package disposal operation on single package. Depending on disposal process type, setting and package disposal action the package is shredded or an appropriate tag is applied. Run after transfer to archive is committed at Disposal Service side.

```
@POST
@Path("/processes/{disposalProcessId}/included_packages/{packageId}/dispose")
@Consumes({ "*/*" })
/*
  @Operation(summary = "Execute final disposal operation (shred or tag with final state)", tags={ "executeDisposal" })
  @ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "successful operation"),
    @ApiResponse(responseCode = "400", description = "Not found"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })*/
public void executePackageDisposalAction(@PathParam("disposalProcessId") String disposalProcessId, @PathParam("packageId") String packageId)
```

4.1.2. Get package metadata

Get metadata content, including child packages metadata

```
@GET
@Path("/packages/{packageId}/metadata")
@Produces({ "application/zip" })
/*
  @Operation(summary = "Get package metadata", tags={ "executeDisposal" })
  @ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schema(implementation = File.class))),
    @ApiResponse(responseCode = "400", description = "Not found"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })
*/
public File getPackageMetadata(@PathParam("packageId") String packageId);
```

4.1.3. Get package metadata and components

Package content

```
@GET
@Path("/packages/{packageId}/content")
@Produces({ "application/zip" })
/* @Operation(summary = "Get package metadata and components", tags={ "executeDisposal" })
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schema)),
    @ApiResponse(responseCode = "400", description = "Not found"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })
*/
public File getPackageMetadataAndContentIncludingChildPackagesDataAndComponents(@PathParam("packageId") String packageId) {
}
```

4.1.4. Get codelist and classification

Get codelist and classification

```
@GET
@Path("/archives/codelist/{name}")
@Produces({ "application/json" })
/* @Operation(summary = "Get codelist and classification", tags={ "manageArchives" })
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schema)),
    @ApiResponse(responseCode = "400", description = "Not found"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })
*/
public CodeList getCodeList(@PathParam("name") String name);
```

4.1.5. Get user credentials

Get user credentials

```
@GET
@Path("/archives/user")
@Produces({ "application/json" })
/* @Operation(summary = "Get user credentials", tags={ "manageArchives" })
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schema)),
    @ApiResponse(responseCode = "400", description = "Not found"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })
*/
public User getUserCredentials(@QueryParam("login") @NotNull String login, @QueryParam("password") @NotNull String password);
```

4.1.6. Put document into archive system

Put document into archive system

```
@POST
@Path("/archives/document")
@Consumes({ "multipart/form-data" })
@Produces({ "application/json" })
// @Operation(summary = "Put document into archive system", tags={ "manageArchives" })
// @ApiResponses(value = {
//     @ApiResponse(responseCode = "200", description = "uid attached file version", content = @Content(schema = @Schema)),
//     @ApiResponse(responseCode = "400", description = "Not found"),
//     @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
//     @ApiResponse(responseCode = "404", description = "Invalid data") })
//
public String putDocument(@Multipart(value = "pdfDocument") File pdfDocument, @Multipart(value = "fileMetadata") File fileMetadata);
```

4.1.7. Run disposal process action.

Run sync. or async. action of the whole disposal process at RMS side. Action may initiate change of process state. In case of long running job (database updates, integrity check) the action is asynchronous and returning action state RUNNING. For synchronous actions FINISHED state is returned immediately.

```
@POST
@Path("/processes/{disposalProcessId}/actions")
@Consumes({ "application/json" })
@Produces({ "application/json" })
/*
  @Operation(summary = "Run disposal process action.", tags={ "manageProcesses", "preparePackages", "executeDisposalProcess" })
  @ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schema(implementation = DisposalProcessActionProgress.class))),
    @ApiResponse(responseCode = "400", description = "Invalid data"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })
*/
public DisposalProcessActionProgress changeDisposalProcessState(@PathParam("disposalProcessId") String disposalProcessId, @Body DisposalProcessActionRequest request) {
    // ...
}
```

4.1.8. Create disposal process instance

Create disposal process with parameters in input

```
@POST
@Path("/processes")
@Consumes({ "application/json" })
@Produces({ "application/json" })
/*
  @Operation(summary = "Create disposal process instance", tags={ "manageProcesses" })
  @ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schema(implementation = DisposalProcessState.class))),
    @ApiResponse(responseCode = "400", description = "Invalid data"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })
*/
public DisposalProcessState createDisposalProcess(@QueryParam("organisation") @NotNull String organisation, @Body DisposalProcessRequest request) {
    // ...
}
```

4.1.9. Get disposal process detail

Get disposal process detail

```
@GET
@Path("/processes/{disposalProcessId}")
@Produces({ "application/json" })
/*
  @Operation(summary = "Get disposal process detail", tags={ "manageProcesses" })
  @ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schema(implementation = DisposalProcess.class))),
    @ApiResponse(responseCode = "400", description = "Invalid data"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })
*/
public DisposalProcess getDisposalProcess(@PathParam("disposalProcessId") String disposalProcessId) {
    // ...
}
```

4.1.10. Get list of disposal processes

Get list of disposal processes

```
@GET
@Path("/processes")
@Produces({ "application/json" })
/*
  @Operation(summary = "Get list of disposal processes", tags={ "manageProcesses" })
  @ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schema(implementation = DisposalProcessList.class))),
    @ApiResponse(responseCode = "400", description = "Invalid data"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })
*/
public DisposalProcessList getDisposalProcesses() {
    // ...
}
```

```
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(array = @ArrayS
    @ApiResponse(responseCode = "400", description = "Invalid data"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })
*/
public List<DisposalProcess> getDisposalProcesses(@QueryParam("organisation") @NotNull String organisation);
```

4.1.11. Get disposal process action progress

Get disposal process action progress or result. Returns any of finished jobs in history.

```
@GET
@Path("/processes/{disposalProcessId}/actions/{processActionId}")
@Produces({ "application/json" })
/*
    @Operation(summary = "Get disposal process action progress", tags={ "manageProcesses" })
    @ApiResponses(value = {
        @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schem
        @ApiResponse(responseCode = "400", description = "Invalid data"),
        @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
        @ApiResponse(responseCode = "404", description = "Invalid data") })
*/
public DisposalProcessActionProgress processesDisposalProcessIdActionsProcessActionIdGet(@PathParam("disposalPr
```

4.1.12. Get current state of disposal process

Get current state of disposal process like "canceled".

```
@GET
@Path("/processes/{disposalProcessId}/state")
@Produces({ "application/json" })
/*
    @Operation(summary = "Get current state of disposal process", tags={ "manageProcesses" })
    @ApiResponses(value = {
        @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schem
        @ApiResponse(responseCode = "400", description = "Invalid data"),
        @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
        @ApiResponse(responseCode = "404", description = "Invalid data") })
*/
public DisposalProcessState processesDisposalProcessIdStateGet(@PathParam("disposalProcessId") String disposalP
```

4.1.13. Disposal process candidate

Vyhledání kandidátů skartačního řízení

```
@GET
@Path("/processes/{disposalProcessId}/candidates")
@Produces({ "application/json" })
/*
    @Operation(summary = "Disposal process candidates", tags={ "preparePackages" })
    @ApiResponses(value = {
        @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schem
        @ApiResponse(responseCode = "400", description = "Not found"),
        @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
        @ApiResponse(responseCode = "404", description = "Invalid data") })
*/
public PackagesList listCandidates(@PathParam("disposalProcessId") String disposalProcessId, @QueryParam("packa
```

4.1.14. Register packages into disposal process

Register packages into disposal process. Packages must be part of candidates list. Packages are locked for disposal operation by the records management system.


```
@POST
@Path("/processes/{disposalProcessId}/included_packages")
@Consumes({ "application/json" })
@Produces({ "application/json" })
/*
  @Operation(summary = "Register packages into disposal process", tags={ "preparePackages", "reviewPackages" })
  @ApiResponses(value = {
    @ApiResponse(responseCode = "207", description = "multiple status response", content = @Content(array = @Array
    @ApiResponse(responseCode = "400", description = "Not found"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })
  */
public List<StatusResponse> registerPackages(@PathParam("disposalProcessId") String disposalProcessId, @Valid L
```

4.1.15. Remove packages from disposal process

Remove selected packages from disposal process

```
@DELETE
@Path("/processes/{disposalProcessId}/included_packages")
@Consumes({ "*/*" })
@Produces({ "application/json" })
/*
  @Operation(summary = "Remove packages from disposal process", tags={ "reviewPackages" })
  @ApiResponses(value = {
    @ApiResponse(responseCode = "207", description = "multi-status response", content = @Content(array = @ArrayS
    @ApiResponse(responseCode = "400", description = "Not found"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })
  */
public List<StatusResponse> disposalProcessRemovePackage(@PathParam("disposalProcessId") String disposalProcess
```

4.1.16. Disposal process registered packages

Seznam balíčků ve skartačním řízení

```
@GET
@Path("/processes/{disposalProcessId}/included_packages")
@Produces({ "application/json" })
/*
  @Operation(summary = "Disposal process registered packages", tags={ "reviewPackages" })
  @ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schem
    @ApiResponse(responseCode = "400", description = "Not found"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })
  */
public PackagesList listRegisteredPackages(@PathParam("disposalProcessId") String disposalProcessId, @QueryPar
```

4.1.17. Change parameters of packages in disposal process

Notify change of package disposal action, or delay disposal with new period

```
@PATCH
@Path("/processes/{disposalProcessId}/included_packages")
@Consumes({ "*/*" })
@Produces({ "application/json" })
/*
  @Operation(summary = "Change parameters of packages in disposal process", tags={ "updatePackages" })
  @ApiResponses(value = {
    @ApiResponse(responseCode = "207", description = "multi-status response"),
    @ApiResponse(responseCode = "400", description = "Not found"),
    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid"),
    @ApiResponse(responseCode = "404", description = "Invalid data") })
  */
public List<StatusResponse> changePackageState(@PathParam("disposalProcessId") String disposalProcessId, @Valid
```

4.2. manage document REST API

Record manage document REST API

```
@Path("/")
public interface ManageDocumentsApi
```

4.2.1. Get document content

Get document content

```
@GET
@Path("/document/{uid}/content")
@Produces({ "application/pdf" })
// @Operation(summary = "Get document content", tags={ "manageDocuments" })
// @ApiResponses(value = {
//     @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schema(
//     @ApiResponse(responseCode = "404", description = "Not found"),
//     @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid") })
public File getDocumentContent(@PathParam("uid") String uid);
```

4.2.2. Get document metadata

Get document metadata

```
@GET
@Path("/document/{uid}/metadata")
@Produces({ "application/json" })
// @Operation(summary = "Get document metadata", tags={ "manageDocuments" })
// @ApiResponses(value = {
//     @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schema(
//     @ApiResponse(responseCode = "404", description = "Not found"),
//     @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid") })
public Document getDocumentMetadata(@PathParam("uid") String uid);
```

4.2.3. Save document with metadata

Save document with metadata

```
@POST
@Path("/document")
@Consumes({ "multipart/form-data" })
@Produces({ "application/json" })
// @Operation(summary = "Save document with metadata", tags={ "manageDocuments" })
// @ApiResponses(value = {
//     @ApiResponse(responseCode = "200", description = "uid attached file version", content = @Content(schema = @Schema(
//     @ApiResponse(responseCode = "404", description = "Not found"),
//     @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid") })
public String putDocumentAndMetadata(@Multipart(value = "documentContent") File documentContent, @Multipart(value = "metadata")
```

4.2.4. Deprecate document

Deprecate document

```
@POST
@Path("/document/{uid}/deprecate")
@Consumes({ "*/*" })
// @Operation(summary = "Deprecate document", tags={ "manageDocuments" })
```

```
//@ApiResponseResponses(value = {  
//    @ApiResponse(responseCode = "200", description = "successful operation"),  
//    @ApiResponse(responseCode = "404", description = "Not found"),  
//    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid") })  
public void deprecateDocument(@PathParam("uid") String uid);
```

4.2.5. Save NSESSS document

Save NSESSS document

```
@POST  
@Path("/nsesss")  
@Consumes({ "multipart/form-data" })  
@Produces({ "application/json" })  
//@Operation(summary = "Save NSESSS document", tags={ "manageDocuments" })  
//@ApiResponseResponses(value = {  
//    @ApiResponse(responseCode = "200", description = "uid attached file",  
//    @ApiResponse(responseCode = "404", description = "Not found"),  
//    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid") })  
public String putDocumentNSESSS(@Multipart(value = "documentNSESSS") File documentNSESSS);
```

4.2.6. Get NSESSS document

Get NSESSS document

```
@GET  
@Path("/nsesss/{uid}")  
@Produces({ "application/zip" })  
//@Operation(summary = "Get document content", tags={ "manageDocuments" })  
//@ApiResponseResponses(value = {  
//    @ApiResponse(responseCode = "200", description = "successful operation",  
//    @ApiResponse(responseCode = "404", description = "Not found"),  
//    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid") })  
public File getDocumentNSESSS(@PathParam("uid") String uid);
```

4.2.7. Save ISSD document

Save ISSD document

```
@POST  
@Path("/issd")  
@Consumes({ "multipart/form-data" })  
@Produces({ "application/json" })  
//@Operation(summary = "Save ISSD document", tags={ "manageDocuments" })  
//@ApiResponseResponses(value = {  
//    @ApiResponse(responseCode = "200", description = "uid attached file",  
//    @ApiResponse(responseCode = "404", description = "Not found"),  
//    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid") })  
public String putDocumentISSD(@Multipart(value = "documentISSD") File documentISSD  
    , @QueryParam("rc") String rc  
    , @QueryParam("ident") String ident  
    , @QueryParam("action") String action  
    , @QueryParam("period") Integer period  
    , @QueryParam("szacr") String szacr);
```

4.2.8. Get document STATUS

Get document STATUS

```
@GET
```

```
@Path("/status/{id}")
@Produces({ "application/json" })
//@Operation(summary = "Get document status", tags={ "manageDocuments" })
//@ApiResponse(value = {
//    @ApiResponse(responseCode = "200", description = "successful operation",
//    @ApiResponse(responseCode = "404", description = "Dokument podle ID nebyl nalezen"),
//    @ApiResponse(responseCode = "400", description = "Bad request, id is missing."),
//    @ApiResponse(responseCode = "401", description = "Authentication information is missing or invalid") })
public String getDocumentStatus(@PathParam("id") String id);
```

4.2.9. Get document metadata by producer and document ident

```
@GET
@Path("/document/metadata")
@Produces({ "application/json" })
//@Operation(summary = "Get document metadata", tags={ "manageDocuments" })
//@ApiResponse(value = { //    @ApiResponse(responseCode = "200", description = "successful operation"), //    @ApiR
// return document metadata
public Document getDocumentMetadata(
    @QueryParam("producer") String producer
, @QueryParam("ident") String ident);
```

4.2.10. Get document metadata collection by document id and producer collection

```
@POST
@Path("/document/metadata/identifiers")
@Consumes({ "application/json" })
@Produces({ "application/json" })
//@Operation(summary = "Get document metadata collection", tags={ "manageDocuments" })
//@ApiResponse(value = { //    @ApiResponse(responseCode = "200", description = "successful operation"), //    @ApiR
)
// return list document metadata
public List<Document> getDocumentMetadata(@Valid List<ProducerIdentifier> identifiers);
}
```

4.2.11. Deprecate document by producer and document ident

```
@GET
    @Path("/document/deprecate")
    @Consumes({ "/" })
//@Operation(summary = "Deprecate document", tags={ "manageDocuments" })
//@ApiResponse(value =
{ //    @ApiResponse(responseCode = "200", description = "successful operation"), //    @ApiResponse(responseCode =
)
    public void deprecateDocument(
        @QueryParam("producer") String producer
, @QueryParam("ident") String ident);
```

4.2.12. Deprecate documents by collection of document id and producer

```
@POST
    @Path("/documents/deprecate")
    @Consumes({ "/" })
//@Operation(summary = "Deprecate document", tags={ "manageDocuments" })
```

```
//@ApiResponse(value =
{ //      @ApiResponse(responseCode = "200", description = "successful operation"), //      @ApiResponse(responseCode = "200", description = "successful operation")
})
// return HTTP status code
public void deprecateDocuments (@Valid List<ProducerIdentifier> identifiers);
```

4.2.13. Get document content by producer and document ident

```
@GET
@Path("/document/content")
//@Produces({ "plain/text" })
//@Operation(summary = "Get document content", tags={ "manageDocuments" })
//@ApiResponse(value =
{ //      @ApiResponse(responseCode = "200", description = "successful operation"), //      @ApiResponse(responseCode = "200", description = "successful operation")
})
// return pdf in base64
public byte[] getDocumentContent(
    @QueryParam("producer") String producer
    , @QueryParam("ident") String ident);
```

4.2.14. Get document identifiers by document producer and patient namespace and identifier

```
@GET
@Path("/document/identifiers")
//@ApiResponse(value =
{ //      @ApiResponse(responseCode = "200", description = "Successful operation"), //      @ApiResponse(responseCode = "200", description = "Successful operation")
})
// return json of document's identifier, origin(datetimeTolong) and valid(true|false)
public String getDocumentIdentifiersByPatient(
    @QueryParam("producer") String producer
    , @QueryParam("namespace") String nspace
    , @QueryParam("identifier") String ident);
```

Příloha A. Příloha: Příklad SOAP klienta v PHP

A.1. Instalace a konfigurace (CentOS Linux 5.x)

Pro vytvoření PHP SOAP klienta je nutné mít nainstalované následující balíky:

- php
- php-soap

a v `/etc/php.ini` mít povoleno rozšíření pro SOAP:

```
; Enable soap extension module
extension=soap.so
```

A.2. Vytvoření klienta

A.2.1. WS-Security

Balík `php-soap` nemá přímou podporu pro WS-Security. Tuto podporu implementuje následující třída `WSSoapClient`, kterou na webu www.phpclasses.org publikoval Roger Veciana a dal k dispozici pod BSD licenci. Tato třída bude využita v příkladech níže.

```
<?php
class WSSoapClient extends SoapClient {

    private $username;
    private $password;
    /*Generates de WSSecurity header*/
    private function wssecurity_header() {

        /* The timestamp. The computer must be on time or the server you are
        * connecting may reject the password digest for security.
        */
        $timestamp = gmdate('Y-m-d\TH:i:s\Z');
        /* A random word. The use of rand() may repeat the word if the server is
        * very loaded.
        */
        $nonce = mt_rand();
        /* This is the right way to create the password digest. Using the
        * password directly may work also, but it's not secure to transmit it
        * without encryption. And anyway, at least with axis+wss4j, the nonce
        * and timestamp are mandatory anyway.
        */
        $passdigest = base64_encode(
            pack('H*',
                sha1(
                    pack('H*', $nonce) . pack('a*', $timestamp) .
                    pack('a*', $this->password))));

        $auth = '
<wsse:Security SOAP-ENV:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.'.
'org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
<wsse:UsernameToken>
    <wsse:Username>' . $this->username . '</wsse:Username>
```

```

    <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-'.
'wss-username-token-profile-1.0#PasswordText">'. $this->password.' </wsse:Password>
    <wsu:Created xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-'.
'200401-wss-wssecurity-utility-1.0.xsd">'. $timestamp.' </wsu:Created>
    </wsse:UsernameToken>
</wsse:Security>
';

    /* XSD_ANYXML (or 147) is the code to add xml directly into a SoapVar.
    * Using other codes such as SOAP_ENC, it's really difficult to set the
    * correct namespace for the variables, so the axis server rejects the
    * xml.
    */
    $authvalues = new SoapVar($auth,XSD_ANYXML);
    $header = new SoapHeader("http://docs.oasis-open.org/wss/2004/01/oasis-".
        "200401-wss-wssecurity-secext-1.0.xsd", "Security", $authvalues,
        true);

    return $header;
}

/* It's necessary to call it if you want to set a different user and
 * password
 */
public function __setUsernameToken($username, $password) {
    $this->username = $username;
    $this->password = $password;
}

/* Overwrites the original method adding the security header. As you can
 * see, if you want to add more headers, the method needs to be modified
 */
public function __soapCall($function_name, $arguments, $options=null,
    $input_headers=null, $output_headers=null) {

    $result = parent::__soapCall($function_name, $arguments, $options,
        $this->wssecurity_header());
    return $result;
}
}
?>

```

A.2.2. Příklady

Předpokládejme, že v APFS je vytvořená SOAP aplikační entita, která má login "admin" a heslo "admin".

A.2.2.1. Úspěšná autentizace

```

<?php
include('WSSoapClient.php');

#WS-ecurity login, password
$username = 'admin';
$password = 'admin';

#instance klienta
try {
    $client = new WSSoapClient('http://192.168.242.8:9090/geniface?wsdl');
} catch (Exception $exception){
    var_dump($exception);
    die;
}

```

```
#nastav login a heslo instanci
$client->__setUsernameToken($username, $password);

#volani metody getVersion() + pripadne odchyceni vyjimky
try {
    $result = $client->__soapCall('getVersion',array(),null,null,null);
    printf($result->return . "\n");
} catch (SoapFault $exception) {
    var_dump($exception);
    die;
}
?>
```

A.2.2.2. Odchycení a zpracování vyjímky

```
<?php
include('WSSoapClient.php');

#WS-ecurity login, password
$username = 'admin';
$password = 'admin';

#instance klienta
try {
    $client = new WSSoapClient('http://192.168.242.8:9090/testiface?wsdl');
} catch (Exception $exception){
    var_dump($exception);
    die;
}

#volani metody throwExceptionWithText + odchyceni vyjimky
try {
    $result = $client->__soapCall('throwExceptionWithText',
                                array('parameters'=>array('text'=>"hello"),
                                null,null,null);

    echo($result->return);
} catch (SoapFault $exception) {
    var_dump($exception);
}
?>
```

A.2.2.3. Průchod hierarchickou strukturou

```
<?php
include('WSSoapClient.php');

#WS-ecurity login, password
$username = 'admin';
$password = 'admin';

#instance klienta
try {
    $client = new WSSoapClient('http://192.168.242.8:9090/geniface?wsdl');
    $client->__setUsernameToken($username, $password);
} catch (Exception $exception){
    var_dump($exception);
    die;
}

# informace o pacientovi
$sp = array( 'lastName' => '*S*',
    'birthDate' => '19000101-19801231',
    'identifiers' => array('namespace' => 'RC', 'id' => '*0*')
```



```
);
$s = array('accessionNumber' => '*');

try {
    #findStudies()
    $result = $client->__soapCall('findStudies',
        array('parameters' => array('patientPattern' => $p,
                                    'studyPattern' => $s)),
        null, null, null);
    foreach ($result->return as $uid) {
        printf("SeriesUid: " . $uid . "\n");
    }
    printf("OK\n");

    #getStudyInfo()
    $studUid = "1.3.46.670589.11.0.0.11.4.2.0.18006.5.67464.2006053106571578000";
    $result = $client->__soapCall('getStudyInfo',
        array('parameters' => array('uid' => $studUid)),
        null, null, null);
    printf("PatientUID: " . $result->return->patientUid);
    printf("\nOK\n");

    #getPatientInfo()
    $pacUid = "1.3.6.1.4.1.20744.3.1.2.2.3100.1277986117668.2246";
    $result = $client->__soapCall('getPatientInfo',
        array('parameters' => array('uid' => $pacUid)),
        null, null, null);

    var_dump($result);
    printf("\nOK\n");

    #getSeriesInfo()
    $result = $client->__soapCall('getSeriesInfo',
        array('parameters' => array('uid' => $studUid)),
        null, null, null);

    var_dump($result);
    printf("\nOK\n");

    #getImageInfo
    $imgUid = "1.2.392.200036.9125.0.19950720105640";
    $result = $client->__soapCall('getImageInfo',
        array('parameters' => array('uid' => $studUid)),
        null, null, null);

    var_dump($result);
    printf("\nOK\n");
} catch (SoapFault $exception) {
    var_dump($exception);
}
?>
```