TANDLÆGEATTESTER

Bilag 1
Rest Api

Version 1.0 Revision 2

Dokumentoplysninger

Titel:	Guide for Tandlægeattester, Bilag 1 Rest Api
Projekt:	EDI kontorets branchekoordinerede dataudveksling
Forfatter:	Morten Lassen, F&P IT-afdelingen
Bidragsydere til dokumentet:	
Godkendt af:	Martin Petersen, F&P EDI-kontoret
Dokumentansvarlig:	Martin Petersen, F&P EDI-kontoret
Fordeling:	EDI kontoret, Forsikring & Pension Udleveres til interessenter i dataudvekslingen
Bemærkning:	Dokumentet kan rekvireres hos Forsikring & Pension

Ændringslog

Version	Dato	Forfatter	Ændrede sider eller afsnit
1.0 Draft A	19.04.2021	MLA	Første udkast
1.0 Draft B	06.05.2021	MLA	Kodeeksempler tilføjet
1.0	04.10.2021	MLA	Første udgave
1.0 Revision 1	24.06.2022	MPE	Afsnit 7 opdateret med koder og tekster fra Web EDI
1.0 Revision 2	30.06.2022	MLA	Manglende InvoiceId i operation GET Invoice Præcisering af GET attachment

Forkortelser og definitioner:	
https	En krypteret udgave af http som benyttes til datakommunikation over internettet
Api	Applikations Program Interface
Rest	Representational State Transfer
Rest Api	Rest baseret Program interface, der benytter http forespørgsler til dataudveksling vha. GET, PUT, POST og DELETE
Token	En dynamisk nøgle/adgangsbillet der identificerer afsenderen
OAuth2	En åben standard for at give adgang til data
JSON	JavaScript Object Notation. Et tekstbaseret format til dataudveksling
Web-Patient	Portal til underskrivning med digital signatur
SYNLAB	Samarbejdspartner der håndterer Web-Patient
Nasure	Samarbejdspartner der kommunikerer med tandklinikkerne

Referencer:
Vejledning for Api-administrator DateTime datatyper følger ISO 8601 standarden `CCYY-MM-DDThh:mm:ss[Z|(+|-)hh:mm]`. For eksempel `2016-12-31T20:00:01`. Se https://en.wikipedia.org/wiki/ISO_8601

Datoer uden værdi angives med værdien null

Beskrivelse af Rest Api
https://en.wikipedia.org/wiki/Overview_of_RESTful_API_Description_Languages

Indholdsfortegnelse:

1. Indledning

Dette dokument beskriver selskabernes integration til løsningen tandlægeattester.
Oprettelse og afsendelse af anmodningen sker via indtastning på Web/EDI-serveren. Den resterende del af flowet herunder hentning af dokumenter er understøttet i integrationen, der er baseret på Rest standarden.

Til at sikre Rest Api kaldet benyttes en standardiseret OAuth2 arkitektur, som udsteder en Access Token også kaldet en "adgangsbillet". Denne access token giver selskabet adgang til at kalde tandlægeattest Api'et.

Rest Api returner en http status 200/OK, hvis operationen gik godt. Returdata sendes i http body. Ved eventuelle fejl returneres en standard http statuskode.

Beskrivelsen af access token og sikkerheden generelt fremgår af kapitel 2.

2. Sikkerhed

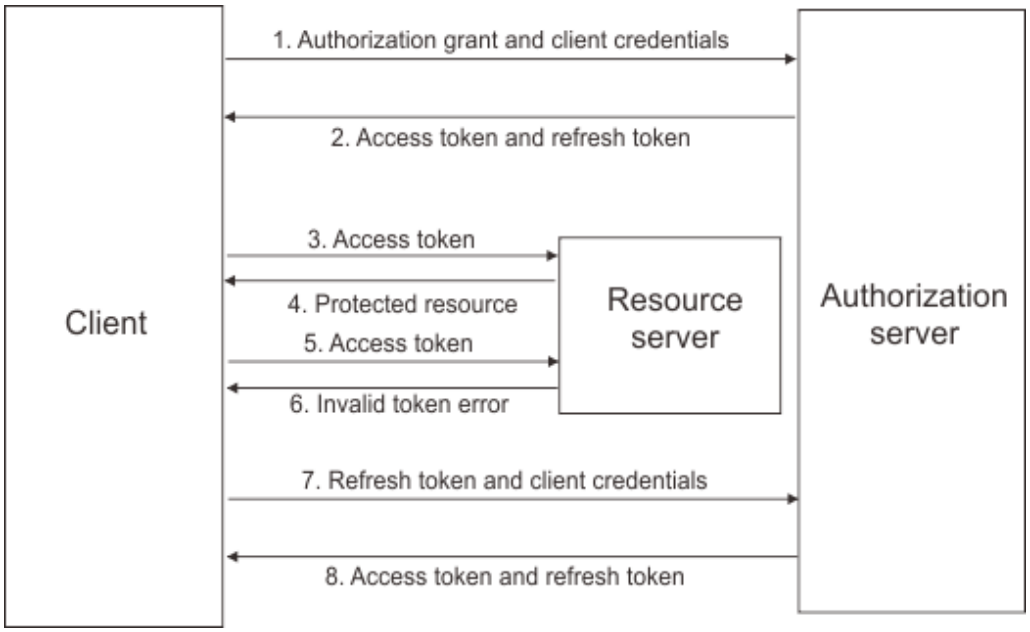
Forsikring & Pensions Rest Api er beskyttet med flg. teknologier:

- https / SSL / TLS der sikrer en krypteret dataforbindelse
- ip whitelist der forhindrer uautoriseret adgang
- Access token der identificerer selskabet

For at hente en Access token skal der bruges en klient-id (`ClientId`) og en hemmelig nøgle (`ClientSecret`). Klient-id er unikt for selskabet og tildeles af Forsikring & Pension. Den hemmelig nøgle skal selskabets Api-administrator selv danne. Se separat vejledning for Api-administrator.

Autorisation

Access Token benytter OAuth2 standarden og flowet er vist herunder.



Access token
Før selskabet kan kalde en operation på Forsikring & Pensions Rest Api, skal det anmode om en access token. Denne token udstedes af EDI autorisationsserveren på baggrund af en klient-id (`ClientId`) samt en klient nøgle (`ClientSecret`). Sammen med access token returneres også en refresh token, som kan benyttes til at hente en ny access token, når denne er udløbet. Det er ikke et krav at refresh token skal benyttes.

Access token har en levetid på 20 minutter, og skal angives i alle efterfølgende kald til serverens Rest Api.

Access tokens har sin egen livscyclus og der kan derfor være udstedt flere aktive tokens til samme selskab, hvis selskabet har kaldt flere gange samtidigt.

ClientId og ClientSecret
ClientId er unikt for selskabet og genereres af EDI-serveren. ClientId er en 32 karakterer hex streng, og kan ses i selskabsadministrationen.

ClientSecret er en hemmelig kryptografisk genereret streng, som kun er kendt af selskabet og autorisationsserveren. Da autorisationsserveren gemmer hash værdien af denne streng, er det ikke muligt at få oplyst ClientSecret efter den er genereret. Selskabet kan selv generere en ny ClientSecret. ClientSecret er valid i 365 dage, og skal skiftes af selskabet inden udløb.
Der sendes automatisk en reminder til Api Administratoren 30 dage før udløb.

Anmodning om en token sker på adressen:

Test: https://testedi.forsikringogpension.dk/authserver/oauth/token
Produktion: https://edi.forsikringogpension.dk/authserver/oauth/token

Flg. oplysninger skal angives:

Grant_type skal have værdien `client_credentials`
Scope skal have værdien `tlattest`
Client_id og Client_secret

Eksempel på anmodning om en access token vha. programmet Curl:

```
curl -X POST
-u "30feadba6f234a4986c04b1c7fae092b:RzLsDZlmwvyoWE/+BPApwCKsC0aoh1rvM+6IjT7mpkQ=" -d
"grant_type=client_credentials&scope=tlattest"
https://testedi.forsikringogpension.dk/authserver/oauth/token
```

Ved korrekt klient id og -nøgle returnerer serveren en access token og en refresh token.

Eksempel på svar:

```
< HTTP/1.1 200 OK
< Cache-Control: no-cache
< Pragma: no-cache
< Content-Type: application/json;charset=UTF-8
< Expires: -1
< Server: Microsoft-IIS/10.0
```

```
< X-Powered-By: ASP.NET
< Date: Mon, 19 Apr 2021 09:43:16 GMT
< Content-Length: 858
<
{
 "access_token":Vv7gPtgWE4YQ6JqgweOv5k_OFouRsT6Y7J3BKgxClTotPCFSbSUrdT7ieqBN
0vuc1jtgZDk1m1FI7qwn5I4Kh_AcVtOnvJYPZFDViMHjUf07R16cLxPe2VPrc_peGsXQbFncjtd6S
lHEGNw7Zo6eeNkQY1t2ga97PakAWmwAiXOdnjnUsxcbDoG_W8bnQBNEkPEGetXHtvQ9V21hLoxeoL4
UQ6pe3XyrfruAxEvIZ0wKAtdWTSaebWXHnudYJwtXoAA9IHPfSGJWHdD92vIgljxrNEj7Ln6ycW7ICrR
IUk2FxpiK6woPIt4szcCdic1_nhVYnySzdjBM7wHeRv-2cV4-IIvz_6u4i7od_zxjh6o1m1pfGoy9-
 e-5vtXOQUA",
 "token_type": "bearer",
 "expires_in": 1200,
 "refresh_token": "x-GAzugdkRSQUh9akGI1ROYEe9Is0uAE7URgd2P6ezigovhbccpGNggyHA4r
IO_H9aA3Ue7JqHpakl4GtaQbK2RfWEM8kdeUCIMQ6am4p5ASJl8ktxL81-qq7iIGQ1b4pj9sDu5Ddov7Vv
6sZbVbIkJmyx2JfaLuvVElujIrMLE8AGdtFFl340fr9-87r-R7sZaBrXi1qp0AVxTlRLPGLA9A0On5hwlQ
6-qXangZAP6tq_sIbhEp5VnAIbhfPrHMA6h81NKa-MwAs2fxl4oRtjp7W1N6oNPrsh-zlsG82HHq3yfbY5
B07WYLVJBteWQsa525s1CHhkVqofURm57i60tfcKdYOReW0ZOrnyStmcyHzWC_D_q33argKWfwxE33Q"
}
```

Access token (`access_token`) skal angives i http headeren ved alle efterfølgende kald
`Authorization: Bearer <token>`

Access token udløber efter en kort tidsperiode som er bestemt af udstederen. I ovenstående eksempel udløber (`expires_in`) den efter 1200 sekunder hvilket svarer til 20 minutter.

Eksempel på kald af operationen status:

```
curl -X GET "https://testedi.forsikringogpension.dk/api/v1/TLAttestService/status" -H
"accept: */*"
-H  "Authorization: Bearer 8yyy2M1crVXkTD6fOmYWyDkyrOOxRml1Scw9ly-
755j3aYt5jh5MRKsas97ohmiyt9vi3Dfh87OP9sF_LIjQNyxFtBc1wlMQ7OEHcsMSoTv-
8FwYo4jqXuQbkEHm1uOTdK9yj6op2vU2WZ8L6Ii5NkHOpafTHC7I0Z53n3hEDt8pVLr8yJtxAb1EQybxtQ48yS
xqgoWGBdirVnOONdzKWm-L20aBDEB8CRDkO-_H-sX_8m65gzcP3q9FWoYnqa-eS4j-J2Kcu4OtYTTZET-
Q_Pl5H2poOwlBoq25a8CQnVFJ1YH3gkytacromrfQWVfwqQid2VF1BQ3a1-Zqp4XO0H6kA-0"
```

Eksempel på svar ved korrekt angivet token:

```
< HTTP/1.1 200 OK
< Transfer-Encoding: chunked
< Content-Type: text/plain; charset=utf-8
< Server: Microsoft-IIS/10.0
< X-Powered-By: ASP.NET
< Date: Mon, 19 Apr 2021 09:45:23 GMT
<
Hello Pensionsselskab 1 (test)
```

Eksempel på svar ved udløbet eller forkert token:

```
< HTTP/1.1 401 Unauthorized
< Transfer-Encoding: chunked
< Server: Microsoft-IIS/10.0
< X-Powered-By: ASP.NET
< Date: Mon, 19 Apr 2021 10:26:37 GMT
<
```

Refresh token

Refresh token har en levetid på 48 timer og fornyes hver gang der hentes en ny access token.
Når access token er udløbet kan selskabet hente en ny access token ved at angive refresh token i requestet. Bemærk – det er ikke et krav at selskaberne skal anvende refresh token til at hente ny token.

Flg. oplysninger skal angives som POST parametre:

Grant_type skal være `refresh_token`
Client_id er den samme som ved anmodning om en access token
Refresh_token skal have værdien for refresh_token som blev returneret ved anmodning om en access token.

Eksempel på anmodning om en token:

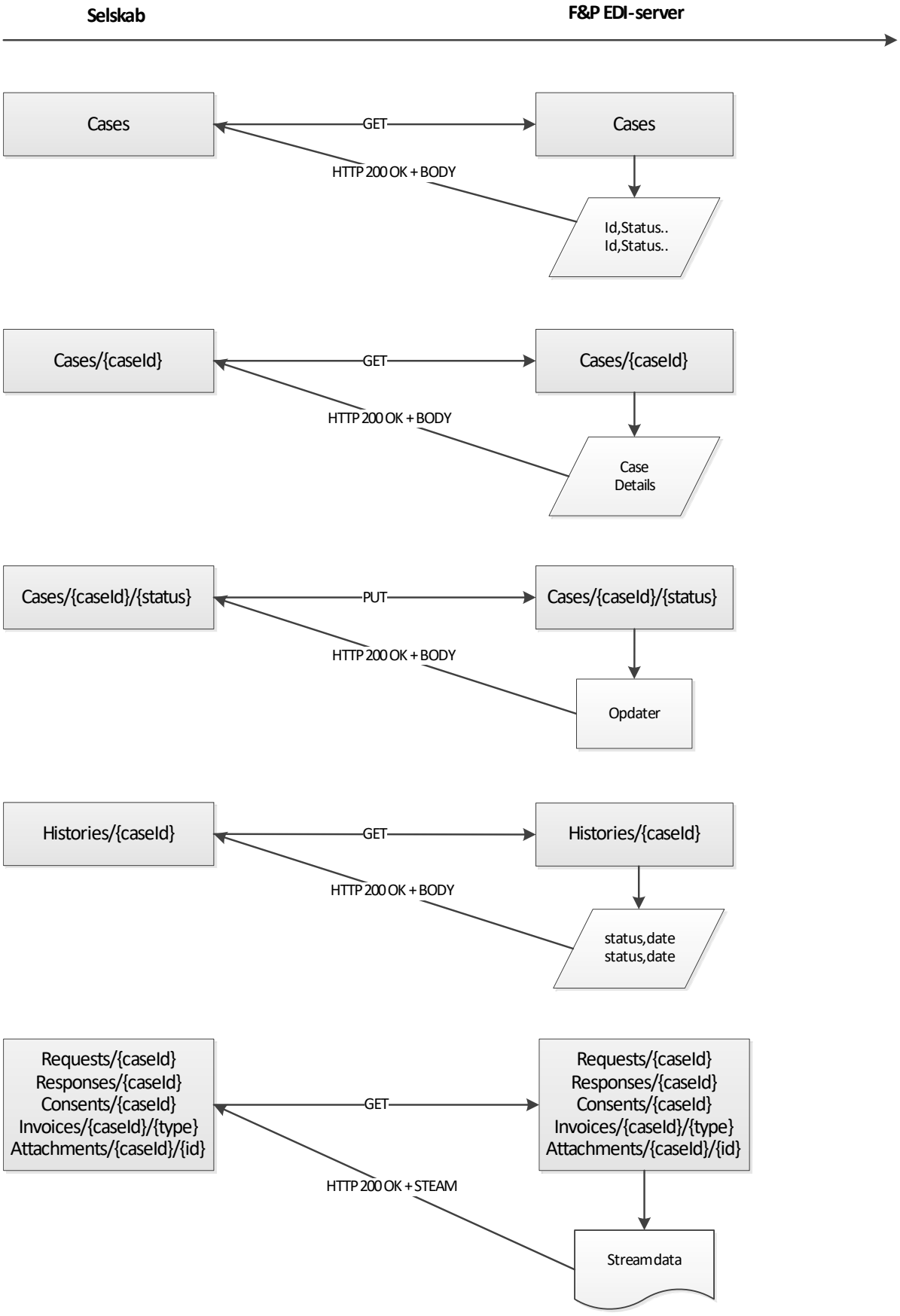
```
POST /authserver/oauth/token HTTP/1.1
Host: testedi.forsikringogpension.dk
grant_type=refresh_token
&client_id=a18344f95c694272b6203b35a1ffe059
&refresh_token=x-GAzugdkRSQUh9akGI1ROYEe9Is0uAE7URgd2P6ezigovhbccpGNggyHA4r
IO_H9aA3Ue7JqHpakl4GtaQbK2RfWEM8kdeUCIMQ6am4p5ASJl8ktxL81-qq7iIGQ1b4pj9sDu5Ddov7Vv
6sZbVbIkJmyx2JfaLuvVElujIrMLE8AGdtFFl340fr9-87r-R7sZaBrXi1qp0AVxTlRLPGLA9A0On5hwlQ
6-qXangZAP6tq_sIbhEp5VnAIbhfPrHMA6h81NKa-MwAs2fxl4oRtjp7W1N6oNPrsh-zlsG82HHq3yfbY5
B07WYLVJBteWQsa525s1CHhkVqofURm57i60tfcKdYOReW0ZOrnyStmcyHzWC_D_q33argKWfwxE33Q
```

Ved korrekt klient id og refresh token returnerer serveren en ny access token og en ny refresh token. Hvis både access token og refresh token er udløbet, skal der anmodes som en ny access token ved hjælp af grant typen `client_credentials`.

3. Flowdiagram

I det følgende beskrives flow mellem selskab og Forsikring & Pensions EDI-server.

Kommunikation mellem selskaberne og de privatpraktiserende tandlæger sker via Forsikring & Pensions EDI-server og Nasure med SYNLAB som mellemliggende proxy.
EDI-server kommunikerer med SYNLAB, når en sag oprettes. SYNLAB håndterer Web-patient, rykkere og statushåndtering.
EDI-server kommunikerer med Nasure, når svar og bilag skal hentes samt ved søgning på tandklinik.


Selskab
F&P EDI-server
Cases
GET
Cases
HTTP 200 OK + BODY
Id,Status..
Id,Status..
Cases/{caseId}
GET
Cases/{caseId}
HTTP 200 OK + BODY
Case
Details
Cases/{caseId}/{status}
PUT
Cases/{caseId}/{status}
HTTP 200 OK + BODY
Opdater
Histories/{caseId}
GET
Histories/{caseId}
HTTP 200 OK + BODY
status,date
status,date
Requests/{caseId}
Responses/{caseId}
Consents/{caseId}
Invoices/{caseId}/{type}
Attachments/{caseId}/{id}
GET
Requests/{caseId}
Responses/{caseId}
Consents/{caseId}
Invoices/{caseId}/{type}
Attachments/{caseId}/{id}
HTTP 200 OK + STEAM
Stream data

4. Beskrivelse

Selskabet kan benytte operationen **status** til at kontrollere, at forbindelsen samt access token fungerer. Ved korrekt verificering vil operationen returnere selskabets navn. Operatioen benyttes primært i opstartsfasen, hvor kommunikationen skal testes.

Selskabet starter med at kalde operationen **cases**, som returnerer en liste af sager, hvor sagen har skiftet til en status, som kræver selskabets opmærksomhed – se afsnit 7 statuskoder. Listen indeholder en entydig identifikation af sagen samt de dokumenttyper, der er klar til hentning. Endvidere returneres sagens seneste status og statusdato. For at markere, at status er behandlet, skal der sendes en kvittering – se **cases/{caseId}/{status}.**

Herefter kalder selskabet **cases/{caseId}** for hente oplysninger om sagen samt en liste af dokumenter tilknyttet sagen. For at hente et dokument skal respektive operation kaldes.

Operationen **requests/{caseId}/{type}** returnerer anmodningen som json eller PDF. Anmodningen vil være klar umiddelbart efter den er oprettet på Web.

Operationen **consent/{caseId}** returnerer det underskrevet samtykke dokument som PDF.

Såfremt det allerede er underskrevet, så vil den være klar til hentning umiddelbart efter, at det er afsendt. Hvis det skal underskrives i WebPatient, så vil den være klar til hentning umiddelbart efter, at EDI-serveren har modtaget en status 20 – se afsnit 7 statuskoder.

Operationen **responses/{caseId}** returnerer tandklinikkens svar som PDF. Se slettefrister.

Operationen **invoices/{caseId}/{invoiceId}/{type}** returnerer en faktura. Formatet er XML eller PDF og angives i kaldet. Faktura er klar til henting, når serveren har modtaget en status 85.
Der kan være 2 typer af faktura til en sag. Faktura for erklæring og faktura for behabndling.

Operationen **attachments/{caseId}/{attachmentId}** returnerer et vedhæftet bilag.

Operationen **links/{caseId}/{attachmentId**} returnerer et midlertidigt link til et røntgenbillede samt link til en thumbnail. Link er gyldigt i 5 minutter. Der kan hentes et nyt link indtil tandklinikkens svar er slettet. Røntgenbilleder hentes af selskabet, direkte hos Nasure via links, og derved uden om EDI-serveren.

Når selskabet har behandlet sagens nye status fx når et dokument er hentet, skal der kvitteres ved kald af operationen **cases/{caseId}/{status}**, da sagen ellers vil dukke op igen i listen fra Cases. I kaldet kan man angive om visning af meddelelsen på Web skal markeres som ny, læst eller arkiveret.

Sagshistorikken for en sag kan hentes ved at kalde operationen **histories/{caseId}**.

Slettefrister

Svaret fra tandklinikken, vedhæftede filer og adgangen til røntgenbilleder bliver slettet fra serveren efter 14 dage, fra selskabet har hentet klinikkens svar første gang.

Svaret fra tandklinikken, vedhæftede filer og adgangen til røntgenbilleder slettes automatisk efter 3 måneder, hvis svaret ikke bliver hentet. Selskabet vil modtage en påmindelse før sletning. Dette sker via status 90, som sendes 10 dage før sletning. Sagen vil således optræde i Cases.

Oplysninger, som selskabet har tastet ind i anmodningen om kunden helbred, slettes efter 6 måneder.

Hele sagen, inkl. samtykke og faktura, slettes efter 5 år.

5. TLAttestService Api

I det følgende afsnit beskrives Forsikring & Pensions Rest Api for tandlægeattester – benævnt TLAttestService Api.

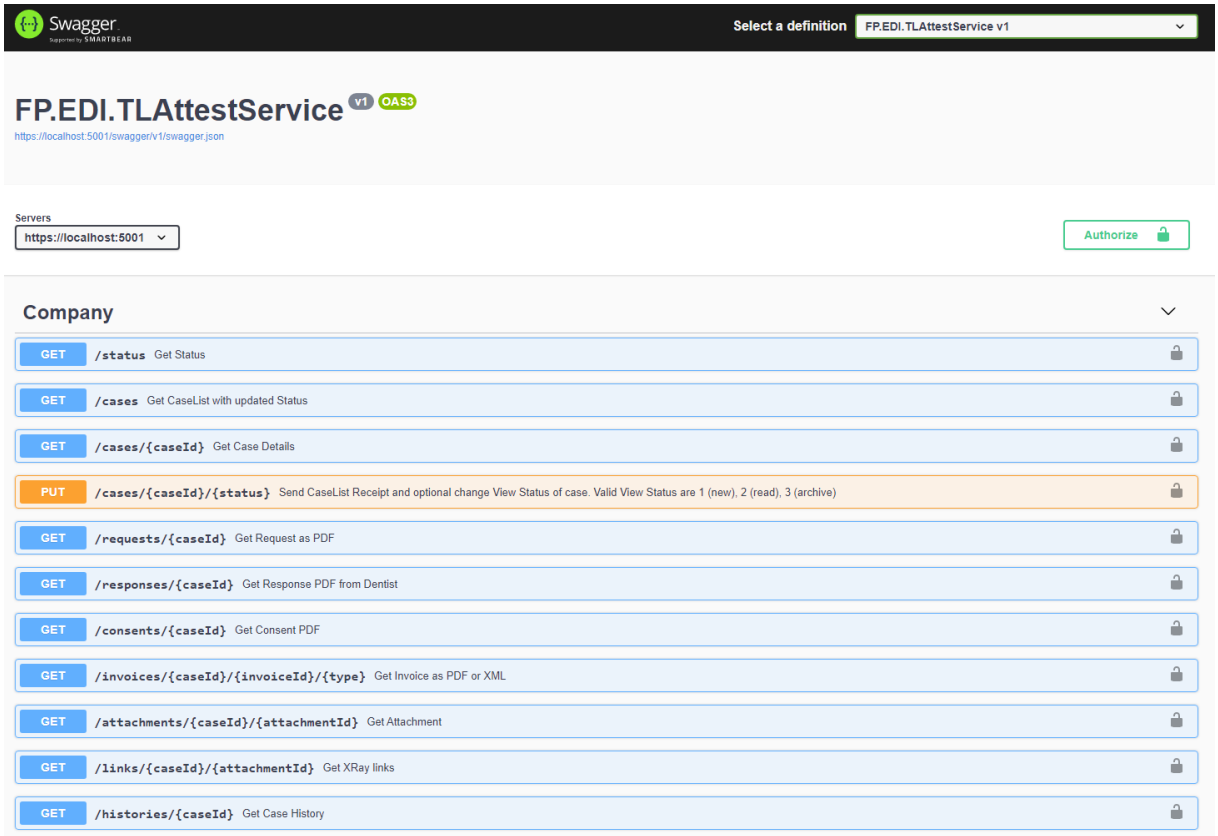
Api'et har følgende endpoints:

Test: https://testedi.forsikringogpension.dk/api/v1/tlattestservice
Produktion: https://edi.forsikringogpension.dk/api/v1/tlattestservice

Online dokumentation

TLAttestService Api'et har en online dokumentation (Swagger), der findes på nedenstående adresse:

https://testedi.forsikringogpension.dk/api/v1/TLAttestService/swagger/index.html



Swagger er et nyttigt værktøj, der beskriver alle operationer og datastrukturer, samt integrerer token og test af operationer.

Operationer

I det følgende beskrives operationer, som Rest Api'et udstiller. Som nævnt findes disse også i en online udgave. Felter der ikke antager nogen værdi kan enten udelades eller udfyldes med null.

Status (GET)			
Denne operation benyttes til at teste autorisationen.			
Input			
Parameter	**Type**	**Obligatorisk**	**Beskrivelse**
Output			
Parameter	**Type**	**Obligatorisk**	**Beskrivelse**
	String		Hello <selskabsnavn>

Cases (GET)			
Denne operation returnerer en liste af sager, hvor der er dokumenter klar til hentning eller hvor sagen får en status, som kræver selskabets opmærksomhed. Se afsnit 7 om statuskoder.			
Input			
Parameter	**Type**	**Obligatorisk**	**Beskrivelse**
caseId	Guid	Ja	Sagens unikke identifikation
Output			
Parameter	**Type**	**Obligatorisk**	**Beskrivelse**
	List[]	Nej	Liste
caseId	Guid	Ja	Sagens unikke identifikation
status	Int	Ja	Sagens aktuelle status
statusDate	DateTime	Ja	Dato/tid for seneste status
requestReady	Bool	Ja	Anmodning klar til hentning
responseReady	Bool	Ja	Svar klar til hentning
consentReady	Bool	Ja	Samtykke klar til hentning
invoiceReady	Bool	Ja	Faktura klar til hentning

Cases/{caseId} (GET)			
Denne operation returnerer oplysninger om en sag.			
Input			
Parameter	**Type**	**Obligatorisk**	**Beskrivelse**
caseId	Guid	Ja	Sagens unikke identifikation
Output			
Parameter	**Type**	**Obligatorisk**	**Beskrivelse**
caseId	Guid	Ja	Sagens unikke identifikation
status	Int	Ja	Sagens aktuelle status
companyName	String	Ja	Selskabets navn
departmentName	String	Ja	Selskabets afdeling
dentistName	String	Ja	Navn på tandklinik
customerName	String	Ja	Kundens navn
customerId	String	Ja	CPR-nummer. Format ddmmåååå-nnnn
referenceNumber	String	Ja	Sagens reference
formType	String	Ja	Blankettype fx TL100
orderNumber	String	Nej	Selskabets ordrenummer. Er kun udfyldt hvis selskabet modtager OIO faktura direkte
createUserName	String	Ja	Navn på bruger der har oprettet anmodnngen
createDate	DateTime	Ja	Dato/tid for oprettelse
responseCopied	DateTime	Nej	Dato/tid for seneste hentning
invoiceCopied	DateTime	Nej	Dato/tid for seneste hentning
consentCopied	DateTime	Nej	Dato/tid for seneste hentning
finishDate	DateTime	Nej	Dato/tid for afslutning af sag
statusDate	DateTime	Ja	Dato/tid for seneste status
deleteDate	DateTime	Nej	Dato/tid for sletning
cancelText	String	Nej	Årsag til fortrydelse
attachments	attachment[]	Nej	Liste af bilag
attachment			
attachmentId	Guid	Ja	Unik bilag id
name	String	Ja	Fx bilag28.docx
size	Integer	Ja	Størrelse i byte
type	Integer	Ja	Type af attachment 1 = faktura, erklæring 2 = faktura, behandling 3 = bilag 4 = røntgenbillede
date	DateTime	Ja	Dato/tid for bilag

Cases/{caseId}/{status} (PUT)

Denne operation benyttes til at kvittere for en række i Cases så den ikke dukker op igen. Endvidere kan man ændre visningen af en sag på Web til ny (ulæst), læst eller arkiveret.

Input

Parameter	Type	Obligatorisk	Beskrivelse
caseId	Guid	Ja	Sagens unikke identifikation
status	Integer	Nej	Sagens nye status 1: Ny 2: Læst 3: Arkiveret

Output

Parameter	Type	Obligatorisk	Beskrivelse

Requests/{caseId} (GET)

Denne operation returnerer anmodningen som PDF.

Input

Parameter	Type	Obligatorisk	Beskrivelse
caseId	Guid	Ja	Sagens unikke identifikation

Output

Parameter	Type	Obligatorisk	Beskrivelse
	Stream		Anmodning som PDF

Responses/{caseId} (GET)

Denne operation returnerer tandklinikkens svarblanket som PDF.			
Input			
Parameter	**Type**	**Obligatorisk**	**Beskrivelse**
caseId	Guid	Ja	Sagens unikke identifikation
Output			
Parameter	**Type**	**Obligatorisk**	**Beskrivelse**
	Stream		Svar blanket som PDF

Consents/{caseId} (GET)			
Denne operation returnerer det underskrevet samtykkeerklæring som PDF.			
Input			
Parameter	**Type**	**Obligatorisk**	**Beskrivelse**
caseId	Guid	Ja	Sagens unikke identifikation
Output			
Parameter	**Type**	**Obligatorisk**	**Beskrivelse**
	Stream		Samtykkeerklæring som PDF

Invoices/{caseId}/{invoiceId}/{type} (GET)			
Denne operation returnerer en faktura som XML og PDF. Default er PDF.			
Input			
Parameter	**Type**	**Obligatorisk**	**Beskrivelse**
caseId	Guid	Ja	Sagens unikke identifikation
invoiceId	Guid	Ja	Faktura id, som er attachmentId fra attachment listen, hvor typen er 1 eller 2
type	String	Ja	Type. pdf eller xml
Output			

Parameter	Type	Obligatorisk	Beskrivelse
	Stream		Faktura i respektive format

Attachments/{caseId}/{attachmentId} (GET)

Denne operation returnerer bilag.
Det er kun bilag af typen 3 der kan hentes med denne operation.

Input

Parameter	Type	Obligatorisk	Beskrivelse
caseId	Guid	Ja	Sagens unikke identifikation
attachmentId	Guid	Ja	Bilag id, som er attachmentId fra attachment listen, hvor typen er 3

Output

Parameter	Type	Obligatorisk	Beskrivelse
	Stream		Bilag

Links/{caseId}/{attachmentId} (GET)

Denne operation returnerer en midlertidigt link til at hente røntgendata. Der returneres et link til thumbnail samt link til selve røntgenbilledet. Linket har en begrænset levetid på 5 minutter.

Input

Parameter	Type	Obligatorisk	Beskrivelse
caseId	Guid	Ja	Sagens unikke identifikation
attachmentId	Guid	Ja	Røntgenbillede id, , som er attachmentId fra attachment listen, hvor typen er 4

Output

Parameter	Type	Obligatorisk	Beskrivelse
name	String	Ja	Filnavnet på røntgenbilledet. Fx Bitewings.png
url	String	Ja	Link til billedet. Dette link indeholder også en SAS token.
thumbnailUrl	String	Nej	Hvis det er muligt at lave en thumbnail, så vil der være et link her.

			Thumbnails er altid i JPG.

Histories/{caseId} (GET)			
Denne operation returnerer en liste over historik på sagen.			
Input			
Parameter	**Type**	**Obligatorisk**	**Beskrivelse**
caseId	Guid	Ja	Sagens unikke identifikation
Output			
Parameter	**Type**	**Obligatorisk**	**Beskrivelse**
	List[]		Liste af historik
status	Integer	Ja	Status
statusDate	DateTime	Ja	Dato/tid for status

6. Blankettyper

Der er følgende blankettyper:

Blankettype	Beskrivelse
TL100	Kæbefunktionsattest
TL110	Kæbefunktionsattest og journaloplysninger
TL200	Tandlægeerklæring
TL210	Journaloplysninger og journaloplysninger
TL300	Journaloplysninger

7. Statuskoder

I det følgende vises en oversigt over statuskoder samt hvornår en sag vises i Cases. Koderne 20x – er ikke statuskoder på sagen, men en handling, der logges i historikken.

Endvidere viser oversigten hvornår et dokument er klar til hentning. Hvis en sag får en status som kræver selskabets opmærksomhed (kolonnen "Vis"), så nulstilles kvitteringsflaget og sagen vises i Cases. Bemærk at sagen kan have skiftet status inden selskabet kalder Cases.

Kode	Statustekst i EDI	Vis	Request Ready	Consent Ready	Invoice Ready	Response Ready
5	Anmodning afsendt	X	Ja	Nej	Nej	Nej
6	Genanmodning afsendt	X	Ja	Ja	*	Nej
8	Samtykke er genanvendt		Ja	Ja	Nej	Nej
10	Afventer kundens samtykke		Ja	Nej	Nej	Nej
20	Samtykke underskrevet af kunden	X	Ja	Ja	Nej	Nej
25	Kunden er rykket for samtykke		Ja	Nej	Nej	Nej
26	Kunden har ikke afgivet samtykke trods rykker	X	Ja	Nej	Nej	Nej
30	Afventer kundens valg af tandklinik		Ja	Ja	Nej	Nej
32	Kunden rykket for valg af tandklinik		Ja	Ja	Nej	Nej
34	Kunden har ikke valgt tandklinik trods rykker	X	Ja	Ja	Nej	Nej
40	Anmodning afventer afsendelse til tandklinik		Ja	Ja	Nej	Nej
49	Anmodning tilgængelig for tandklinik		Ja	Ja	Nej	Nej
50	Anmodning hentet af tandklinik		Ja	Ja	Nej	Nej
55	Tandklinik rykket for manglende svar 1. gang		Ja	Ja	Nej	Nej
56	Tandklinik rykket for manglende svar 2. gang		Ja	Ja	Nej	Nej
57	Tandklinik har ikke svaret	X	Ja	Ja	Nej	Nej
59	Anmodning afvist af tandklinik	X	Ja	Ja	Nej	Ja
75	Anmodning fortrudt/tilbagekaldt af selskabet	X	Ja	Ja	Nej	Nej
80	Svar modtaget fra tandklinikken	X	Ja	Ja	Nej	Ja
85	Faktura modtaget	X	Ja	Ja	Ja	Ja
88	Sag afsluttet af tandklinik	X	Ja	Ja	Ja	Ja
90	Tandklinikkens svar er markeret til sletning	X	Ja	Ja	Ja	Ja
95	Tandklinikkens svar er sendt til sletning		Nej	Nej	Nej	Nej
97	Tandklinikkens svar er slettet		Nej	Nej	Nej	Nej
98	Sag slettet pga. databrud		Nej	Nej	Nej	Nej
201	Attest Hentet					

202	Samtykke hentet					
203	Faktura hentet					
204	Fortrudt afsendt					
205	Kontaktoplysninger rettet					

*Ja, hvis sagen har fået kode 85. Nej, hvis sagen ikke har fået kode 85

8. Returkoder

Rest Api'et returnerer standard http statuskoder.

Her er en oversgt over de mest benyttede koder.

Code	Message	Beskrivelse
200	OK	OK
400	Bad request	Forkert data
401	Unauthorized	Unauthorized
404	Not found	Id blev ikke fundet
500	Internal System Error	Systemfejl

9. Kodeeksempler i C# .NET 5.0

I det følgende vises kodeeksempler på, hvordan man henter en accesstoken, kalder sagslisten, henter en PDF via stream samt kvitterer for en sag.

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;

namespace TLAttestDemo
{
   class Program
   {
      private static readonly HttpClient _httpClient = new HttpClient();
      private static string tokenUrl="https://testedi.forsikringogpension.dk/AuthServer/Oauth/Token";
      private static string _serviceUrl= https://testedi.forsikringogpension.dk/api/v1/TLAttestService";
      private static string _clientId = "selskabets client id";
      private static string _clientSecret = "selskabets client secret";

      public class CaseInfo
      {
         public Guid CaseId { get; set; }
         public int Status { get; set; }
         public DateTime StatusDate { get; set; }
         public bool RequestReady { get; set; }
         public bool ResponseReady { get; set; }
         public bool ConsentReady { get; set; }
         public bool InvoiceReady { get; set; }
      }

      public class OauthToken
      {
         public string access_token { get; set; }
         public string token_type { get; set; }
         public long expires_in { get; set; }
         public string refresh_token { get; set; }
         public DateTime issued { get; set; }
```

```
        public DateTime expires { get; set; }
    }
static void Main(string[] args)
{
   try
   {
      var accessToken = GetOAuth2Token(_clientId, _clientSecret, "tlattest");
      Console.WriteLine("AccessToken, size=" + accessToken.Length);

      var caseList = GetCaseList(accessToken);
      Console.WriteLine("CaseList, count:" + caseList.Count);

      if (caseList.Count > 0)
      {
         var caseId = caseList[0].CaseId; // first case

         Stream requestPdf = GetRequest(caseId, accessToken);
         using (var fileStream = new FileStream(@"c:\download\request" + caseId.ToString() + ".pdf",
         FileMode.Create, FileAccess.Write))
         {
            requestPdf.CopyTo(fileStream);
         }

         var receiptStatus = SendReceipt(caseId, 2, accessToken);
         Console.WriteLine("ReceiptStatus:" + receiptStatus);
      }
   }
   catch (Exception ex)
   {
      Console.WriteLine("Error: " + ex);
   }
}


private static string GetOAuth2Token(string clientId, string clientSecret, string scopes)
{
   string credentials = String.Format("{0}:{1}", clientId, clientSecret);

   //Define Headers
   _httpClient.DefaultRequestHeaders.Clear();
   _httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
   _httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic",
   Convert.ToBase64String(Encoding.UTF8.GetBytes(credentials)));

   //Prepare Request Body
   var requestData = new List<KeyValuePair<string, string>>();
   requestData.Add(new KeyValuePair<string, string>("grant_type", "client_credentials"));
   requestData.Add(new KeyValuePair<string, string>("scope", scopes));
   var requestBody = new FormUrlEncodedContent(requestData);

   //Request Token
   using (var response = _httpClient.PostAsync(_tokenUrl, requestBody).Result)
   {
      response.EnsureSuccessStatusCode();
      var result = response.Content.ReadAsStringAsync().Result;
      var data = JsonConvert.DeserializeObject<OauthToken>(result);
      return data.access_token;
   }
 }

private static List<CaseInfo> GetCaseList(string accessToken)
{
   var requestEndPoint = _serviceUrl + "/cases";

   //Define Headers
   _httpClient.DefaultRequestHeaders.Clear();
   _httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
   _httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", accessToken);

   using (var response = _httpClient.GetAsync(requestEndPoint).Result)
```

```
    {
        response.EnsureSuccessStatusCode();
        var result = response.Content.ReadAsStringAsync().Result;
        var data = JsonConvert.DeserializeObject<List<CaseInfo>>(result);
        return data;
    };
}

private static Stream GetRequest(Guid caseId, string accessToken)
{
    var requestEndPoint = _serviceUrl + "/requests/" + caseId;

    //Define Headers
    _httpClient.DefaultRequestHeaders.Clear();
    _httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
    _httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", accessToken);

    var streamToWriteTo = new MemoryStream();
    using (var response = _httpClient.GetAsync(requestEndPoint,
        HttpCompletionOption.ResponseHeadersRead).Result)
    {
        response.EnsureSuccessStatusCode();
        using (Stream streamToReadFrom = response.Content.ReadAsStreamAsync().Result)
        {
            streamToReadFrom.CopyTo(streamToWriteTo);
            streamToWriteTo.Position = 0;
        }
        return streamToWriteTo;
    };
 }

private static string SendReceipt(Guid caseId, int status, string accessToken)
{
    var requestEndPoint = _serviceUrl + "/cases/" + caseId + "/" + status;

    //Define Headers
    _httpClient.DefaultRequestHeaders.Clear();
    _httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
    _httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", accessToken);

    using (var response = _httpClient.PutAsync(requestEndPoint, null).Result)
    {
        response.EnsureSuccessStatusCode();
        var result = response.Content.ReadAsStringAsync().Result;
        return result;
    };
}

} //class
} //namespace
```