**e-Tinglysning**

Integration -

Kommunikationsspecifikation

Version 1.0, 14.05.2009

**CSC Danmark A/S**

Retortvej 8, Valby

DK-1780 København

Phone: +45 3614 4000



Dokumentoplysninger

|  |  |
| --- | --- |
| **Titel:** | Integration - Kommunikationsspecifikation  |
| **Projekt:** | eTinglysning |
| **Version** | 1.0 |
| **Forfatter:** | Hans Ernst Lassen |
| **Bidragsydere til dokumentet:** |  |
| **Aktivitet ID:** | IS-013 (gl. 33.080) |
|  |  |
| **Udskrevet:** | 10.11.08 |

Ændringslog

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Dato** | **Ændrede sider eller afsnit** | **Kommentarer** |
| 0.5 | 08.09.2008 | Alle | Initiel version |
| 0.6 | 01.06.2008 | 3.2 og 3.5 | Rettet *timeout* til *fejl* i afsnit 3.2. Tilføjet kolonne i tabellen i afsnit 3.5. |
| 0.7 | 09.10.2008 | Afsnit 3 og frem | Dokumentet er klargjort til møde.Flyttet beskrivelsen om fejl til eget afsnit. Indført kort beskrivelse af beskedflowet i afsnit 3. |
| 0.8 | 24.10.2008 | Alle | Mere detaljeret fejlbeskrivelse og kommunikationsbeskrivelse. |
| 0.8a | 07.11.2008 | 5.2.4 og 5.2.5 | Rettet beskrivelse af SOAP fault |
| 0.9 | 10.11.2008 |  | Version sendt til godkendelse i DSS |
| 1.0 | 14.05.2009 |  | Godkendt (HHJ) |

Indholdsfortegnelse

1. Indledning 4

1.1 Formål 4

1.2 Omfang 4

1.3 Udeståender 4

2. Baggrund 5

2.1 Kommunikation med SOAP over HTTPS 5

3. Kommunikationsdesign 6

3.1 Koncept for kommunikation 6

3.2 Sikkerhedshåndtering 6

3.3 Synkron interaktion 6

3.4 Rækkefølge 6

3.5 Snitflader 6

3.6 Reliability 7

3.6.1 Besked tabel 7

3.7 Tinglysningsflow, med offentliggørelse til eksterne datamodtagere 8

4. Fejl scenarier 10

4.1 Problemer ved ”unreliable” kommunikation 10

4.2 Result-response-received 10

4.3 Fault-response-received 10

4.4 No-response-received 10

4.4.1 No request 11

4.4.2 No acknowledge 11

4.5 Consumer-has-crashed 12

4.6 Resumé over de problemer der er forbundet med hver type fejlsituation 12

5. Fejlhåndtering 14

5.1 Fejl situationer hos datamodtageren 14

5.1.1 Teknisk driftsstop 14

5.1.2 Skema versions missmatch 14

5.1.3 Signeringsfejl 14

5.1.4 Certifikatproblemer 14

5.1.5 Ukendt fejl 14

5.2 SOAP Faults 14

5.2.1 faultcode (krævet) 15

5.2.2 faultstring (krævet) 15

5.2.3 faultactor (valgfri) 15

5.2.4 detail (valgfri) 15

5.2.5 Eksempel på SOAP Fault 15

5.3 Fejlkoder 16

6. Meta-data 17

6.1 WS-Addressing 17

6.2 Sekvensnummer 17

# Indledning

## Formål

Formålet med dette dokument er at beskrive løsningsdesignet for pålidelig (reliable) kommunikation mellem datamodtagere og e-TL. Datamodtagere er typisk offentlige myndigheder der skal modtage en kopi af de anmeldelsessvar e-TL udsender til aktørerne. Der er tale om en overgangsløsning, da DSS ønsker at e-TL på sigt understøtter pålidelig kommunikation via RASP.

Datamodtagere er i skrivende stund SKAT, ESR, etc. En vigtig funktionalitet er at alle systemer kommunikerer via en reliable protokol, da datamodtagerne skal have oplysninger vedrørende tinglysninger. DSS har bedt om at datamodtagerne kan kommunikere med eTL via en reliable protokol, da HTTP/SOAP adgangen ikke i sig selv stiller nogen form for reliability til rådighed. e-TL og datamodtagerne har ikke selv mulighed for at understøtte en ægte reliable protokol (f.eks. JMS eller MQ). Nedenstående beskriver, hvordan vi i så vid udstrækning som muligt kan håndtere situationer hvor beskeder kan mistes.

## Omfang

Dokumentet beskriver den overordnede arkitektur for kommunikationen mellem datamodtagere og e-TL og de arkitekturmæssigt mest betydningsfulde komponenter, deres afhængigheder, interaktion og kommunikationsmekanismer.

Dokumentet er et ’levende’ dokument, og vil blive opdateret løbende gennem udviklingsforløbet.

## Udeståender

Følgende kendte emner udestår:

* Ingen

# Baggrund

Dette afsnit beskriver det overordnede design der skal implementeres i e-TL og integrationsløsningen. Hovedsigtet med det præsenterede design er at tilgodese de behov datamodtagere måtte have til reliable kommunikation med e-TL og samtidig muliggøre en realisering, der i meget vid udstrækning gør brug af funktionalitet der findes i e-TL og datamodtagernes systemer.

## Kommunikation med SOAP over HTTPS

Da HTTPS kører over TCP, er protokollen pålidelig så længe hverken consumer, provider eller nettet mellem dem går ned. Med SOAP/HTTPS risikerer man dog følgende fejlsituationer: No-response-received og Consumer-has-crashed.

# Kommunikationsdesign

I det følgende beskrives designet for kommunikationen mellem e-TL og datamodtagerne. Kommunikationen foregår med SOAP over HTTPS. Kommunikationen foregår med SOAP over HTTPS og benytter de services der er beskrevet i service kataloget, f.eks. AnmeldelseKopiSvar servicen.

Udgangspunktet er at kommunikationen i det store hele forløber uden problemer og uden mistede beskeder, men at der i sjældne tilfælde kan opstå fejl, der kan være svære at spore. Designet prøver at tage hensyn til begge situationer samtidigt med at driften involveres minimalt.

## Koncept for kommunikation

Integration til datamodtagere realiseres således at e-TL sender beskeder i formater, som er defineret for e-TL løsningen. Det vil sige at e-TL ved afsendelse til en datamodtager opretter en forsendelse, hvor forsendelseskroppen er i præcis det format, som er defineret for e-TL (og som datamodtageren forventer at modtage). Tilsvarende vil e-TL modtage svar, hvor forsendelseskroppen er i præcis det format, som e-TL har defineret.

For at kunne gennemføre den nødvendige behandling af beskeder via integrationen til datamodtagerne er det nødvendigt, at datamodtagerne specificerer e-TL specifikke elementer som del af forsendelseshovedet (i SOAP-headeren).

## Sikkerhedshåndtering

e-TL autentificeres datamodtagerens system baseret på e-TL signatur af forsendelseskroppen. Det er ikke nødvendigt, at datamodtageren signerer kvitteringen for modtagelsen.

Kommunikationen foregår via HTTPS

## Synkron interaktion

Det synkrone interaktionsmønster svarer til et traditionelt funktionskald i normale software termer. Den aktiverede funktion har typisk en forholdsvis kort aktiveringsperiode, dvs. svaret returneres inden for en kortere tidshorisont (målt i sekunder).

## Rækkefølge

e-TL kan ikke sikre rækkefølge af forsendelser til datamodtageren. Da det kræves at datamodtageren behandler meddelelserne i en bestemt rækkefølge, er det sikret ved at forsendelserne indeholder et sekvensnummer. Datamodtageren skal aflæse nummeret og sortere meddelelserne derefter.

Sekvensnummeret dannes ud fra tinglysningstidspunktet og da dette tidspunkt dannes inden for e-TLs tinglysningslås, garanteres det at anmeldelsessvar, som omhandler samme tinglysningsobjekt, nummereres i den rækkefølge som e-TL har modtaget og tinglyst anmeldelserne i.

Sekvensnummeret er bundet til en web service, ikke per operation. Det vil i dette tilfælde sige AnmeldelseKopiSvar.wsdl.

## Snitflader

I det følgende er overordnet skitseret funktionerne der indgår i integration til datamodtagerne. De konkrete detaljerede snitfladebeskrivelser for funktionaliteten, gøres tilgængelige via WSDL-filer og XML-skemaer.

Følgende service samt tilhørende operationer gøres tilgængelige for integrationen (baseret på de e-TL WSDL som var tilgængelige pr 23-10-2008):

* AnmeldelseKopiSvar
	+ AnmeldelseSvarModtag
	+ AnmaerkningStatusModtag

For en forretningsmæssig beskrivelse af de enkelte operationer, henvises til løsningsbeskrivelserne fra Domstolsstyrelsen.

## Reliability

Løsningen baseres på HTTPS, der i sin natur ikke er reliable. For at komme det problem i møde er der opstillet en række fejlscenarier med en løsningsmodel for, hvordan disse håndteres korrekt. Disse løsningsmodeller stiller krav til hvordan og i hvilken rækkefølge afsendelse, modtagelse og behandling af beskederne foregår.

Nedenstående figur viser hvilke handlinger e-TL og datamodtageren skal foretage sig og i hvilken rækkefølge. I figuren antages det at ingen beskeder mistes eller systemer går ned.



*sendTinglysningsSvar:* e-TL integrationsprojektet får en notifikation om at skulle sende et tinglysningssvar til en datamodtager.

*insertTLSvar:* Der genereres et nyt nummer (et unikt sekvensnummer, herefter SN) i en fortløbende nummerrækkefølge der kobles til beskeden og indsættes i en database med dette SN samt en status ’afsendt’.

*modtagTinglysningsSvar:* e-TL kalder en web service hos datamodtageren og sender tinglysningssvaret inklusiv SN’et.

*gemTinglysningsSvar:* Ved modtagelse af tinglysningssvaret persisterer datamodtageren beskeden, hvilket sikrer at datamodtageren ikke mister beskeden efter afsendelse af kvitteringen, som afsendes senere.

*tinglysningsSvarBehandling:* Datamodtageren behandler beskeden, f.eks. ved at mappe den til datamodtagerens interne model. Bemærk at behandlingen bør foregå asynkront for at kunne lave et hurtigt acknowledge til e-TL.

*acknowledge:* Når tinglysningssvaret er persisteret og behandlingen af svaret er sat i gang, svares der tilbage med et acknowledge som beskrevet i WSDL’en.

*updateTLSvar:* Ved modtagelse af et acknowledge fra datamodtageren, opdaterer e-TL sin database således at den afsendte besked får status ’modtaget’.

### Besked tabel

Når beskeder skal sendes til en datamodtager, oprettes der en status for beskeden i en tabel i databasen. Denne tabel indeholder felterne; *ID*, *besked* og *status*. ID er et unikt id for beskeden. Besked er indholdet af den besked, der skal sendes, f.eks. et tinglysningssvar. En status kan være *afsendt*, *modtaget*, *timeout* eller *fejl*.

## Tinglysningsflow, med offentliggørelse til eksterne datamodtagere

Nedenstående figur viser et eksempel på et tinglysningsflow, hvor anmeldelser modtages, offentliggøres og eksterne datamodtagere underrettes med en kopi af tinglysningssvaret. I figuren antages det at der kun modtages anmeldelser, der har relevans for den eksterne datamodtager.



**Modtagelseskø:** eTL modtager en række anmeldelser, der sorteres og lægges i eTL modtagelseskøen. Sorteringen foregår således, at anmeldelser der vedrører samme tinglysningsobjekt behandles sekventielt.

**Prøvelse:** Under prøvelsen behandles anmeldelserne parallelt, dog sikres det at to anmeldelser, der vedrører samme tinglysningsobjekt ikke trækkes af modtagelseskøen og dermed ikke behandles parallelt. Dette håndteres af låsninger på modtagelseskøen.

**Offentliggørelse/Integration:** Ved offentliggørelsen sendes en kopi af tinglysningssvaret til den eksterne datamodtager. Svarene nummereres efter tinglysningstidstemplet. Det kan ikke sikres at denne rækkefølge stemmer overens med rækkefølgen, hvormed anmeldelserne er trukket af modtagelseskøen.

**Afsendelse over HTTPS:** Tinglysningssvarene sendes over HTTPS til den eksterne datamodtager. Denne afsendelse er ikke længere underlagt tinglysningslåsen og rækkefølgen kan ikke længere garanteres, da den foregår parallelt.

**Modtagelse:** Datamodtageren modtager svaret og kvitterer for modtagelsen. Under denne modtagelse kan datamodtageren heller ikke garantere for rækkefølgen.

**Behandling:** Datamodtageren skal sortere svarene efter nummerrækkefølgen og sikre at der ikke behandles svar, hvis der er huller i nummerrækkefølgen.

# Fejl scenarier

## Problemer ved ”unreliable” kommunikation

Vi har to systemer: DSS (*consumer*) og datamodtager (*provider*), hvor consumer kalder en operation hos provider (request), hvorefter consumer får et svar retur fra provider (response).

Hvis consumer kalder provider vha. en alm. unreliable request/response-protokol som fx SOAP/HTTPS kan man underopdele de mulige udfald i 4 forskellige kategorier, som gennemgås én af gangen i det efterfølgende:

*Result-response-received*

Consumer modtager et svar fra provider. Consumer ved dermed, at operationen er udført af provider med success. M.a.o. ved consumer, at tilstandsændringer som følge af operationens kørsel hos provideren er blevet commited.

*Fault-response-received*

Consumer modtager en fejl fra provider. Consumer ved dermed, at kaldet er fejlet hos provider. M.a.o. ved consumer, at tilstandsændinger som følge af operationens kørsel hos provider er blevet rolledback.

*No-response-received*

Consumer modtager intet svar fra provider, men får sikkert en fejl genereret af kommunikations­framewor­ket i stedet. Consumer ved nu ikke, om kaldet nåede at blive kørt hos provider, og om det i så fald commitede eller resulterede i rollback. Fejlen kan jo skyldes, at kaldet aldrig nåede frem til provider, at provider crashede eller at svaret fra provider ikke nåede tilbage til consumer.

*Consumer-has-crashed*

Consumer når at lave kaldet og consumer crasher så – enten inden consumer har modtaget svaret eller efter consumeren har modtaget svaret, men inden consumeren kunne nå at afslutte sin egen transaktion og / eller at gemme svaret. Provideren ved ikke noget om, at consumer er crashet / vil crashe, og forsøger derfor på at udføre (commite) operationen. Resultatet er, at consumeren ikke når at reagere på svaret inden consumeren selv crasher. Da consumeren ikke ved om kaldet lykkedes, vil consumeren evt. prøve at køre kaldet endnu engang, når consumeren er genstartet (hvilket kan medføre uønskede konsekvenser).

## Result-response-received

Figuren samt tilhørende forklaringstekst i afsnit 3.6 er et eksempel på result-response-received.

## Fault-response-received

Ved modtagelse af en SOAP fault fra datamodtagerens servicelag, vil vi forsøge at gensende beskeden 2 gange. Hvis alle forsøg fejler, opdateres besked tabellen så status på beskeden er *fejl*. Herefter håndteres fejlen af drift. Se i øvrigt afsnit **Error! Reference source not found.** om SOAP faults.

## No-response-received

Hvis e-TL ikke modtager et acknowledge eller en fault fra en datamodtager, kan der opstå to situationer som begge skal håndteres af drift.

### No request



*sendTinglysningsSvar, insertTLSvar:*Som ved Result-response-received*.*

*modtagTinglysningsSvar:* e-TL kalder en web service hos datamodtageren, men kaldet udføres ikke hos datamodtageren.

*timeout:* Forbindelsen til datamodtageren timer ud. Besked tabellen opdateres så status på beskeden er *timeout*.

### No acknowledge



*sendTinglysningsSvar, insertTLSvar*, *modtagTinglysningsSvar, gemTinglysningsSvar, tinglysningsSvarBehandling:* Som ved Result-response-received.

*http acknowledge:* Acknowledge modtages ikke korrekt hos e-TL.

*updateTLSvar:* Forbindelsen til datamodtageren timer ud. Besked tabelen opdateres så status på beskeden er *timeout*.

## Consumer-has-crashed



e-TL kan crashe på et vilkårligt tidspunkt efter svaret er sendt og dermed ved vi ikke om det er blevet modtaget af datamodtageren, da vi ikke modtager noget acknowledge eller nogen fault.

## Resumé over de problemer der er forbundet med hver type fejlsituation

Tabellen nedenfor opstiller, hvilke problemer der er associeret med hvilke typer af fejlsituationer.
Dvs. tabellen er i praksis blot et overbliksorienteret resume af ovenstående sektioner.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Consumers tilstand** | **Providers tilstand** | **Besked table** | **Håndtering** |
| *Result-response-received* | Velkendt (consumer bør selv commite, og hvis det fejler, så bør consumeren initiere en kompenserende TX, der annulerer tilstandsændringen hos provideren) | Velkendt (commited) | Status:modtaget | Ingen fejl |
| *Fault-response-received* | Velkendt (bør selv lave rollback og bør initiere kompenserende handling i forbindelse med operationer den har kaldt hos andre providers) | Velkendt (rolledback) | Status:fejl | Gensendelse og derefter håndtering af drift. |
| *No-response-received* | Velkendt (kan evt. logge problemet) – men det kan være svært at afgøre, om consumeren bør commite eller lave rollback, da den ikke aner, hvad der skete i den anden ende. | Ukendt (kan dog muligvis researches frem vha. log-filerne og / eller analyse af data i providerens databaser) | Status:timeout | Gensendelse og derefter håndtering af drift. |
| *Consumer-has-crashed* | Ukendt (systemet sikrer dog nok rollback). Problemet er her, at vi ikke umiddelbart kan advare om problemet (da vi er crashet). | Ukendt for consumeren (da consumeren ikke får / ikke når at reagere på svaret) – kan dog nok researches frem vha. log-filerne og / eller analyse af providerens databaser. | Status:afsendt | Gensendelse og derefter håndtering af drift. |

Det er både *No-response-received* og *Consumer-has-crashed* kan principielt håndteres ved at consumer bliver ved med at forsøge at lave kaldet igen indtil det lykkes, dette kan gøres med idempotente operationer. Der er dog visse former for kald, som er årsag til ændringer på provider, hvorfor man ikke uden videre kan blive ved og ved med at kaldet dem, indtil man får enten et resultat eller en fejl retur – fx en operation, som tilføjer 100kr til en konto. Da vi ikke har idempotente operationer kan vi ikke benytte gensendelse ved andre situationer end fault-response-received, hvor vi ved at operationen ikke er udført.

# Fejlhåndtering

## Fejl situationer hos datamodtageren

Når en datamodtager registrerer en fejl, der skal rapporteres til e-TL, gøres dette vha. SOAP faults. Der er identificeret fem forskellige situationer, der kan resultere i at en datamodtager sender en SOAP fault til e-TL, disse er beskrevet i det følgende.

### Teknisk driftsstop

Denne exception kan datamodtager (f. eks. ESR) anvende, når de ønsker at signalere store problemer i infrastrukturen, der forhindrer dem i at modtage beskeder. Det kan f. eks. være de situationer, hvor de IPL'er deres mainframe. Uden mainframe kan de ikke modtage.

### Skema versions missmatch

Denne exception er udtryk for at datamodtager (f. eks. KMD) er kommet ud af sync med de XML Schemaer e-TL løsningen baserer sig på. Resultatet af denne situation skal være den samme som teknisk driftstop. Det er datamodtagers (f. eks. KMD) pligt i denne situation at kontakte Tinglysningsrettens helpdesk.

### Signeringsfejl

Denne exception er udtryk for enten

* at tinglysningsrettens signatur er brudt og beskeden dermed er ugyldig. CSC skal afklare nærmere og efter løsning gensende beskeden.
* at datamodtager (f. eks. KMD) har fejl i deres løsning, som betyder at de ikke kan validere timglysningsrettens signatur. Datamodtager skal i denne situation løse fejlen.

### Certifikatproblemer

Denne exception er udtryk for at KMD ikke kan validere certifikatet der er anvendt i forbindelse med Tinglysningsrettens signatur.

* CSC forsøger at gensende beskeden efter et tidsinterval.

### Ukendt fejl

Denne exception dækker alle andre fejl situationer. Det skal aftales nærmere, hvilken information der skal medtages i denne situation.

## SOAP Faults

Opstår der en fejl ved modtagelse eller behandling af forsendelse, skal fejlen rapporteres af den synkrone svar-kanal. Der benyttes SOAP fejlrapporteringsmodel. Dette er vist på nedenstående figur, hvor e-TL kalder en datamodtager og modtager en SOAP Fault.

Aktør

e-TL

WS-operation

**Reglerne for SOAP Fault generering følges***WS-I BP 1.1 reglerne for generering af SOAP Fault følges. Reglerne tillader visse specialtilpasninger. Disse tilpasninger beskrives i yderligere detaljer nedenfor.*

De overordnede regler fra WS-I BP 1.1 dikterer at der returneres en <soap:Body> med præcist et <soap:Fault> child element. Det eneste indhold som tillades i <soap:Fault> elementet er elementerne <faultcode>, <faultstring>, <faultactor> og <detail>.

Bemærk specielt at ”child” elementerne *ikke* må have et namespace prefix (unqualified).

Skabelonen for en SOAP Fault som returneres til e-tinglysningssystemet vil derfor se ud som følger:

<soap:Fault xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' >
 <faultcode> … </faultcode>
 <faultstring> … </faultstring>
 <faultactor> … </faultactor>
 <detail> … </detail>
</soap:Fault>

Der benyttes soap-fault meddelelsen defineret i SOAP 1.1 standarden. Følgende er en overordnet beskrivelse af soap-fault elementerne der skal benyttes i løsningen:

### faultcode (krævet)

Faultcode elementet er tiltænkt anvendt til overordnet klassificering af fejltypen, og som en basal identifikation af om der systemmæssigt kan gøres noget for at håndtere fejlen.

Faultcode elementet er tiltænkt anvendt til overordnet klassificering af fejltypen, og som en basal identifikation af om der systemmæssigt kan gøres noget for at håndtere fejlen.

Elementet *faultcode* vil indeholde en af de 4 standard fejlkoder fra SOAP 1.1 W3C Note (VersionMismatch, MustUnderstand, Client eller Server).

* **Client**"Client" indikerer at meddelelsen var ukorrekt formateret eller ikke indholdsmæssigt lever op til service kontrakten, således at servicen kunne fortolke forespørgslen korrekt. Dette er generelt en indikation af at beskeden ikke skal gensendes før den er modificeret.
* **Server**"Server" indikerer at meddelelsen ikke kunne behandles af årsager der ikke direkte er knyttet til indholdet i selve beskeden, men til selve servicen. Dette indikerer at beskeden kan gensendes senere uden modifikation.

### faultstring (krævet)

Faultstring elementet har til formål at give en menneskelig læsbar forklaring af en fejl, og er ikke egnet til algoritmisk behandling. I e-TL vil denne indeholde en kort beskrivelse af fejlen.

### faultactor (valgfri)

Faultactor elementet er af typen URI, og har til formål at beskrive hvor fejlen kan spores til. I elementet medsendes information om den entitet som har givet anledning til fejlen. Denne information vil være på formen ”urn:#AnmeldelseSvarModtag” (eksempel).

### detail (valgfri)

Detail elementet skal og må jf. SOAP 1.1 kun bruges til fejl informationer omhandlende SOAP body. I e-TL benytter vi den til at sende de i afsnit 5.3 definerede fejlkoder samt en beskrivelse.

Der benyttes de vedlagte WSDL’er og XML-Schemaer der beskriver syntaksen for fejlkoden og beskrivelsen.

### Eksempel på SOAP Fault

<soap:Fault xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' >
 <faultcode>Client</faultcode>
 <faultstring>Signeringsfejl</faultstring>
 <faultactor>urn:#AnmeldelseSvarModtag</faultactor>
 <detail>

 <Modtagerfejl>

 <Fejlkode>3</Fejlkode>

 <Fejlbeskrivelse>Der mangler en ID attribut i signaturen</Fejlbeskrivelse>

 </Modtagerfejl>

 </detail>
</soap:Fault>

## Fejlkoder

Der er identificeret følgende fejlkoder.

|  |  |
| --- | --- |
| 1 | Teknisk driftsstop |
| 2 | Skema versions mismatch |
| 3 | Signeringsfejl |
| 4 | Certifikatproblemer |
| 5 | Ukendt fejl |

# Meta-data

## WS-Addressing

WS-Addressing benyttes som beskrevet i Service Kataloget.

## Sekvensnummer

Det beskrevne sekvensnummer i afsnit 3.4, der knyttes til alle beskeder der sendes fra e-TL placeres i SOAP headeren (se WSDL for AnmeldelseKopiSvar servicen).

Et eksempel på en sådan header kunne se således ud (der er udeladt namespace declarations af pladsmæssige hensyn):

<soap:Envelope>

 <etle:Header>

 <etl:Sekvensnummer>243</etl:Sekvensnummer>

 </soap:Header>

 <soap:Body>
 <msg:AnmeldelseSvarModtag>

 <!-- indhold -->
 </msg:AnmeldelseSvarModtag>

 </soap:Body>

</soap:Envelope>