

KOMBIT

Kommunernes it-fællesskab



Transaktionsspor og Fejlbehandling ved servicekald

version 1.7

Kommunernes Data & Infrastruktur - KDI

Indhold

Indhold.....	2
1 Servicedesign.....	4
1.1 Løsningsmønstre	4
1.1.1 Synkron kald.....	4
1.1.2 Asynkron kald med korrelation	5
1.1.3 Asynkron kald uden korrelation	6
2 Transaktioner og fejlbehandling	7
2.1 Transaktionsspor og RequestId.....	7
2.1.1 Sammenhæng mellem kald og svar.....	7
2.1.2 Synkron kald.....	7
2.1.3 Asynkron kald med korrelation	7
2.1.4 Asynkron kald uden korrelation	8
2.1.5 Transaktionsid og –tid	8
2.2 Brug af RequestId.....	8
2.2.1 RequestId i synkron kald.....	8
2.2.2 RequestId i asynkron kald.....	9
2.3 Tværgående transaktionsspor	9
2.3.1 Tværgående transaktionsidentifiser	9
2.4 Kaldskontekst for SOAP/XML Webservices.....	11
2.4.1 HovedOplysninger	11
2.4.2 Transaktionsspor	11
2.4.3 Kaldskontekstelementer.....	12
2.4.4 Rute.....	12
2.4.5 Processingelementer	13
2.5 Kaldskontekst for RESTful web APIs.....	13
2.5.1 HTTP-Header.....	13
2.5.2 Transaktionsspor	14
2.5.3 RequestId.....	14
2.5.4 Kaldskontekstelementer	15
2.5.5 Rute.....	15
2.5.6 Processingelementer	16
2.6 Fejlhåndtering for SOAP/XML Webservices.....	17
2.6.1 Transaktionsspor	17
2.6.2 HovedOplysningerSvar.....	17

2.6.3	Struktur af en fejl	18
2.6.4	FejlId og AdvisId	18
2.6.5	FejlTekst og AdvisTekst.....	19
2.6.6	Kildeld	19
2.6.7	Identifikation.....	19
2.7	Fejlhåndtering for RESTful web APIs.....	19
2.7.1	Transaktionsspor	20
2.7.2	HTTP Header	20
2.7.3	Struktur af en SvarReaktion.....	21
2.7.4	FejlId og AdvisId	23
2.7.5	FejlTekst og AdvisTekst.....	23
2.7.6	Kildeld	24
2.7.7	Identifikation.....	24
2.7.8	Status	24
2.8	Forventninger til kalders håndtering af fejl og transaktionsspor	29
2.8.1	Kalders håndtering af gentagne kald med samme transaktionsId	29
2.8.2	Kalder håndtering af fejl i kald.....	29
2.9	Forventninger til udstillers håndtering af fejl og transaktionsspor	29
2.9.1	Serviceudstiller ved ren KOMBIT contract-first	29
2.9.2	Serviceudstiller ved andre integrationer	30
2.10	Forventninger til mediators håndtering af fejl og transaktionsspor	30
2.10.1	Mediator håndtering af syntaksfejl i kald.....	31
2.10.2	Mediator håndtering af systemexceptions i kald.....	31
2.10.3	Mediator håndtering af gentagne kald med samme transaktionsId	31
2.10.4	Mediator håndtering af fejl i viderekald	31
2.10.5	Mediator håndtering af fejl i ugyldigt svar	32
2.10.6	Mediator håndtering af fejl i gyldigt svar	32

1 Servicedesign

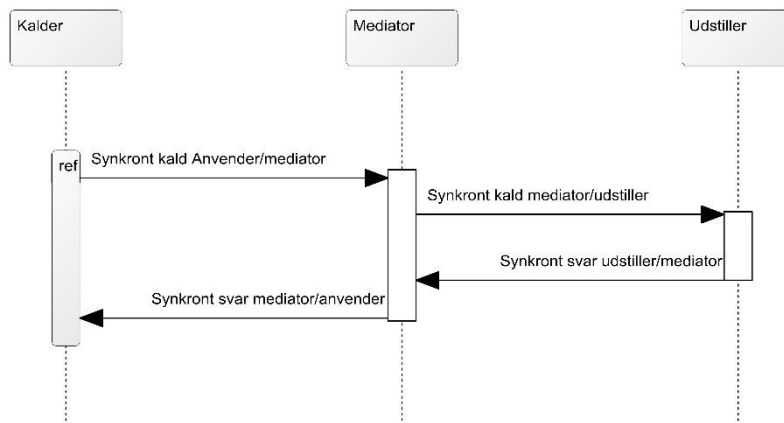
Valg af et hensigtsmæssigt løsningsmønster er en del af ethvert servicedesign. I KOMBIT anvendes der en række forskellige løsningsmønstre, som hver især har forskellige egenskaber mht. forventninger mellem parterne, ansvarsoverdragelse, initiativ og timing.

1.1 Løsningsmønstre

Nedenfor beskrives nogle af de mest almindeligt anvendte løsningsmønstre.

1.1.1 Synkrone kald

Ved simple synkrone kald består konversationen af kaldet og dets synkrone svar.



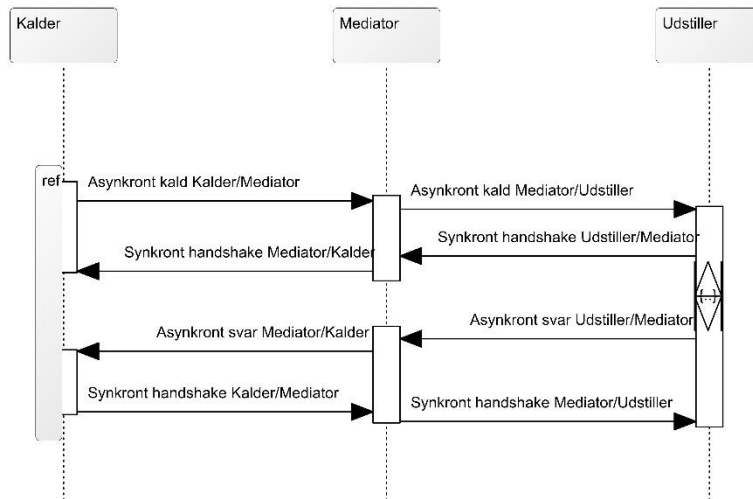
I det simple synkrone kald sker ansvarsoverdragelsen mellem Kalder og Udstiller når det synkrone handshake fra Udstiller når tilbage til Kalder, som ved modtagelse kan afslutte behandlingen af kaldet.

1.1.2 Asynkrone kald med korrelation

Et andet almindeligt integrationsmønster er det asynkrone kald, hvor der er et tidsspænd imellem kald og svar i den samme konversationen. Dette kan implementeres på flere måder, som beskrevet nedenfor, men alle situationer betragtes som en samlet konversation.

1.1.2.1 Push svar

I denne variant består konversationen af et synkront kald, med kalder som initiativtager, samt et svar der foretages som et andet synkron tilbagekald med oprindelig Udstiller som kalder og Kalder som udstiller.

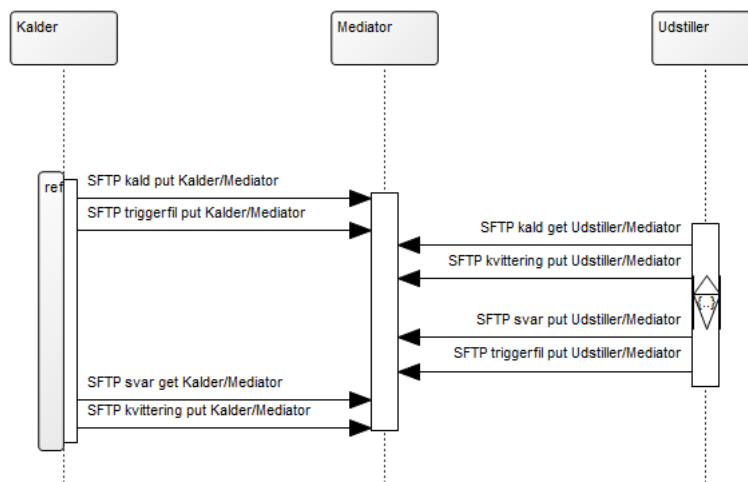


Dette er et generelt løsningsmønster for integration, og der er ikke nogen bindinger på protokollen. Man kan for så vidt bruge mønsteret med en kombination af protokoller.

Ansvarsoverdragelsen fra Kalder til Udstiller kan afsluttes når det asynkrone svar når tilbage til Kalder.

1.1.2.2 Push svar SFTP

I denne variant består konversationen af en overførsel af en payload til mediator, med kalder som initiativtager. Udstiller forventes aktivt at "lytte" på fil samt et svar der foretages som et andet synkron tilbagekald med oprindelig Udstiller som kalder og Kalder som udstiller.



Dette er et løsningsmønster for integration baseret på FTP, SFTP eller FTPS. Kald sker ved overførsel af kaldspayload, efterfulgt af overførsel af en "triggerfil". Kalder har ikke en teknisk garanti for overførsel til Udstiller, idet tilstandstyringen af overførslen er i hænderne på Mediatoren.

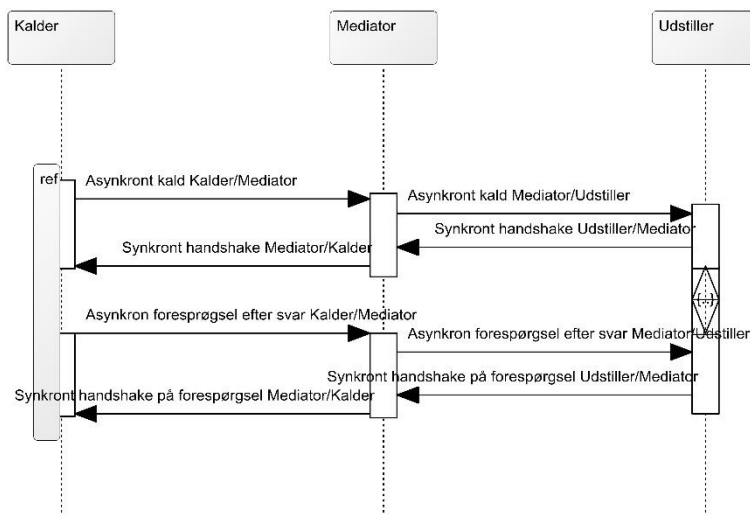
Overførsel er først sket når Udstiller kvitterer for afhentning til mediatoren. Denne kvittering logges i Mediator, men når ikke Kalder.

Forretningsmæssigt svar, sker som en tilsvarende overførsel fra Udstiller til Kalder. Også her gælder, at overførsel af svar sker ved overførsel fra Udstiller til Mediator af et svarpayload efterfulgt af overførsel af en triggerfil.

Ansvarsoverdragelsen fra Kalder til Udstiller kan afsluttes når det asynkrone svar når tilbage til Kalder.

1.1.2.3 Pull Svar

I denne variant består konversationen af et initierende synkront kald, med Kalder som initiativtager, samt et varierende antal kald, også initieret af Kalder, til at forespørge efter et svar.



Alle de involverede kald indgår i den samme konversation.

Ansvarsoverdragelsen er gennemført, når Kalder får et positive svar på svarforespørgslen.

1.1.3 Asynkrone kald uden korrelation

Dette er en use case, der er set f.eks. i ØiR-projektet. Her har man et antal kald med et antal informationer i hver payload til modtagersystemerne. Disse svarer senere tilbage med resultatet af behandlingen af informationerne, men ikke i svar der relaterer sig til et specifikt kald. Derimod kombineres der informationer fra forskellige kald i svarene.

I denne use case bryder man princippet om en lineær sammenhæng mellem kald og svar, og det vil sige at sige, at kald og svar reelt ikke indgår i den samme konversation. Sporing af de enkelte stykker information håndteres derfor forretningspecifikt i opbygningen af payloads forretningsindhold, og er derfor ikke omfattet af det generiske transaktionsspor.

2 Transaktioner og fejlbehandling

Dette afsnit beskriver hvordan man behandler transaktionsspor og fejlmelding i en konversation.

”Kalder” er en betegnelse for den part der initierer en konversation. Kalder vil ofte svare til ”anvendelsesystem”.

”Udstiller” er en betegnelse for den endelige modtager af kaldet, altså den der tilbyder en service. Udstiller vil ofte svare til ”kildesystem”.

”Mediator” er en samlebetegnelse, der dækker over formidler af kald. Kald kan være synkrone webservices, asynkrone kald, filoverførsler, beskedformidling. Mediator er således eksempelvis Serviceplatformen, Beskedfordeler eller SFTP-server.

2.1 Transaktionsspor og RequestId

Transaktionssporet har den overordnede hensigt at give en unik identifikation af en simpel konversation mellem to systemer, så det er muligt at styre og tilknytte fejlhåndtering unikt for hver enkelt konversationen.

Transaktionsspor identificeres ved brug af et transaktionsid. Det er kalder der udsteder transaktionsid, og da transaktionsid er en identifier for konversationen, så skal denne være unik for hvert enkelt konversation. Dette suppleres med en transaktionstid, som udtrykker kaldtidspunktet set fra kalders perspektiv.

Konceptet transaktionsspor er tænkt til anvendelse både ved webservicekald, men også i forbindelse med andre kaldsparadigmer, f.eks. overførsel af filer. Der menes således konversationer i bred forstand.

Transaktionssporet passerer som grundregel videre igennem til udstiller, således at transaktionssporet er komplet fra kalder til udstiller. Transaktionssporet logges af kalder, udstiller og mediator ved både kald og svar.

De enkelte kald og svar i konversationen udstyres med unikke RequestId (se nedenfor).

2.1.1 Sammenhæng mellem kald og svar

Forudsætningen for at etablere et sådant transaktionsspor er, at der eksisterer en direkte lineær sammenhæng mellem et kald og et svar, forstået således, at en instans af et svar i sin helhed er et svar på netop ét specifikt kald.

Der anvendes i KOMBIT også konversationer hvor en instans af et kald og en instans af et svar ikke kan korreleres lineært.

Nedenfor beskrives de mest almindelige situationer, og hvordan transaktionssporet skal håndteres i hver enkelt situation.

2.1.2 Synkrone kald

Ved simple synkrone kald vil ethvert svar hænge sammen med et synkront kald. Transaktionssporet etableres ved at give hvert synkront kald et unikt transaktionsid, der skal returneres til kalder i det synkrone svar.

2.1.3 Asynkrone kald med korrelation

Ved asynkrone kald der er korellerede betragter man kaldet og dets asynkrone svar som tilhørende den samme konversation, og man vil derfor knytte dem sammen med et fælles transaktionsid og transaktionstid. Transaktionstid refererer altid til det oprindelige kaldstidspunkt.

2.1.3.1 Push svar

Da asynkrone kald og svar betragtes som en del af den samme konversation, så skal der anvendes samme transaktionsid i begge de involverede synkrone kommunikationer. Det har den konsekvens, at transaktionsid skal være kendt af mediatoren (f.eks. Serviceplatformen) så kaldets transaktionsid kan returneres i det efterfølgende asynkrone svar.

Dette gælder også i situationer hvor kald og svar sker over forskellige protokoller.

2.1.3.2 Pull Svar

Da asynkrone kald og svar betragtes som en del af den samme konversation, så skal der anvendes samme transaktionsid i de involverede synkrone kommunikationer, dvs. der anvendes samme transaktionsid i både de initierende kald og alle følgende forespørgsler efter svar. Brugen af et transaktionsid i forespørgsel efter et svar indikerer, at man forespørger efter et svar på det kald der havde samme transaktionsid.

2.1.4 Asynkrone kald uden korrelation

Kald og svar, indgår ikke direkte i den samme konversation. Transaktionsid er derfor reduceret til kun at spore de enkelte informationsudvekslinger, og der anvendes således ét transaktionsid i kaldet, som udstedes af kalder, og et andet i svaret som udstedes af den part der svarer, da dette betragtes som en anden konversation.

Det er selvfølgelig muligt at benytte kaldets transaktionsid til at tagge de enkelte stykker information, men dette skal altså håndteres af payloaden selv, og er ikke håndteret af det generiske transaktionsspor.

2.1.5 Transaktionsid og -tid

Transaktionsid er altid koblet med en transaktionstid, der ligesom transaktionsid udstedes af kalder. Transaktionstid påstemples kaldstidspunktet, sådan som kalder ser det. Transaktionsid og -tid logges af kalder, udstiller og mediator. Transaktionstid er sat på, da det gør det nemmere af finde kaldet i loggen ud fra kalders information, og det giver mulighed for at estimere forskelle i systemtiderne hos kalder og udstiller.

2.2 Brug af RequestId

Det kan være nødvendigt at styre de enkelte kald og/eller svar som indgår i en konversation. For eksempel kan det være nødvendigt at registrere hvor mange gange man har forsøgt at foretage det samme kald i den samme konversation. I den situation er det ikke tilstrækkeligt at udstyre kaldet med et TransaktionsId. Man har derfor også muligheden for at påtrykke hvert unikt kald/kaldssforsøg med et unikt RequestId, så hvis et kald for eksempel timer ud og man gentager kaldet, så kan man adskille de enkelte forsøg med RequestId, selvom de deler TransaktionsId.

Af hensyn til bagudkompatibilitet er RequestId erklæret som et optionelt felt, men ved implementering af nye løsninger er det et krævet felt.

RequestId er en UUID

2.2.1 RequestId i synkrone kald

Synkrone kald er for eksempel simple webservicekald eller aflevering på en kø. Her udstyres hvert unikke kald med en RequestId. Ved timeout og efterfølgende genkald udstyres hvert genkald med sit eget unikke RequestId.

2.2.2 RequestId i asynkrone kald

Princippet er det samme som for synkrone kald. Det/de asynkrone kald udstyres med unikke RequestId, men svar – både efter push- og pull-modellen – udstyres med et andet unikt RequestId.

Når asynkront svar formidles med pull, er det oprindeligt kalder der udsteder RequestId til sit pull.

Når asynkront svar formidles med push, er det svarende system der udsteder RequestId idet det push'er.

2.3 Tværgående transaktionsspor

I forbindelse med afvikling af længere processer i f.eks. et system, kan der være behov for at vedligeholde et såkaldt tværgående transaktionsspor, der holder samling på alle handlinger der foretages i hele procesforløbet. Hvis en proces for eksempel beskriver et sagsbehandlingsforløb, så vil et tværgående transaktionsspor således være en teknisk metode til at holde styr på alle skridt foretaget i sagsbehandlingen.

Afvikling af processer i et system vil typisk omfatte konversationer med andre systemer, som derfor også kommer til at være en del af den samlede proces. Disse systemer kan tilmed igangsætte underprocesser der igen har konversationer med yderligere systemer osv.

Det bliver derfor vigtigt, at man kan opsamle og spore det tværgående transaktionsspor på tværs af systemer og efter et veldefineret mønster. Dette mønster beskrives her.

2.3.1 Tværgående transaktionsidentifiser

Transaktionsidentifiser for en tværgående transaktion udstedes på samme måde som en almindelig transaktionsid. Dens væsentligste egenskab er, at den skal være unik for den samlede transaktion. Dette betyder også, at den tværgående transaktionsid ikke kan være den samme som transaktionsid for en konversation, da disse altid skal være unikke for den enkelte konversation.

Men samtidig er det hensigtsmæssigt hvis der er en genkendelig sammenhæng mellem den tværgående transaktionsid og de forskellige transaktionsid'er for de enkelte konversationer. Og det vil være yderligere hensigtsmæssigt, hvis de enkelte konversationer i processen kan ordnes i forhold til hinanden. Mønsteret for de to transaktionsidentifiers sammenhæng er derfor, at den enkelte konversations transaktionsid skal dannes ud fra det tværgående transaktionsid, med en dot-notation på følgende måde:

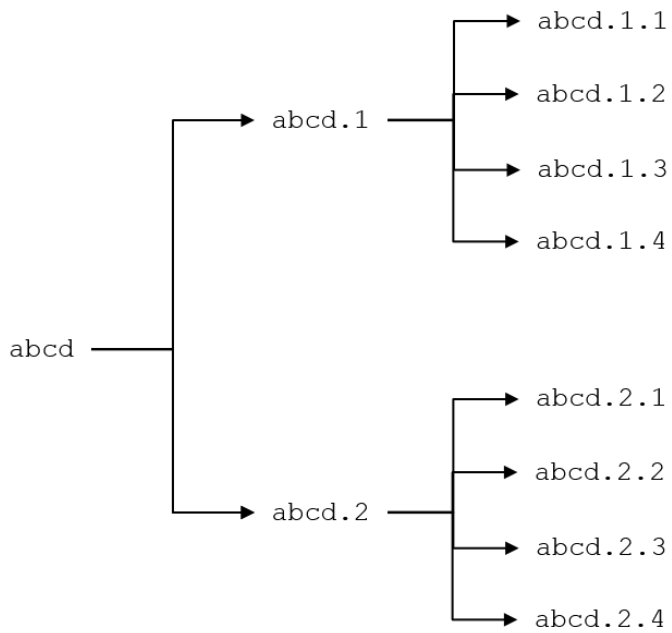
Lad os sige, at det tværgående transaktionsspor får transaktionsid "abcd". Alle konversationer der baseres på dette transaktionsspor vil have "abcd" som basis for transaktionsid.

Transaktionsid for første konversation der initieres i transaktionssporet "abcd", vil få transaktionsid "abcd.1". De følgende kald får transaktionsid "abcd.2", "abcd.3", "abcd.4"... og så fremdeles.

Som nævnt ovenfor kan en konversation i sig selv igangsætte en selvstændig proces i det kaldte system, og denne proces kan initiere selvstændige konversationer. I denne situation anvender man transaktionsid fra den oprindelige konversation som et tværgående transaktionsid efter samme mønster, og det betyder, at videre konversationer i processen får transaktionsid ved yderligere anvendelse af dot-notation.

Hvis vi som eksempel har en konversation med transaktionsid "abcd.2", som igangsætter en proces med egne konversationer, så vil disse konversationer få transaktionsid "abcd.2.1", "abcd.2.2", "abcd.2.3"... og så fremdeles.

Kaldstræet kommer derved til at se ud som for eksempel:



Med denne konvention for dannelse af transaktionsid fra tværgående transaktionsid bliver det også muligt at sammenstille og ordne det samlede flow ud fra transaktionsid'er fra konversationer. Eksemplet ovenfor giver:

abcd
abcd.1
abcd.1.1
abcd.1.2
abcd.1.3
abcd.1.4
abcd.2.1
abcd.2.2
abcd.2.3
abcd.2.4

2.4 Kaldskontekst for SOAP/XML Webservices

Transaktionssporet er placeret i en formel kaldskontekst, der placeres i toppen af payload i en struktur der hedder HovedOplysninger. Udover transaktionsid og –tid indeholder kaldskonteksten et antal optionelle felter der understøtter KOMBITs standard for kontekst og identifikation af kalder.

2.4.1 HovedOplysninger

For SOAP/XML webservices håndteres kaldskonteksten med den generiske struktur ”HovedOplysninger”, der optræder som den øverste struktur i et givent kaldspayload, lige under payloadens top-niveauelement.

```
<kombit2017:HentDebitorkonto_I
  xmlns:kombit2017="http://www.kombit.dk/2017/01/01/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:kontekst="http://kombit.dk/xml/schemas/kontekst/2017/01/01/"
  <kontekst:HovedOplysninger>
  ...
  </kontekst:HovedOplysninger>
  ...
</kombit2017:HentDebitorkonto_I>
```

Følgende er et eksempel på en komplet HovedOplysninger-sektion:

```
<kontekst:HovedOplysninger
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:kontekst="http://kombit.dk/xml/schemas/kontekst/2017/01/01/"
  <kontekst:TransaktionsId>d9b021ed-0881-4b57-9a66-3c1820e7e37f</kontekst:TransaktionsId>
  <kontekst:TransaktionsTid>2001-12-17T09:30:47Z</kontekst:TransaktionsTid>
  <kontekst:RequestId>187fe7d5-4b81-4429-b5ee-72dc190bc95a</kontekst:RequestId>
  <kontekst:OnBehalfOfUser>a</kontekst:OnBehalfOfUser>
  <kontekst:CallersServiceCallIdentifier>a</kontekst:CallersServiceCallIdentifier>
  <kontekst:AccountingInfo>a</kontekst:AccountingInfo>
  <kontekst:AuthorityContext>
    <kontekst:MunicipalityCVR>12345678</kontekst:MunicipalityCVR>
  </kontekst:AuthorityContext>
  <kontekst:Rute>
    <kontekst:AfsenderOrganisation>12345678</kontekst:AfsenderOrganisation>
    <kontekst:AfsenderItSystemInstans>ee8ed739-2af6-4b8b-9bc6-
73995240f9df</kontekst:AfsenderItSystemInstans>
    <kontekst:ModtagerOrganisation>87654321</kontekst:ModtagerOrganisation>
    <kontekst:ModtagerItSystemInstans>842b6355-2879-43d0-9903-
b09ef4501ee7</kontekst:ModtagerItSystemInstans>
  </kontekst:Rute>
  <kontekst:Processing>
    <auto-generated_for_wildcard/>
  </kontekst:Processing>
  <kontekst:Processing>
    <parasoft:svar namespace="https://www.parasoft.com/">svar1</parasoft:svar>
  </kontekst:Processing>
</kontekst:HovedOplysninger>
```

XML-schema for HovedOplysninger-strukturen vedligeholdes i KOMBIT subversion versionskontrol, og den aktuelle version kan findes under:

<http://kombitsvn01.kombit.local/svn/kombit/tool/xsdgenerator/trunk/customize/context/HovedOplysningerType.xsd>

2.4.2 Transaktionsspor

Selve transaktionssporet består af elementerne

```
<kontekst:TransaktionsId> og  
<kontekst:TransaktionsTid>
```

- TransaktionsId vil typisk være udfyldt med en UUID, men andre identifiere er formelt også tilladt. Det centrale er, at feltet er unikt på tværs af mange kald over lang tid. Ved asynkrone kaldsmønstre bruges samme TransaktionsId i kald og svar. Når man anvender tværgående transaktionsspor med dot-notation, så er det den dot-noterede transaktionsid for konversationen der placeres i TransaktionsId-feltet.
- TransaktionsTid er en xs:dateTime, og skal udfyldes med kaldstidspunktet af kalder. Ved svar - synkrone såvel som asynkrone – bruges samme TransaktionsTid som ved kaldet.

Derudover har hvert kald et

```
<kontekst:RequestId>
```

- RequestId er en version 4 UUID, og skal udfyldes med en unik værdi for hvert kaldsforsøg, også under brug af samme TransaktionsId. RequestId skal anvendes i alle nye implementeringer

2.4.3 Kaldskontekstelementer

Dette består af elementerne

```
<kontekst:OnBehalfOfUser>  
<kontekst:CallersServiceCallIdentifier>  
<kontekst:AccountingInfo>  
<kontekst:AuthorityContext>  
<kontekst:Rute>  
...  
</kontekst:Rute>
```

Reglerne for anvendelse af disse felter svarer til de regler der hidtil har været gældende, bortset fra, at de nu er placeret i HovedOplysninger-strukturen i namespace "<http://kombit.dk/xml/schemas/kontekst/2017/01/01/>".

2.4.4 Rute

Rute-strukturen er opbygget således:

```
<kontekst:Rute>  
  <kontekst:AfsenderOrganisation>12345678</kontekst:AfsenderOrganisation>  
  <kontekst:AfsenderItSystemInstans>ee8ed739-2af6-4b8b-9bc6-  
73995240f9df</kontekst:AfsenderItSystemInstans>  
  <kontekst:ModtagerOrganisation>87654321</kontekst:ModtagerOrganisation>  
</kontekst:Rute>
```

Rute er en optionelt struktur, der gør det muligt, at indikere kalder og modtager af kaldet som juridiske enheder/organisationer, samt involverede it-systeminstanser. Strukturen skal facilitere, at formidlersystemerne kan opsætte rutningsregler.

AfsenderOrganisation skal ved certifikat-baseret kommunikation matche værdien af feltet *AuthorityContext*.

AfsenderOrganisation skal ved SAML-baseret kommunikation matche værdien af SAML-tokens attribut "*dk:gov:saml:attribute:CvrNumberIdentifier*".

ModtagerOrganisation skal ved SAML-baseret kommunikation kunne henføres til Scope i SAML-Tokenets PrivilegeGroup.

AfsenderItSystemInstans skal ved SAML-baseret kommunikation kunne henføres til SAML-token subject i Assertion. AfsenderItSystemInstans skal være det af STS Administration provisionerede UUID for brugersystemet, der er tilknyttet certificatatet brugt til kald og som token subject.

ModtagerItSystemInstans bruges i forbindelse med EXPLICIT routing, og bestemmer det af STS Administration provisionerede UUID for modtagende serviceudstillers It-systeminstans.

2.4.5 Processingelementer

Det er nul til mange elementer af typen:

```
<kontekst:Processing>
```

Hensigten med disse felter er, at gøre det muligt, at transportere vilkårlige signaler til modtagerens processing. Dette er tænkt som en mulighed for f.eks. at instruere teststubbe i forventet opførsel. Disse elementer skal som hovedregel ignoreres.

2.5 Kaldskontekst for RESTful web APIs

Transaktionssporet er placeret som en del af http headerinformationen i kaldet. Udover transaktionsid og – tid indeholder kaldskonteksten et antal optionelle felter, der understøtter KOMBITs standard for kontekst og identifikation af kalder.

2.5.1 HTTP-Header

For RESTful web API services håndteres kaldskonteksten med et antal KOMBIT-definerede HTTP headerfelter, der er placeret side om side med de headerfelter der formidles.

Følgende er et eksempel på en komplet HTTP Header:

```
GET /services/DUPLA/%C3%85rsopg%C3%B8relse
x-TransaktionsId: d9b021ed-0881-4b57-9a66-3c1820e7e37f
x-TransaktionsTid: 2001-12-17T09:30:47Z
x-RequestId: 187fe7d5-4b81-4429-b5ee-72dc190bc95a
x-OnBehalfOfUser: Greve Kommune
x-Rute-AfsenderOrganisation: 12345678
x-Rute-AfsenderItSystemInstans: ee8ed739-2af6-4b8b-9bc6-73995240f9df
x-Rute-ModtagerOrganisation: 87654321
x-Rute-ModtagerItSystemInstans: 842b6355-2879-43d0-9903-b09ef4501ee7
x-Processing: svar1
Accept: */*
Host: prod.serviceplatformen.dk
accept-encoding: gzip, deflate
HTTP/1.1 200
status: 200
Content-Length: 29
Content-Type: application/json
```

Specifikationerne for HTTP Headers er indarbejdet i OpenAPI 3.0 specifikationen for den udstillede service på Serviceplatformen. Det er ikke muligt med OpenAPI 3.0 at isolere alle generelle aspekter af mønsteret i en separat file, hvorfor de for nærværende er indarbejdet i hver enkelt specifikation.

2.5.2 Transaktionsspor

Selve transaktionssporet består af HTTP Headers:

```
x-TransaktionsId  
x-TransaktionsTid
```

Disse er defineret således i OpenAPI:

```
"parameters": {  
  "x-TransaktionsId": {  
    "name": "x-TransaktionsId",  
    "in": "header",  
    "required": true,  
    "description": "Unik identifier for konversation. Når kald og svar hænger sammen som dele  
af samme konversation, bruges samme TransaktionsId i både kald og svar. Når et kald passerer videre  
til eller fra et undersystem, passerer TransaktionsId også videre uændret.",  
    "schema": {  
      "type": "string",  
      "pattern": "^[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[4][0-9A-Fa-f]{3}-[0-9A-Fa-f]{4}-[0-9A-Fa-  
f]{12}\\. [0-9]+* $"  
    }  
  },  
  "x-TransaktionsTid": {  
    "name": "x-TransaktionsTid",  
    "in": "header",  
    "required": true,  
    "description": "Tidsstempel der påsættes af oprindelig kalder, som indikering af  
kaldstidspunktet som det opfattes af kalder. Når et kald passerer videre til eller fra et  
undersystem, passerer TransaktionsTid også videre uændret.",  
    "schema": {  
      "type": "string",  
      "format": "xs:dateTime",  
      "pattern": "^ $"  
    }  
  }  
},
```

- TransaktionsId vil typisk være udfyldt med en UUID, men andre identifiere er formelt også tilladt. Det centrale er, at feltet er unikt på tværs af mange kald over lang tid. Ved asynkrone kaldsmønstre bruges samme TransaktionsId i kald og svar. Når man anvender tværgående transaktionsspor med dot-notation, så er det den dot-noterede transaktionsid for konversationen der placeres i TransaktionsId-feltet.
- TransaktionsTid er en xs:dateTime, og skal udfyldes med kaldstidspunktet af kalder. Ved svar - synkrone såvel som asynkrone – bruges samme TransaktionsTid som ved kaldet.

2.5.3 RequestId

RequestId består af http header:

```
x-RequestId
```

og er defineret således i OpenAPI:

```
"x-RequestId": {  
  "name": "x-RequestId",  
  "in": "header",  
  "required": true,  
  "description": "Unik identifier for kald.",  
  "schema": {  
    "type": "string",
```

```

        "pattern": "^[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[4][0-9A-Fa-f]{3}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12}$"
      }
    },

```

2.5.4 Kaldskontekstelementer

Dette består af elementet

x-OnBehalfOfUser

(SOAP/XML-specifikationernes CallersServiceCallIdentifier, AccountingInfo, AuthorityContext er ikke defineret for RESTful web API, da disse modeller er under udfasning).

Feltet er defineret således i OpenAPI:

```

"parameters": {
  ...
  "x-OnBehalfOfUser": {
    "name": "x-OnBehalfOfUser",
    "in": "header",
    "required": false,
    "description": "Identifikation af en bruger i det kaldende system",
    "schema": {
      "type": "string",
      "pattern": "^[.]{0,256}$"
    }
  }
},

```

Reglerne for anvendelse af x-OnBehalfOfUser svarer til de regler der hidtil har været gældende i KOMBIT.

2.5.5 Rute

Denne består af elementerne:

x-Rute-AfsenderOrganisation
 x-Rute-AfsenderItSystemInstans
 x-Rute-ModtagerOrganisation
 x-Rute-ModtagerItSystemInstans

Felterne er i OpenAPI defineret således:

```

"parameters": {
  ...
  "x-Rute-AfsenderOrganisation": {
    "name": "x-Rute-AfsenderOrganisation",
    "in": "header",
    "required": false,
    "description": "Identificerer kaldende organisation.",
    "schema": {
      "type": "string",
      "pattern": "^[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]$"
    }
  },
  "x-Rute-AfsenderItSystemInstans": {
    "name": "x-Rute-AfsenderItSystemInstans",
    "in": "header",
    "required": false,
    "description": "En UUID der identificerer kaldende it-systeminstans, indsat af kalder. UUID refererer til it-systeminstans fra KOMBIT Organisation støttesystem.",
    "schema": {
      "type": "string",
      "format": "xs:dateTime",
      "pattern": "^[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[4][0-9A-Fa-f]{3}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12}$"
    }
  }
},

```

```

    },
    "x-Rute-ModtagerOrganisation": {
      "name": "x-Rute-ModtagerOrganisation",
      "in": "header",
      "required": false,
      "description": "Identificerer modtagende organisation.",
      "schema": {
        "type": "string",
        "pattern": "^[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]$"
      }
    },
    "x-Rute-ModtagerItSystemInstans": {
      "name": "x-Rute-ModtagerItSystemInstans",
      "in": "header",
      "required": false,
      "description": "En UUID der identificerer kaldte it-systeminstans, indsat af kalder. UUID refererer til it-systeminstans fra KOMBIT Organisation støttesystem.",
      "schema": {
        "type": "string",
        "format": "xs:dateTime",
        "pattern": "^[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[4][0-9A-Fa-f]{3}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12}$"
      }
    },
  },
}

```

Rutefelterne gør det muligt, at indikere kalder og modtager af kaldet som juridiske enheder/organisationer, samt involverede it-systeminstanser. Felterne skal facilitere, at formidlersystemerne kan opsætte rutningsregler.

Felterne er optionelle, men hvis et af felterne er til stede skal flere andre af felterne også være det. Dette skal valideres af mediatoren. Da OpenAPI mangler den syntaktiske kontrol af struktur man kender fra XSD, så er samtidig forekomst/samtidigt fravær af rutefelterne, som skal valideres af mediator, beskrevet her i tabelform:

HTTP Headerfelt	Uden ruteinformation	Med ruteinformation
x-Rute-AfsenderOrganisation	-	+
x-Rute-AfsenderItSystemInstans	-	+
x-Rute-ModtagerOrganisation	-	+
x-Rute-ModtagerItSystemInstans	-	

Ved brug af rute gælder følgende, idet det antages, at service er kaldt ved brug af KOMBITs implementering af OIOWS Rest profil v1.0 ([]):

AfsenderOrganisation skal ved SAML-baseret kommunikation matche værdien af attribut *"dk:gov:saml:attribute:CvrNumberIdentifjer"* i det SAML-token der er anvendt til at skaffe Access Token.

ModtagerOrganisation skal kunne henføres til Scope i SAML-Tokenets PrivilegeGroup.

AfsenderItSystemInstans skal kunne henføres til SAML-token subject i Assertion. AfsenderItSystemInstans skal være det af STS Administration provisionerede UUID for brugersystemet, der er tilknyttet certifikatet brugt til kald og som token subject.

ModtagerItSystemInstans bruges i forbindelse med EXPLICIT routing, og bestemmer det af STS Administration provisionerede UUID for modtagende serviceudstillers It-systeminstans.

2.5.6 Processingelementer

Dette er nul til mange HTTP headerfelter af typen:

x-Processing

Feltet er defineret således i OpenAPI:

```
"parameters": {  
  ...  
  "x-Processing": { # Kan nok udelades, idet man under alle omstændigheder kan opdigte og  
    tilføje sine egne headers. Skal i så fald beskrives i Fjl.b. og Ho.opl. dokument.  
    "name": "x-Processing",  
    "in": "header",  
    "required": false,  
    "description": "Mulighed for at passere yderligere kontekstuelle variable, f.eks.  
instrukser til teststubbe.",  
    "schema": {  
      "type": "string",  
    }  
  }  
}
```

Hensigten med disse felter er, at gøre det muligt, at transportere vilkårlige signaler til modtagerens processering. Dette er tænkt som en mulighed for f.eks. at instruere teststubbe i forventet opførsel. Disse elementer skal som hovedregel ignoreres.

2.6 Fejlhåndtering for SOAP/XML Webservices

Her beskrives hvordan man håndterer fejlsituationer i kald med generiske transaktionsspor.

2.6.1 Transaktionsspor

Fejl og Advis ("warnings") håndteres og sendes tilbage til kalder i tilknytning til transaktionssporet. Transaktionssporet givet i kaldet returneres altid til kalder, og enhver fejl eller advis der kan opfanges af programmet - og dette omfatter maskinelt processerbare exceptions - skal omformes til en formel fejlstruktur, der sendes tilbage sammen med transaktionssporet, i en "SvarReaktion".

Af hensyn til bagudkompatibilitet er det vigtigt, at der kun sendes en RequestId med tilbage hvis der er modtaget en i kaldet.

Der kan være multiple SvarReaktion i en enkelt payload. Det er hensigten, at fejl fra udstillersystemer kan sendes videre tilbage til kalder, eventuelt suppleret med fejl fra mediator (f.eks. Serviceplatform) hvis det er nødvendigt. Hver fejl indeholder en indikation af hvor fejlserien stammer fra igennem feltet Kildeld, og derved kan forskellige systemer udstede egne fejlserier uden risiko for nummerkollision i Fejld/Advisld. Dermed er det muligt at lægge sæt af fejl sammen i et svar.

Når der opstår en fejl der er så alvorlig, at systemet ikke er i stand til at danne en forretningsmæssig svarpayload, så skal denne fejl også beskrives i en HovedOplysningerSvar fejlstruktur, og returneres inden i svarpayloadens topniveau, uden resten af svarpayload. Af samme grund vil alle topniveauelementer der følger HovedOplysningerSvar være gjort optionelle.

2.6.2 HovedOplysningerSvar

HovedOplysningerSvar optræder som den øverste struktur i svarpayload, lige under payloadens topniveauelement. Følgende er et eksempel på en komplet HovedOplysningerSvar-sektion:

```

<kontekst:HovedOplysningerSvar
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:n1="http://www.altova.com/samplexml/other-namespace"
  xmlns:kontekst="http://kombit.dk/xml/schemas/kontekst/2017/01/01/">
<kontekst:TransaktionsId>d9b021ed-0881-4b57-9a66-3c1820e7e37f</kontekst:TransaktionsId>
<kontekst:TransaktionsTid>2001-12-17T09:30:47Z</kontekst:TransaktionsTid>
<kontekst:RequestId>18077dae-e205-4594-87cf-5da63ec2dd3e</kontekst:RequestId>
<kontekst:SvarReaktion>
  <kontekst:Fejl>
    <kontekst:FejlId>1003</kontekst:FejlId>
    <kontekst:FejlTekst>Bad xs:dataType</kontekst:FejlTekst>
    <kontekst:KildeId>57112c54-d398-4e46-8d31-a0dd819d384d</kontekst:KildeId>
    <kontekst:Identifikation>
      <n1:auto-generated_for_wildcard></n1:auto-generated_for_wildcard>
    </kontekst:Identifikation>
  </kontekst:Fejl>
</kontekst:SvarReaktion>
</kontekst:SvarReaktion>
  <kontekst:Advis>
    <kontekst:AdvisId>2002</kontekst:AdvisId>
    <kontekst:AdvisTekst>CVRNummer eksisterer ikke</kontekst:AdvisTekst>
    <kontekst:KildeId>57112c54-d398-4e46-8d31-a0dd819d384d </kontekst:KildeId>
    <kontekst:Identifikation>
      <ns2:CVRNummer>12345678</ns2:CVRNummer>
    </kontekst:Identifikation>
    <kontekst:Identifikation>
      <n1:auto-generated_for_wildcard/>
    </kontekst:Identifikation>
  </kontekst:Advis>
</kontekst:SvarReaktion>
</kontekst:HovedOplysningerSvar>

```

XML-schema for HovedOplysningerSvar-strukturen vedligeholdes i KOMBIT svn versionskontrol, og den aktuelle version kan findes under:

<http://kombitsvn01.kombit.local/svn/kombit/tool/xsdgenerator/trunk/customize/context/HovedOplysningerSvarType.xsd>

2.6.3 Struktur af en fejl

Hver fejl eller advis rapporteres i et SvarReaktion-tag. Der er netop en fejl eller advis i hver forekomst af SvarReaktion. Fejl og advis er bygget op af følgende dele:

2.6.4 FejlId og AdvisId

FejlId er fejlens unikke identifikation indenfor det domæne der identificeres af FejlKildeld. Hvert domæne har kontrol over udstedelsen af sine egne fejlidserier. I nogen situationer er fejlidserier forsøgt harmoniseret i en klassifikation, og i den situation er det klassifikationen der udgør FejlKildeld for fejlnummerserien.

```
<kontekst:FejlId>1003</kontekst:FejlId>
```

AdvisId følger samme logik og samme struktur.

På Serviceplatformen er der aftalt et antal fejlId som udstedes af Serviceplatformen selv, i forskellige kendte fejlsituationer. Disse vil kunne findes udstillet på serviceplatformen.dk. Et eksempel på en FejlId fra Serviceplatformen kunne være:

```
<kontekst:FejlId>WrongCertificate</kontekst:FejlId>
```

2.6.5 FejlTekst og AdvisTekst

FejlTekst indeholder en beskrivende tekst for fejlen. FejlTeksten har til formål at beskrive fejlen i en logfil, sammen med fejlnummeret. Der skal typisk defineres fornuftige fejltekster for hver forventet fejlnummer. Uventede fejl og exceptions der ikke gribes kontrolleret, kan returnere (dele af) stack trace i FejlTekst hvis der ikke er defineret andet, men det skal undgås, da det ikke er læseligt.

```
<kontekst:FejlTekst>Bad xs:dataType</kontekst:FejlTekst>
```

AdvisTekst følger samme logik og samme struktur.

```
<kontekst:AdvisTekst>CPRId eksisterer ikke</kontekst:AdvisTekst>
```

Et eksempel på en FejlTekst fra Serviceplatformen kunne være:

```
<kontekst:FejlTekst>Client certificate did not match the subject certificate of the security token</kontekst:FejlTekst>
```

2.6.6 Kildeld

Identificerer det logiske system der udsteder fejlen. Man skal altså sikre, at FejlId+Kildeld er unik.

```
<kontekst:KildeId>ØkonomisystemXYZ</kontekst:KildeId>
```

Kildeld er således en slags "namespace" for det logiske system, og anvendes både i fejl og i advis. Kildeld bør være den UUID IT-systeminstansen er oprettet med i det Fælleskommunale Administrationsmodul.

Når Serviceplatformen udsteder en fejl, vil det være Serviceplatformen selv der optræder i den udstedte fejl som Kildeld. Det logiske system "Serviceplatformen" har Kildeld:

```
<kontekst:KildeId>Serviceplatformen</kontekst:KildeId>
```

2.6.7 Identifikation

Enhver fejl kan, men skal ikke, have et antal Identifikationer. Dette gør det muligt at passere felter med værdier tilbage til kalder, som præciserer fejlens forretningsmæssige kontekst. Der er relativt frie rammer for hvordan Identifikation anvendes, da den er afhængig af den forretningsmæssige kontekst for service.

```
<kontekst:Identifikation>  
  <ns2:CVRNummer>12345678</ns2:CVRNummer>  
</kontekst:Identifikation>
```

Den typiske anvendelse af Identifikation er i kald og svar med lister, og lister i lister, hvor hver forekomst i listen kan identificeres med et felt, eller en kombination af felter i forekomsten. Her er det muligt at knytte fejlen til forekomsten ved at citere dette eller disse felter i Identifikation-felter.

Dette gør det også muligt at sende svar på kald med partiel succes, hvor der er en svar-payload. F.eks. hvis der forespørges på en liste med 1000 forekomster, og der er fejl i en forekomst, mens resten går godt, så er det muligt at returnere de 999 forekomster, samt en HovedOplysningerSvar-struktur indeholdende en enkelt fejl, hvor fejlen knyttes til den fejlende forekomst med en eller flere Identifikation-felter.

2.7 Fejlhåndtering for RESTful web APIs

Her beskrives hvordan man håndterer generiske transaktionsspor og fejlsituationer i kald med generiske transaktionsspor.

2.7.1 Transaktionsspor

Fejl og Advis ("warnings") håndteres i tilknytning til transaktionssporet. Transaktionssporet er givet i HTTP Headeren i kaldet, og skal altid returneres til kalder i svaret. Enhver fejl eller advis der kan opfanges af programmet - og dette omfatter maskinelt processerbare exceptions - skal omformes til en formel fejlstruktur, der sendes tilbage i en "SvarReaktion" sammen med transaktionssporet og en HTTP errorcode.

Af hensyn til bagudkompatibilitet er det vigtigt, at der kun sendes en RequestId med tilbage hvis der er modtaget en i kaldet.

Der kan være multiple SvarReaktion i en enkelt payload. Det er hensigten, at fejl fra udstillersystemer kan sendes videre tilbage til kalder, suppleret med en HTTP kode fra mediator (f.eks. Serviceplatform). Hver fejl indeholder en indikation af hvor fejlserien stammer fra igennem feltet Kildeld, og derved kan forskellige systemer udstede egne fejlserier uden risiko for nummerkollision i Fejlld/AdvisId. Dermed er det muligt at lægge sæt af fejl sammen i et svar såfremt det skulle være nødvendigt.

Når der opstår en fejl der er så alvorlig, at systemet ikke er i stand til at danne en forretningsmæssig svarpayload, så skal denne fejl også beskrives i en fejlstruktur, og returneres uden resten af svarpayload.

2.7.2 HTTP Header

Følgende tabel beskriver opbygningen af en HTTP-response fra Mediator:

Response		
GET /<path>		KOMBIT
Status	Response	
200	content	<i>Content fra kildeservice</i>
		SvarReaktion
	headers	<i>Headers fra kildeservice</i>
		x-TransaktionsId
x-RequestId		
500	content	SvarReaktion
	headers	<i>Headers fra kildeservice</i>
		x-TransaktionsId
		x-RequestId
default	content	SvarReaktion
	headers	<i>Headers fra kildeservice</i>
		x-TransaktionsId
		x-RequestId

Specifikationerne for HTTP Headers og SvarReaktion er indarbejdet i OpenAPI 3.0 specifikationen for den udstillede service på Serviceplatformen. Det er ikke muligt med OpenAPI 3.0 at isolere alle generelle aspekter af mønsteret i en separat file, hvorfor de for nærværende er indarbejdet i hver enkelt specifikation.

2.7.3 Struktur af en SvarReaktion

Hver fejl eller advis rapporteres i en forekomst af SvarReaktion. Der er netop en fejl eller advis i hver forekomst af SvarReaktion. SvarReaktion er i OpenAPI 3.0 bygget således op:

```
"schemas": {
  ...
  "HovedoplysningerSvarREST": {
    "type": "array",
    "items": {
      "allOf" : [
        {
          "type": "object",
          "properties": {
            "SvarReaktion": {
              "$ref": "#/components/schemas/SvarReaktion"
            }
          },
          "xml": {
            "prefix": "kontekst"
          }
        },
        { "required": [ "SvarReaktion" ] }
      ]
    }
  },
  "SvarReaktion": {
    "allOf" : [
      {
        "type": "object",
        "properties": {
          "Fejl": {
            "$ref": "#/components/schemas/Fejl"
          },
          "Advis": {
            "$ref": "#/components/schemas/Advis"
          }
        },
        "xml": {
          "prefix": "kontekst"
        }
      },
      {
        "oneOf": [
          { "required": ["Fejl"] },
          { "required": ["Advis"] },
          {
            "not": {
              "anyOf": [
                { "required": ["Fejl"] },
                { "required": ["Advis"] }
              ]
            }
          }
        ]
      }
    ]
  },
  "xml": {
    "prefix": "kontekst",
    "namespace": "http://kombit.dk/xml/schemas/kontekst/2017/01/01/"
  },
  "Fejl": {
    "allOf" : [
      {

```

```

    "type": "object",
    "properties": {
      "FejlId": {
        "type": "string",
        "description": "Unik identifikation for fejlen fra fejludstederen. Sammen med KildeId er
fejlen unik på tværs af systemer.",
        "xml": {
          "prefix": "kontekst"
        }
      },
      "FejlTekst": {
        "type": "string",
        "description": "Beskrivende tekst for fejlen. Udfyldes af fejludsteder.",
        "xml": {
          "prefix": "kontekst"
        }
      },
      "KildeId": {
        "type": "string",
        "description": "Kontekst for fejlen eller advisen. Unik identifikation af udstederen.
Indsættes af fejl- eller advisudsteder selv. Ved viderepassage af en fejl eller advis, passeres
KildeId også videre uændret.",
        "xml": {
          "prefix": "kontekst"
        }
      },
      "Identifikation": {
        "type": "string",
        "description": "Giver mulighed for, at passere variable værdier der beskriver fejl eller
advis detaljeret, og gøre disse variable maskinelt processerbare for modtager.",
        "xml": {
          "prefix": "kontekst"
        }
      },
      "status": {
        "type": "string",
        "description": "HTTP status kode",
        "xml": {
          "prefix": "kontekst"
        }
      }
    }
  },
  {
    "required": [
      "FejlId",
      "FejlTekst"
    ]
  }
],
"xml": {
  "prefix": "kontekst"
}
},
"Advis": {
  "allOf": [
    {
      "type": "object",
      "properties": {
        "AdvisId": {
          "type": "string",
          "description": "Unik identifikation for advisen fra advisudstederen. Sammen med KildeId
er advisen unik på tværs af systemer.",
          "xml": {
            "prefix": "kontekst"
          }
        },
        "AdvisTekst": {
          "type": "string",
          "description": "Beskrivende tekst for advisen. Udfyldes af advisudsteder.",
          "xml": {
            "prefix": "kontekst"
          }
        }
      }
    }
  ]
},

```

```

    "KildeId": {
      "type": "string",
      "description": "Kontekst for fejlen eller advisen. Unik identifikation af udstederen.
Indsættes af fejl- eller advisudsteder selv. Ved viderepassage af en fejl eller advis, passeres
KildeId også videre uændret.",
      "xml": {
        "prefix": "kontekst"
      }
    },
    "Identifikation": {
      "type": "string",
      "description": "Giver mulighed for, at passere variable værdier der beskriver fejl eller
advis detaljeret, og gøre disse variable maskinelt processerbare for modtager.",
      "xml": {
        "prefix": "kontekst"
      }
    },
    "status": {
      "type": "string",
      "description": "HTTP status kode",
      "xml": {
        "prefix": "kontekst"
      }
    }
  },
  {
    "required": [
      "AdvisId",
      "AdvisTekst"
    ]
  },
  "xml": {
    "prefix": "kontekst"
  }
}

```

2.7.4 FejlId og AdvisId

FejlId er fejlens unikke identifikation indenfor det domæne der identificeres af FejlKildeId. Hvert domæne har kontrol over udstedelsen af sine egne fejlidserier. I nogen situationer er fejlidserier forsøgt harmoniseret i en klassifikation, og i den situation er det klassifikationen der udgør FejlKildeId for fejlnummerserien.

AdvisId følger samme logik og samme struktur.

På Serviceplatformen er der aftalt et antal fejlId som udstedes af Serviceplatformen selv, i forskellige kendte fejlsituationer. Disse vil kunne findes udstillet på digitaliseringskataloget.dk.

Et eksempel på en FejlId fra Serviceplatformen kunne være:

```
"FejlId": "InvalidRequest"
```

2.7.5 FejlTekst og AdvisTekst

FejlTekst indeholder en beskrivende tekst for fejlen. FejlTeksten har til formål at beskrive fejlen i en logfil, sammen med fejlnummeret. Der skal typisk defineres fornuftige fejltekster for hver forventet fejlnummer. Uventede fejl og exceptions der ikke griber kontrolleret, kan returnere (dele af) stack trace i FejlTekst hvis der ikke er defineret andet, men det skal undgås, da det ikke er læseligt.

AdvisTekst følger samme logik og samme struktur.

Et eksempel på en FejlTekst fra Serviceplatformen kunne være:

"FejlTekst": "TransaktionsId in HovedOplysninger exceeded Serviceplatformen max length"

2.7.6 Kildeld

Identificerer det logiske system der udsteder fejlen. Man skal altså sikre, at FejlId+Kildeld er unik.

Kildeld er således en slags "namespace" for det logiske system, og anvendes både i fejl og i advis. Kildeld bør være den UUID IT-systeminstansen er oprettet med i det Fælleskommunale Administrationsmodul.

Når Serviceplatformen udsteder en fejl, vil det være Serviceplatformen selv der optræder i den udstedte fejl som Kildeld. Det logiske system "Serviceplatformen" har Kildeld. Eks.:

"KildeId": "Serviceplatformen"

2.7.7 Identifikation

Enhver fejl kan, men skal ikke, have et antal Identifikationer. Dette gør det muligt at passere felter med værdier tilbage til kalder, som præciserer fejlens forretningsmæssige kontekst. Der er relativ frie rammer for hvordan Identifikation anvendes, da den er afhængig af den forretningsmæssige kontekst for service. Eks.:

"Identifikation": "CVRNummer=12345678, IndkomstÅr=2019"

Den typiske anvendelse af Identifikation er i kald og svar med lister, og lister i lister, hvor hver forekomst i listen kan identificeres med et felt, eller en kombination af felter i forekomsten. Her er det muligt at knytte fejlen til forekomsten ved at citere dette eller disse felter i Identifikation-felter.

Dette gør det også muligt at sende svar på kald med partiel succes, hvor der er en svar-payload. F.eks. hvis der forespørges på en liste med 1000 forekomster, og der er fejl i en forekomst, mens resten går godt, så er det muligt at returnere de 999 forekomster, samt en HovedOplysningerSvar-struktur indeholdende en enkelt fejl, hvor fejlen knyttes til den fejlende forekomst med en eller flere Identifikation-felter.

2.7.8 Status

Den centrale fejlrapporteringsmekanisme i RESTful services er HTTP errorcodes. Hvis man derfor er mediator for en service skal man håndtere Udstillers HTTP errorcodes.

Mediator håndterer dette ved at definere en SvarReaktion, og indlejre Udstillers HTTP errorcode i SvarReaktion, felt "status". Denne SvarReaktion returneres af mediator til Anvender som BODY til en HTTP errorcode der er bestemt ud fra nedenstående mapningstabel.

Mapningstabellen tager aktivt stilling til hvordan specifikke HTTP errorcodes skal returneres fra mediator. Generelt bliver fejl fra kilde mappet til enten HTTP fejl 200 eller HTTP fejl 500.

Andre standard HTTP errorcodes der ikke står i tabellen sendes uændret videre til Anvender. Dette er afspejlet i den udstillede OpenAPI specifikation som "default:" response.

HTTP Status Code konvertering		
KOMBIT	Kilde	
HTTP Status Codes af format 3xx fra KILDE konverteres som nedenstående		
200	300	KILDE 300: Multiple Choices. The target resource has more than one representation, each with its own more specific identifier, and information about the alternatives is being

		<p>provided so that the user (or user agent) can select a preferred representation by redirecting its request to one or more of those identifiers.</p> <p>KOMBIT RESTful webservice returnerer 200: OK. The request has succeeded.</p>
500	301	<p>KILDE 301: Moved Permanently. The target resource has been assigned a new permanent URI and any future references to this resource ought to use one of the enclosed URIs.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>
500	302	<p>KILDE 302: Found. The target resource resides temporarily under a different URI. Since the redirection might be altered on occasion, the client ought to continue to use the effective request URI for future requests.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>
200	303	<p>KILDE 303: See Other. The server is redirecting the user agent to a different resource, as indicated by a URI in the Location header field, which is intended to provide an indirect response to the original request.</p> <p>KOMBIT RESTful webservice returnerer 200: OK. The request has succeeded.</p>
500	305	<p>KILDE 305: Use Proxy. Defined in a previous version of this specification and is now deprecated, due to security concerns regarding in-band configuration of a proxy.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>
500	307	<p>KILDE 307: Temporary Redirect. The target resource resides temporarily under a different URI and the user agent MUST NOT change the request method if it performs an automatic redirection to that URI.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>
500	308	<p>KILDE 308: Permanent Redirect. The target resource has been assigned a new permanent URI and any future references to this resource ought to use one of the enclosed URIs.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>
HTTP Status Codes fra KILDE af format 4xx konverteres som nedenstående		
500	412	<p>KILDE 412: Precondition Failed. One or more conditions given in the request header fields evaluated to false when tested on the server.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>
500	414	<p>KILDE 414: Request-URI Too Long. The server is refusing to service the request because the request-target is longer than the server is willing to interpret.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>
500	418	<p>KILDE 418: I'm a teapot. Any attempt to brew coffee with a teapot should result in the error code "418 I'm a teapot". The resulting entity body MAY be short and stout.</p>

		KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.
500	421	KILDE 421: Misdirected Request. The request was directed at a server that is not able to produce a response. This can be sent by a server that is not configured to produce responses for the combination of scheme and authority that are included in the request URI. KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.
500	423	KILDE 423: Locked. The source or destination resource of a method is locked. KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.
500	424	KILDE 424: Failed Dependency. The method could not be performed on the resource because the requested action depended on another action and that action failed. KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.
500	426	KILDE 426: Upgrade Required. The server refuses to perform the request using the current protocol but might be willing to do so after the client upgrades to a different protocol. KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.
500	444	KILDE 444: Connection Closed Without Response. A non-standard status code used to instruct nginx to close the connection without sending a response to the client, most commonly used to deny malicious or malformed requests. KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.
500	451	KILDE 451: Unavailable For Legal Reasons. The server is denying access to the resource as a consequence of a legal demand. KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.
500	499	KILDE 499: Client Closed Request. KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.
HTTP Status Codes fra KILDE af format 5xx konverteres alle til 500		
500		KILDE 500: Internal Server Error. The server encountered an unexpected condition that prevented it from fulfilling the request.
	500	KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.
		KILDE 501: Not implemented. The server does not support the functionality required to fulfill the request.
	501	KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.
	502	KILDE 502: Bad gateway. The server, while acting as a gateway or proxy, received an invalid response from an inbound server it accessed while attempting to fulfill the request.

		<p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>
503	<p>KILDE 503: Services unavailable. The server is currently unable to handle the request due to a temporary overload or scheduled maintenance, which will likely be alleviated after some delay.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>	
504	<p>KILDE 504: Gateway timeout. The server, while acting as a gateway or proxy, did not receive a timely response from an upstream server it needed to access in order to complete the request.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>	
505	<p>KILDE 505: HTTP Version Not Supported. The server does not support, or refuses to support, the major version of HTTP that was used in the request message.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>	
506	<p>KILDE 506: Variant Also Negotiates. The server has an internal configuration error: the chosen variant resource is configured to engage in transparent content negotiation itself, and is therefore not a proper end point in the negotiation process.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>	
507	<p>KILDE 507: Insufficient storage. The method could not be performed on the resource because the server is unable to store the representation needed to successfully complete the request.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>	
508	<p>KILDE 508: Loop detected. The server terminated an operation because it encountered an infinite loop while processing a request with "Depth: infinity". This status indicates that the entire operation failed.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>	
510	<p>KILDE 510: Not Extended. The policy for accessing the resource has not been met in the request. The server should send back all the information necessary for the client to issue an extended request.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error The server does not support the functionality required to fulfill the request.</p>	
511	<p>KILDE 511: Network Authentication Required. The client needs to authenticate to gain network access.</p> <p>KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.</p>	

	599	KILDE 599: Network Connect Timeout Error. KOMBIT RESTful webservice returnerer 500: Internal Server Error. The server does not support the functionality required to fulfill the request.
--	-----	---

Uanset hvilken HTTP errorcode der returneres fra Udstiller, skal den status code som Mediator modtager fra et kald til Kilde returneres til Anvender via SvarReaktion i response. Undtaget herfra er alle koder i HTTP 1XX-serien, som betyder succes.

2.8 Forventninger til kalders håndtering af fejl og transaktionsspor

2.8.1 Kalders håndtering af gentagne kald med samme transaktionsId

Kalder forventes at udstede og anvende et TransaktionsId til netop en konversation. Kalder skal derfor forvente, at gentagne anvendelser af samme TransaktionsId vil give fejlsvar fra udstiller.

2.8.2 Kalder håndtering af fejl i kald

- *Fejl på protokolniveau:*
En fejl på protokolniveau kan kun håndteres af kalder selv. Man vil typisk aftale et antal retries, inden man fejler ét kald, f.eks. kald og 2 retries.
- *soap:faults (når Udstiller benytter SOAP/XML):*
Skal håndteres som fejl på protokolniveau.
- *Alle HTTP-fejl (både SOAP/XML og RESTful service):*
Skal håndteres som fejl på protokolniveau.
- *Timeout:*
Der kan være aftalt et timeout på en service, som hvis det overskrides giver anledning til at der rejses en fejl. Kalder skal håndtere timeout.
- *Retry:*
Der kan være aftalt et antal retry på en service. Lad os sig at man eksempelvis vil foretage et kald og 2 retry inden man opgiver. Hvis man opgiver efter alle retry, håndteres fejlen på samme måde som et timeout. Alle retry skal anvende samme TransaktionsId, for at indikere overfor udstiller, at der er tale om forsøg på kald i samme konversation. Hvert retry skal anvende et nyt unikt RequestId.

(Der kan være andre fejlsituationer end ovenstående).

Fejl vil som udgangspunkt altid nå kalder, enten i payloadens HovedOplysningerSvar-struktur ved SOAP/XML, eller som SvarReaktion-strukturer ved RESTful service. Der aftales standard fejlId'er for disse standard fejl. Alle fejl logges, med FejlId og FejlTekst.

2.9 Forventninger til udstillers håndtering af fejl og transaktionsspor

Det er serviceudstillers ansvar, at validere modtagne kald og reagere hensigtsmæssigt på dem, herunder at håndtere transaktionsspor. Der er to hovedkategorier af udstillere; udstillere der modtager specifikationer af den udstillede service defineret med KOMBITS contract-first-metode, og udstillere der definerer deres egen udstilling. I det sidste tilfælde skal det aftales individuelt per service hvordan udveksling af transaktionsspor sker, hvordan fejl håndteres og hvordan payload opbygges. Det er beskrevet hvordan servicekald indkapsles til udstilling i et senere afsnit, men ellers beskrives denne situation ikke her. Følgende beskriver forventningerne til udstiller af service når specifikationerne modtages fra KOMBITS contract-first.

2.9.1 Serviceudstiller ved ren KOMBIT contract-first

Her forventes det af serviceudstiller, at man ved korellerede kald og svar passerer kaldets transaktionsspor, som det forefindes i HovedOplysninger (ved SOAP/XML) eller HTTP Header (ved RESTful service), tilbage i det, eller de, tilsvarende svar i HovedOplysningerSvar-strukturen (ved SOAP/XML) eller HTTP Header (ved RESTful service). Alle programmatisk håndterbare fejl skal ophæves til kontrollerede fejl som returneres i SvarReaktion-elementer i HovedOplysningerSvar-strukturen/SvarReaktion-struktur. Det aftales mellem KOMBIT og serviceudstiller hvilke fejl udstiller forventer at returnere.

Det forventes desuden, at serviceudstiller logger RequestId i kald.

Hvis kalder kalder med samme TransaktionsId i flere konversationer, skal udstiller afvise af udføre operationen igen med en SvarReaktion i HovedOplysningerSvar (SOAP/XML) eller SvarReaktion+HTTP Errorcode (RESTful service). Kalder kan dog have brug for, at svar-payload fra det oprindelige kald returneres, såfremt svaret aldrig nåede tilbage til kalder. Dette bør løses i designet af servicen, evt ved at udstiller returnerer svar payload sammen med SvarReaktionen i hovedoplysningerSvar eller via en anden service operation med dette formål.

I tilfælde af et korreleret asynkront kald gælder ovenstående det asynkrone kald. De asynkrone svar vil genanvende TransaktionsId, både i push- og pull-varianterne.

Det aftales mellem KOMBIT og udstiller hvilke fejl der udstiller anvender til de forskellige mulige fejl.

2.9.2 Serviceudstiller ved andre integrationer

De fleste integrationer opererer med en opfattelse af transaktionsspor. Det skal i den enkelte integration aftales mellem KOMBIT og udstiller af service, hvad der identificerer transaktionsspor.

2.10 Forventninger til mediators håndtering af fejl og transaktionsspor

Mediator befinder sig i en rolle der lægger sig midt imellem udstiller og kalder af service, idet man udstiller en viderestillet service og kalder et viderestillet kald, men uden at være hverken egentlig udstiller eller kalder.

Når udstiller ikke udstiller en service defineret ved KOMBITs contract-first-fremgangsmåde, så skal kalders transaktionsspor mappes til udstillers native transaktionsspor og tilbage igen af mediator. Det forventes af mediator, at denne mapning kan vedligeholdes og logges, så mapning mellem kald til mediator og kald til kilde kan spores. Det afhænger af det konkrete servicedesign hvorledes dette skal ske i den enkelte situation.

Det forventes desuden, at mediator logger RequestId fra kald. Mediator forventes også at danne og logge et separat unikt RequestId for viderekald til serviceudstiller.

2.10.1 Mediator håndtering af syntaksfejl i kald

Kald fra kalder indeholder HovedOplysninger, som skal fortolkes af mediator, for at muliggøre håndtering af transaktionsspor og kaldskontekst. Hvis fortolkning fejler, dannes en fejl, som returneres til kalder i en SvarReaktion i en HovedOplysningerSvar-struktur.

Andre syntaksfejl håndteres på samme måde, hvis der er aftalt syntaksvalidering af kaldsstruktur.

Det aftales mellem KOMBIT og mediator hvilke fejl der mediator anvender til de forskellige mulige fejl. Ved udstilling af RESTful web API'er skal det fastlægges hvilke HTTP fejlkode der skal returneres.

På Serviceplatformen er der aftalt et antal fejl id som udstedes af Serviceplatformen selv, i forskellige kendte fejlsituationer. Disse vil kunne findes udstillet på serviceplatformen.dk.

2.10.2 Mediator håndtering af systemexceptions i kald

Hvis kald giver anledning til at mediator får en exception, så skal fejlen fanges af mediator og rapporteres tilbage til kalder som en kontrolleret fejl i HovedOplysningerSvar.

Det aftales mellem KOMBIT og mediator hvilke fejl id mediator anvender til de forskellige mulige fejl. Ved udstilling af RESTful web API'er skal det fastlægges hvilke HTTP fejlkode der skal returneres.

På Serviceplatformen er der aftalt et antal fejl id som udstedes af Serviceplatformen selv, i forskellige kendte fejlsituationer. Disse vil kunne findes udstillet på serviceplatformen.dk.

2.10.3 Mediator håndtering af gentagne kald med samme transaktionsid

Som hovedregel er mediators "transparent", forstået sådan, at mediator ikke forholder sig aktivt til, om et kald er en gentagelse eller ej. Håndtering af denne validering sendes videre til udstiller, som forventes at håndtere situationen.

Dog forventes det, at mediator logger kald og svar med Transaktionsid samt Requestid. Serviceplatformen skal dog ikke logge det forretningsmæssige indhold. Alene hovedoplysninger/transaktionsspor logges og personhenførbare oplysninger må ikke logges som del af dette.

2.10.4 Mediator håndtering af fejl i viderekald

Ved viderekald af services til udstiller, for eksempel igennem en mediator såsom Serviceplatformen, kan der opstå fejltilstande i forhold til viderekaldet, som skal håndteres.

- *Fejl på protokolniveau:*
En fejl på protokolniveau skal fanges af mediator og rapporteres videre til kalder som en kontrolleret fejl i HovedOplysningerSvar ved SOAP/XML-kald eller en HTTP errorcode 500 med en SvarReaktion ved udstilling af RESTful web API. Mediator skal tilføje et Identifikation-felt til fejlen, som indeholder det (fejllende) svar der er modtaget fra kilden, så Anvender nemmere kan fejlsøge. Mediator kan tilføje en selvstændig fejl til fejllisten, hvis den modtagne fejl giver en følgeføj i mediator, f.eks. hvis en forventet behandling ikke kan gennemføres.
- *Timeout:*
Der kan være aftalt et timeout på en service, som hvis det overskrides giver anledning til at der rejses en fejl. Fejlen skal fanges af mediator og rapporteres videre til kalder som en kontrolleret fejl i HovedOplysningerSvar ved SOAP/XML-kald eller en HTTP errorcode 500 med en SvarReaktion ved udstilling af RESTful web API.

- *Retry:*
Der kan være aftalt et antal retry på en service. Hvis man opgiver efter alle retry håndteres fejlen på samme måde som et timeout. Alle retry skal anvende samme TransaktionsId, for at indikere overfor udstiller, at der er tale om forsøg på kald i samme konversation.

Der aftales standard fejlider for disse standard fejl. Alle fejl logges, med FejlId og FejlTekst.

2.10.5 Mediator håndtering af fejl i ugyldigt svar

Ved viderekald af services til andre systemer, for eksempel igennem en mediator, såsom Serviceplatformen, kan man modtage payloads fra kaldte system der er strukturelt ulæselige. Disse fejl skal håndteres på samme måde som fejl på protokolniveau, dvs. at mediator rapporterer en fejl videre til kalder som en kontrolleret fejl i HovedOplysningerSvar ved SOAP/XML-kald eller en HTTP errorcode 500 med en SvarReaktion ved udstilling af RESTful web API..

2.10.6 Mediator håndtering af fejl i gyldigt svar

Ved viderekald af services til andre systemer, for eksempel igennem en mediator såsom Serviceplatformen, kan man modtage payloads fra kaldte system der er strukturelt valid, men har HovedOplysningerSvar eller anden fejlstruktur indeholdende fejl. Disse fejl skal passeres videre til kaldersystemet, indeholdt i HovedOplysningerSvar ved SOAP/XML-kald eller som HTTP code 200 med en eller flere SvarReaktioner ved udstilling af RESTful web API. Hvis den modtagne fejl giver en følgefejl i mediator, f.eks. hvis en forventet behandling i mediator ikke kan gennemføres, skal mediator tilføje en selvstændig fejl/SvarReaktion til fejllisten.