Sikkerhedsarkitektur for Enhanced Healthcare Messaging Infrastructure (EHMI) services

Version 01

Versionshistorik			
Version	Forfatter	Dato	Noter
0.1	Lakeside, CHG	23.02.2024	Første udgave
0.1.1	OVI, MedCom	03.04.2024	Justeret første udgave

Indholdsfortegnelse

1 INTRODUKTION

I dette dokument beskrives sikkerhedsarkitekturen for services i [EHMI] (Enhanced Healthcare Messaging Infrastructure), herunder anvendte sikkerhedsprotokoller, akkreditiver og token-formater.

Under EHMI services forstås de services i beskeds-infrastrukturen som ikke direkte indgår i selve punkt til punkt meddelelseskommunikationen, herunder:

- EHMI Delivery Status (EDS) (På dansk: Forsendelsesstatusservice)
- EHMI Adressering Service (EAS) (På dansk: Sundhedsadresseringsservice)
- EHMI Endpoint Register (EER) (På dansk: Postkasseregister)

Denne første udgave af dokumentet har alene til formål at give parterne som skal tilgå EHMI Delivery Status (EDS) et grundlag til estimering af deres integrationsopgave.

Alle afsnit markeret med *<TBD>* bliver udfyldt senere.

Der vil potentielt blive justeret i ordlyden i det videre arbejde, ligesom småjusteringer af det tekniske indhold vil kunne forekomme.

1.1 Publikum

Dokumentet har en teknisk karakter og henvender sig primært til arkitekter og software-udviklere.

Projektledere og andre projektdeltagere kan med fordel også orientere sig i relevante dele af dokumentet.

1.2 Formål

Dokumentet skal i forbindelse med pilotafprøvningen af kommunale prøvesvar kunne danne grundlag for realisering af sikkerhedsmekanismerne i både udviklingen af EHMI services, etablering af en Authorization Server og hos parterne som skal anvende disse services.

1.3 Baggrund

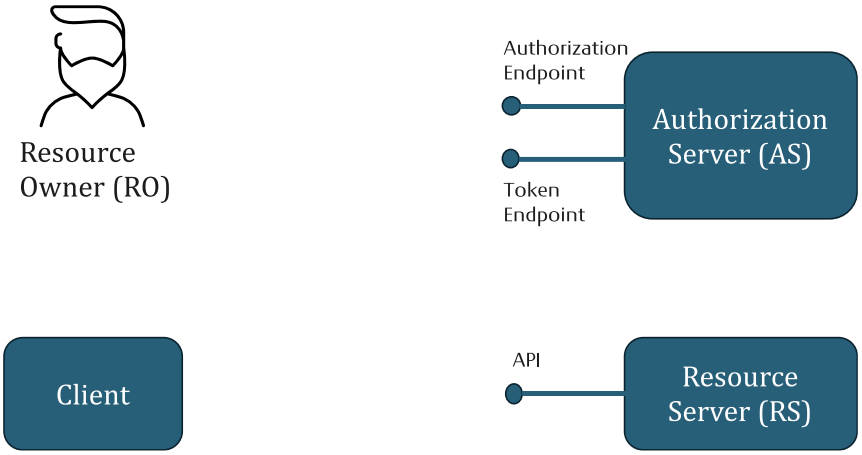
Sikkerhedsarkitekturen tager afsæt i [OAuth2] specifikationen som er en åben standard for adgangsdelegering til HTTP-baserede tjenester, herunder tjenester som udstilles som REST-fulde services, og baserer sig på [JTP-H] som er sundhedsvæsnets profilering af JSON Web Tokens [JWT].

1.3.1 OAuth2

Det generelle OAuth2 model fastsætter protokoller for hvordan en bruger (en *Resource Owner)* autoriserer en *client* (fx en mobil app eller en webapplikation) til at må tilgå beskyttede resurser hos en *Resource Server*. Adgang til beskyttede resurser gives på baggrund af et *access token*, som en *Authorization Server* udsteder til klienten efter brugergodkendelse.

I nedenstående

Figur 1 vises de fire aktører som indgår i OAuth2 modellen – *Resource Owner* er her afbilledet som menneskelig bruger, men kan i OAuth2 også være en systembruger.



Figur 1: Aktørerne i OAuth2

Bemærk i øvrigt, at 'Auth' i OAuth2 er en forkortelse for 'authorization' og ikke 'authentication'.

1.3.2 Confidential clients og public clients

Både mobile apps, IoT-devices, webapplikationer (single-page eller med backend) og server-processer kan optræde som *clients* i OAuth2 forstand, men der skelnes mellem *confidential clients* og *public clients.*

En *confidential client* er en klient som kan beskytte en hemmelighed (et *client secret* i form af en shared key eller et PKI nøglepar) som kan anvendes til autentifikation af klienten i Authorization Server, hvorimod en *public client* er en klient som ikke kan beskytte statiske hemmeligheder. En mobil app som distribueres via en app-store eller en webapplikation som afvikles direkte i browseren (fx en SPA) er som udgangspunkt *public clients,* idet en angriber vil kunne ekstrahere hemmeligheden fra den downloadede app eller fra HTML5/JavaScript-klienten i en ren browserbaseret applikation.

Under de rette forudsætninger kan en public client på runtime via [OAuth2-DCR] dynamisk klient registreringsprotokollen blive indrulleret ved Authorization Server med instans-specifikke akkreditiver og derefter optræde som confidential client.

1.3.3 OpenID Connect

Standarden OpenID Connect [OIDC] udvider OAuth2 fundamentet med et autentifikations- og identitets-lag. OIDC udvider OAuth2 med en brugerautentifikationsprotokol og med et *identity token*, som er et signeret JSON Web Token [JWT] med attributter om den autentificerede bruger. Hvor OAuth2 access tokens er møntet på eksterne API'er, udstedes OIDC identity token i stedet til klienten (når denne har behov for brugeroplysninger).

2 OVERORDNEDE USECASES

<TBD>

3 SIKKERHEDSMODEL

Udgangspunktet for sikkerhedsmodellen for EHMI services er dels OAuth-profileringen [iGOV-OAuth], som i *Appendiks: DK-Health OAuth profile v01* er blevet indsnævret til danske forhold. Se også *Udgangspunkt for profilering af OAuth 2.0 til sundhedsområdet* for en gennemgang af rationalet for valget af [iGOV-OAuth] profilen.

Hvor [iGOV-OAuth] profilen fastlægger sikkerhedsprotokoller og valideringsregler for aktørerne der indgår i et OAuth flows, forholder den sig ikke til indholdet af de tokens som indgår i de forskellige flows. Indhold at tokens tager derimod i EHMI afsæt i det danske sundhedsvæsnets profilering af [JWT] tokens, se [JTP-H].

I dette kapitel præsenteres og gennemgås de elementer af profilerne som er relevante for at understøtte de overordnede brugsscenarier som relaterer sig til EHMI services. Læseren forventes således ikke at have nærlæst [iGOV-OAuth] og [JTP-H] profilerne.

3.1 Klienttyper

I EHMI sikkerhedsmodellen skelnes der mellem to typer af OAuth klienter.

1. En *systemklient* som via system-til-system-integration tilgår en EHMI støtteservice. Selvom klienten måtte foretage opslag på baggrund af en brugerhandling, er systemklienten defineret ved at brugerens identitet ikke er relevant i den givne kontekst og ikke kommunikeres til EHMI støtteservicen. Eksempler på systemklienter er sundhedsadresseringsservicen som tilgår postkasseregisteret eller et fagsystem som tilgår forsendelsesstatusservicen. (I [iGOV-OAuth] kaldes en systemklient for en 'Direct Access Client'.)

2. En *fuld klient med brugerdelegering* et en klient som foretager kald for en autentificeret bruger, hvis identitet kommunikeres til EHMI støtteservicen som en del af tokenet der indgår i kaldet. Klienten er defineret som 'fuld' i OAuth forstand, idet den både kan autentificere sig selv og brugeren (via et webbrowser-baseret flow). Et eksempel på en fuld klient med brugerdelegering i en EHMI kontekst er backenden til en webapplikation som tilbyder søgninger i forsendelsesstatusservicen. (I [iGOV-OAuth] kaldes en fuld klient med brugerdelegering for en 'Full Client with User Delegation'.)

Begge klient-typer er således *confidential clients* i OAuth (se afsnit 1.3.2).

EHMI services tillader ikke direkte tilgange fra mobile/native apps uden backend. Denne klienttype ('Native Client with User Delegation' i [iGOV-OAuth]) indgår således ikke i de efterfølgende beskrivelser.

3.2 Klientautentifikation

Autentifikation af klienter til EHMI services er baseret på OCES systemcertifikater[i], hvor klient skal have sit eget nøglepar.

OCES systemcertifikaterne kan enten være udstedt af den organisation som anvender klienten eller af leverandøren som tilbyder løsningen (potentiel til flere organisationer). Netop ved multi-tenant systemer vil det som regel være mest hensigtsmæssigt at benytte et systemcertifikat udstedt af leverandøren.

I EHMI sikkerhedsmodellen benyttes OCES systemcertifikater alene til autentifikation af klienter hos Authorization Server og ikke til at autorisere adgange baseret på certifikatoplysninger (som fx CVR nummer), for at ikke at skabe unødvendige bindinger mellem certifikater og rettigheder.

3.3 Indrullering af klienter

I EHMI infrastrukturen understøttes kun *confidential clients*, som statisk registreres/indrulleres i Authorization Server.

Til pilotafprøvningen hvor der indgår et begrænset antal klienter registreres disse manuelt via Authorization Serverens administrationssnitflade. På sigt kunne der etableres en selvbetjeningsløsning hvor klienter selv kan anmode om indrullering og efter godkendt anmodning eksempelvis kunne benytte mekanismerne i [OAuth-DCR] til registrering i Authorization Server.

For at blive registreret danner klienten et simpelt metadata dokument som beskrevet nedenunder (metadata som er baseret på [OAuth-DCR]), der benyttes til konfiguration i Authorization Server.

Hver klient instans skal registreres separat og bør anvende sit eget OCES nøglepar og sit eget `client_name`. Ved registrering tildeles klienten et `client_id`, som denne skal benytte ved efterfølgende requests til henholdsvis token-endpointet og authorization-endpointet (for klienter med brugerdelegering) hos Authorization Server.

3.3.1 Metadata for systemklienter

Følgende elementer skal angives i klient metadata dokumentet:

Metadata element	Beskrivelse
`token_endpoint_auth_method`	Hvordan klienten autentificerer sig ved AS' token endpoint. Sættes til den faste værdi `private_key_jwt` dvs. autentifikation via (OCES) nøglepar.
`grant_types`	Hvilke OAuth flows klienten anvender. Sættes til den faste værdi `client_credentials`.
`client_name`	Et sigende navn for klienten, som letter administrationsopgaven i Authorization Server. Fx `EHMI Access Point for Region Midtjylland`.

[i] I OCES3 anvendes begreberne 'organisationscertifikat' og 'systemcertifikat' for hvad der i OCES2 henholdsvis blev kaldt virksomhedscertifikater (VOCES) og funktionscertifikater (FOCES).

Metadata element	Beskrivelse
`scope`	En liste af OAuth scope værdier (adskilt med blank space) som klienten ønsker at kunne lade indgå i access tokens. Fx `EAS EDS`.
`contacts`	Et array med kontaktoplysninger (typisk e-mailadresser) til organisationen, som er ansvarlig for driften af klienten.
`jwks_uri` eller `jwks`	En URI-reference til klientens [JWKS] dokument, som indeholder klientens public key eller selve JWKS dokumentet. Varianten med URI-angivelsen er den fortrukne metode, som har den fordel at klienten kan fornye eller opdatere sit OCES nøglepar uden at skulle opdater konfiguration hos Authorization Server. Se afsnit 3.3.3.

Eksempel metadata dokument for en systemklient:

```
{
  "token_endpoint_auth_method": "private_key_jwt",
  "grant_types": "client_credentials",
  "client_name": "EOJ Systemet i Korsbæk Kommune",
  "scope": "EDS DeliveryStatusEvent.c",
  "contacts": [
    "døgnsupport@korsbæk.dk",
    "+45 1234 5678"
  ],
  "jwks_uri": "https://eoj.korsbæk.dk/oauth/public_keys.jwks"
}
```

3.3.2 Metadata for fulde klienter med brugerdelegering

<TBD>

3.3.3 Indhold af JWKS dokument

JWKS dokumentet er et JSON dokument med et `keys` rodelement som er af array type. Følgende elementer skal angives i JWKS dokumentet for hver OCES public key som indgår som JSON struktur i `keys` arrayet. Typisk vil en klient kun have et aktiv OCES nøglepar (og dermed kun én indgang i `keys` arrayet), men i forbindelse med certifikat fornyelse / key roll-over kan det være nyttigt at kunne angive to.

JWKS element	Beskrivelse
`kty`	Typen af nøgle. Sættes til det faste værdi `RSA` for OCES certifikater (som er baseret på RSA-nøglepar).
`alg`	Algoritmen som klienten benytter til at signere 'client assertions' (se afsnit 3.4). Sættes til `RS256`, medmindre en enden algoritme er afstemt med Authorization Serveren.

JWKS element	Beskrivelse
`use`	Metoden som nøglen benyttes til. Sættes til den faste værdi `sig` (for 'signering').
`kid`	Et id for nøglen (en 'key identifier'). Indgår som en del af signaturen i 'client assertions' (se afsnit 3.4) til autentifikation af klienten. Hver nøgle i JWKS dokumentet skal have en unik `kid` værdi.
`n`	Værdien af modulus fra den underliggende RSA algoritme (beregnes ud fra public key).
`e`	Værdien af eksponenten fra den underliggende RSA algoritme (beregnes ud fra public key).
`x5c`	Et array med et enkelt element: OCES systemcertifikatet (uden den private nøgle) i base64-encoded form. (I x5c elementet kan der også angives hele certifikatkæden. I EHMI konteksten angives kun selve certifikatet.)

Eksempel JWKS dokument som `jwks_uri` kunne pege på ('*Komplet JWKS dokument*' viser det fulde dokument):

```
{
  "keys": [
    {
      "kty": "RSA",
      "alg": "RS256",
      "use": "sig",
      "kid": "Region_Midt_AP-key-1",
      "n": "yiM ... hd5",
      "e": "AQAB",
      "x5c": [
        "MIIG ... okI="
      ]
    }
  ]
}
```

Tips til udviklere: *Kommandolinje toolet `openssl` kan benyttes til at udtrække x509 certifikatet samt public key fra OCES nøgleparret i PKCS12 format (se fx https://medium.com/rahasak/openssl-293fead5576e eller https://www.sslshopper.com/article-most-common-openssl-commands.html). Et værktøj som fx https://russelldavies.github.io/jwk-creator/ eller et bibliotek som https://jwcrypto.readthedocs.io kan efterfølgende benyttes til at danne JWKS strukturen.*

3.4 Integrationsflows og protokoller

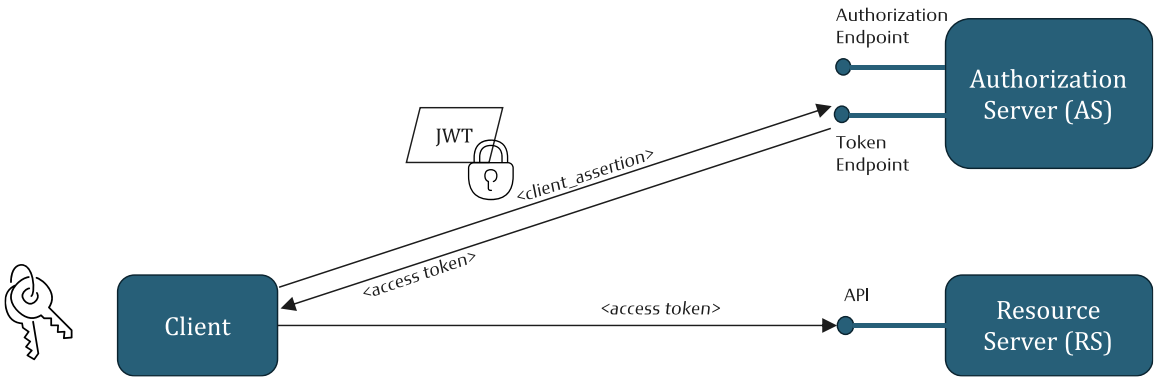
I både systemkalds- og brugerkalds-scenarierne foregår klient autentifikation via standard [JWT-Grant] flowet og med `private_key_jwt` mekanismen som defineret i [OIDC].

OBS: Der findes en lang række forskellige OAuth kodebiblioteker[ii] og JWT kodebiblioteker[iii] som implementer de standardflows og mekanismer som er defineret i de forskellige OAuth-relaterede specifikationer, som de her beskrevne flows baserer sig på. I en praktisk implementering kan der med fordel tages udgangspunkt i standard kodebibliotekerne.

I det følgende gennemgås der trin for trin integrationsflows for de to anvendelsesscenarier systemkald og brugerkald.

3.4.1 Systemkalds-scenarie

I systemkalds-scenariet danner og signerer klienten et token som autentifikationsbevis, som benyttes i kaldet til Authorization Serverens Token Endpoint. Authorization Serveren valideret tokenet og kaldet og returner et Access Token til klienten som denne efterfølgende benytter til at tilgå en EHMI støtteservice (som optræder som Ressource Server), se nedenstående figur.



Figur 2: Systemkalds-scenarie

I det følgende gennemgås de enkelte skridt i processen.

Skridt 1: Dannelse af JWT client assertion

Klienten danner et JWT og benytter `RS256` algoritmen (medmindre andet er aftalt) til at signere tokenet med den nøgle (eller en af de nøgler) som den har registreret i Authorization Server. Denne signerede JWT client assertion benyttes efterfølgende som autentifikationsbevis i kaldet til Authorization Serverens Token Endpoint.

Følgende JSON elementer skal angives i JWT client assertionen:

JSON element	Beskrivelse
`iss`	Udstederen (*issuer*) af tokenet. Sættes til `client_id` som klienten har fået tildelt af Authorization Server under registrering.

[ii] Se fx https://oauth.net/code/

[iii] Se fx https://openid.net/developers/jwt-jws-jwe-jwk-and-jwa-implementations/

JSON element	Beskrivelse
`sub`	Subjektet/anvenderen af tokenet. Sættes ligeledes til `client_id` som klienten har fået tildelt af Authorization Server under registrering.
`aud`	Modtager (*audience*) af tokenet. Sættes til URL'en for Authorization Serverens Token Endpoint.
`iat`	Udstedelsestidspunktet (*issued at*) for tokenet angivet i antal sekunder efter 1970-01-01T00:00:00Z UTC (*'Seconds Since the Epoch'*). Sættes til 'nu'.
`exp`	Gyldighedstidspunktet (*expiration time*), ligeledes angivet i '*Seconds Since the Epoch*'. Sættes til 'nu' plus 60 sekunder.
`jti`	En unik ID for tokenet som genereres af klienten, fx en UUID. Værdien af `jti` må ikke genbruges i efterfølgende tokens.
`kid`	En optionel angivelse af ID'et for den anvendte nøgle. Skal angives hvis klienten er konfigureret med mere end en offentlige nøgle (se JWKS i ovenstående).

Bemærk at JWT client assertion ikke følger [JTP-H] profilen, men bruges til autentifikation hos Authorization Server (et anvendelsesscenarie som [JTP-H] ikke forholder sig til).

Eksempel for en JWT client assertion (i ikke signeret form):

```
{
  "iss": "0ba284d1-8974-4241-bce1-0498bc2d48ea",
  "sub": "0ba284d1-8974-4241-bce1-0498bc2d48ea",
  "aud": "https://authorization.sunhedsdatastyrelsen.dk/token",
  "iat": 1710941316,
  "exp": 1710941376,
  "jti": "42a4b1c9-a7ec-4538-8f5a-ee687553870b"
}
```

Appendikset *JWT client assertion* indeholder et komplet signeret og encoded eksempel.

Skridt 2: Kald af Token Endpoint hos Authorization Server

Klienten laver et HTTP POST kald til Token Endpointet med angivelse af følgende kaldsparametre:

Parameter	Beskrivelse
`grant_type`	Sættes til den fast værdi `client_credentials`.
`scope`	Ønskede scope(s). Se nedenstående beskrivelser af sikkerhedsmodellen for de enkelte EHMI services for de konkrete scopes der skal angives. (Authorization Server benytter desuden `scope` værdien til at fastlægge en eventuel audience/aftager for access tokenet som den udsteder).
`client_id`	Sættes ti den client_id som blev tildelt klienten under indrullering ved Authorization Server.

Parameter	Beskrivelse
`client_assertion_type`	Sættes til den fast værdi: `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`
`client_assertion`	Sættes til den signerede og encodede JWT client assertion.

Klienten URL-encoder alle parameter værdierne, sætter dem sammen og laver POST kaldet til Token Endpoint.

Eksempel kald:

```
POST /token HTTP/1.1
Host: authorization.sunhedsdatastyrelsen.dk
User-Agent: curl/8.4.0
Accept: */*
Proxy-Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 863

grant_type=client_credentials&scope=EDS%20EAS&client_id=0ba284d1-
8974-4241-bce1-
0498bc2d48ea&client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclie
nt-assertion-type%3Ajwt-
bearer&client_assertion=eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiIwYmEyODRkMS04
OTc0LTQyNDEtYmNlMS0wNDk4YmMyZDQ4ZWEiLCJzdWIiOiIwYmEyODRkMS04OTc0LTQyN
DEtYmNlMS0wNDk4YmMyZDQ4ZWEiLCJhdWQiOiJodHRwczovL2F1dGhvcml6YXRpb24uc3
VuaGVkc2RhdGFzdHlyZWxzZW4uZGsvdG9rZW4iLCJpYXQiOjE3MTA5NDEzMTYsImV4cCI
6MTcxMDk0MTQzNiwianRpIjoiNDJhNGIxYzktYTdlYy00NTM4LThmNWEtZWU2ODc1NTM4
NzBiIn0.OG3a8WWiMt3kDKgFds3Xn_k7-
fh9rDQlLwbC9x_z0TkVr5QEbCJjRImFB5q4V_KlN6S2QPsGTl9jVeAWXD9ZeIAMIZc4Ig
eLtbgGQdwXkwNfc3TFxXg8IDAkB85bWXQRa0r-OyYdV07fqwj_IdRmnV--
sgWXR4Nttkd088Jbsslt81O9y-WmK4r13H6A13wYU-
1cAb3Bi07BfDQQJcQX_ip2wsODO6PAambjQgcXpbdNpwEtXwu_QV2elaUhanpEj1mtcEc
IWqL17rVcp2PeCrMbt7Bo0Two7_c5JtAG13nrH0cc7qo_VtYFhO45BEgxSr6NEHAXormH
uBYLTKAwpg
```

Skridt 3: Retursvar

Svaret fra Token Endpointet består af en JSON struktur med et access token, samt information om tokenet.

Følgende værdier returneres:

Returværdi	Beskrivelse
`access_token`	Access tokenet som klienten har requestet. Benyttes til efterfølgende kald til en EHMI støtteservice.
`token_type`	Typen af tokenet. Har altid den fast værdi `Bearer` i EHMI.
`expires_in`	Tokenets gyldighed i sekunder.
`scope` (Optionel returværdi)	Ønskede scope(s). Inkluderes altid hvis scopet er mindre end requestet.

Eksempel response fra Authorization Server:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
  "access_token":"ey ... J9.ey ... n0.OG ... pg",
  "token_type":"Bearer",
  "expires_in":300,
  "scope":"EDS"
}
```

Skridt 4: Kald EHMI støtteservice

I REST-kaldet til EHMI støtteservicen inkluderes Access Tokenet i en `Authorization` HTTP header som angivet:

```
Authorization: Bearer <access token>
```

3.4.2 Brugerkald (borgere/fagpersoner)

<TBD>

3.4.3 Token-indhold

<TBD>

3.5 Validerings- og forretningsregler

På alle HTTP forbindelser skal der som minimum anvendes TLS 1.2.

3.5.1 Regler for klienter

Klienter skal validere TLS/SSL certifikater fra såvel Authorization Server og de EHMI services som de integrerer til, herunder også foretage hostname-verifikation af certifikaterne, for at sikre sig at de har fat i de rette parter.

Klienterne skal behandle alle access tokens som de modtager fra Authorization Server som opaque tokens, dvs. som tokens i et proprietær format som er møntet på EHMI services. Klienten kan i stedet benytte relevante meta-informationer (token levetid og scopes) i svaret fra Token Endpointet.

3.5.2 Regler for Authorization Servere

<TBD>

3.5.3 Regler for Resource Servere (EHMI services)

<TBD>

4 EHMI DELIVERY STATUS (EDS)

I [EHMI] er der følgende *stationer* som indgår i punkt-til-punkt beskedsforsendelser: Fagsystemer, message-service-handlere og access-points.

Alle stationer der indgår i en EHMI beskedsforsendelse skal registrere deres beskedshåndteringer i forsendelsesstatusstatusservicen EDS, som beskrevet i FHIR implementation guiden på https://build.fhir.org/ig/medcomdk/dk-ehmi-eds/.

Som det fremgår af FHIR implementation guiden realiseres forsendelsesstatus som en profilering af FHIR `AuditEvent` ressourcen hvor der er defineret følgende typer af forsendelsesstatus som registreres:

Type	Beskrivelse
`created`	Når en station (typisk et fagsystem) danner beskeden.
`sent`	Når en station (videre)sender beskeden.
`created-and-sent`	Når en station danner og sender beskeden.
`received`	Når en station modtager beskeden.
`finalized`	Når en station som slutmodtager behandler beskeden.
`received-and-finalized`	Når en station modtager og som slutmodtager behandler beskeden.

Stationerne oprettes i EHMI Endpoint registeret (EER) og tildeles i forbindelse med oprettelsen et unikt *device_id.*

4.1 Usecases

Der er to overordnende usecases for anvendelsen af forsendelsesstatusservicen EDS.

Stationerne i en EHMI forsendelse foretage hver især en *registrering af forsendelsesstatus* i EDS.

EDS stiller en grænseflade til *søgning og opslag* til rådighed, som kan benyttes til track'n'trace af beskedsforsendelser eller til fejlsøgning.

4.2 Specificering af sikkerhedsmodel til EDS

Udgangspunkt er ovenstående generelle '*Sikkerhedsmodel*' med følgende udvidelser.

4.2.1 Indrullering/whitelisting af systemklienter til registrering i EDS

Udover de i afsnit '*3.3 Indrullering af klienter'* beskrevne elementer skal der under indrullering af systemklienter der foretager registreringer i EDS angives følgende:

- Det *device_id* som stationen er registreret med i EER
- En liste af *SOR-organisationer* som stationen sender/modtager beskeder for i form af SOR kode og GLN lokationsnummer

Under indrullering angives som scope element:

```
EDS system/AuditEvent.c
```

(Ovenstående `system/AuditEvent.c` syntaks er baseret på definitionen af scopes for FHIR ressourcer i [SMART].)

Metadata for en EDS systemklient

Udover de i afsnit '*3.3.1 Metadata for systemklienter*' beskrevne metadata elementer skal ovenstående elementer angives på følgende form for systemklienter der skal indrulleres til at måtte foretage registreringer i EDS.

Metadata element	Beskrivelse
`ehmi:eer:device_id`	En angivelse af device_id som stationen er registreret med i EER.
`ehmi:org_context`	Array af JSON objekter bestående af `name` (organisationsnavn), `sor` (SOR kode) og `gln` (lokationsnummer) som stationen sender/modtager beskeder for.

Eksempel metadata dokument for en EDS systemklient:

```
{
  "token_endpoint_auth_method": "private_key_jwt",
  "grant_types": "client_credentials",
  "client_name": "Apotekssystemet for Aarhus Åbyhøj Apoteket",
  "scope": "EDS system/AuditEvent.c",
  "contacts": [
    "døgnsupport@aarhus-aabyhoej-apoteket.dk",
    "+45 1234 5678"
  ],
  "jwks_uri": "https://eer.ehmi.dk/device/c4b8d3ea-b187-426b-be77-
bffd9f593d84/public_keys.jwks",
  "ehmi:eer:device_id": "c4b8d3ea-b187-426b-be77-bffd9f593d84",
  "ehmi:org_context": [
    {
      "name": "Aarhus Åbyhøj Apotek",
      "sor": "306861000016006",
      "gln": "5790000173372"
    },
    {
      "name": "Bruun's Apotek",
      "sor": "625961000016008",
      "gln": "5790002275296"
    }
  ]
}
```

4.2.2 Indrullering/whitelisting af klienter til søgning og opslag

<TBD>

4.2.3 Kald til Token Endpoint

I tilgangen til EDS opereres der med organisations-specifikke tokens, dvs. klienter som optræder i flere organisatoriske kontekster skal trække et særskilt token hos Authorization Server for hver SOR/GLN kontekst.

For at få udstedt et access token til EDS angives følgende scopes:

scope	Beskrivelse
`EDS`	En angivelse af det er for EDS at klienter ønsker et access token.
`system/AuditEvent.c`	En angivelse at tokenet skal kunne registrere/oprette forsendelsesstatus ressourcer (som er profileringer af FHIR's AuditEvent).
`SOR:<XXXXX>`	En angivelse af organisationens SOR kode, hvor `<XXXXX>` sættes til selve koden.
`GLN:<YYYYY>`	En angivelse af organisationens GLN lokationsnummer, hvor `<YYYYY>` sættes til selve lokationsnummeret.

Eksempel på en samlet scope som indgår i kaldet:

```
EDS system/AuditEvent.c SOR:306861000016006 GLN:5790000173372
```

Eksemplet på et kald til Token Endpointet med ovenstående eksempel scope:

```
POST /token HTTP/1.1
Host: authorization.sunhedsdatastyrelsen.dk
User-Agent: curl/8.4.0
Accept: */*
Proxy-Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 973

grant_type=client_credentials&scope=EDS%20system%2FAuditEvent.c%20SOR
%3A306861000016006%20GLN%3A5790000173372&client_id=0ba284d1-8974-
4241-bce1-
0498bc2d48ea&client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclie
nt-assertion-type%3Ajwt-bearer&client_assertion=eyJh...Awpg
```

Valideringer af kaldet hos Authorization Server

Kaldet til Token Endpointet valideres hos Authorization Server, som tjekker at klienten er indrulleret/whitelistet med de angivne scopes. Authorization Serveren mapper således `client_id` fra kaldet til det registrerede device_id og validerer at klienten er whitelistet til såvel EDS servicen, den angivne 'create' operation for `AuditEvent` ressourcen og den angivne organisatoriske kontekst i form af SOR kode og GLN lokationsnummer.

4.2.4 Kald til EDS

For at registrere en forsendelsesstatus opretter klienten lokal et `AuditEvent` FHIR objekt (som overholder `EdsPatientDeliveryStatus` eller `EdsBasicDeliveryStatus` profilen) og kalder forsendelsesstatusservicen med et HTTP POST kald, som har 'create' semantikken. I POST kaldet inkluderes access tokenet som beskrevet i ovenstående og det nyoprettede `AuditEvent` FHIR objekt placeres i HTTP body-delen.

Eksempel på kald til EDS med AuditEvent ressourcen angivet som JSON objekt:

```
POST /base/AuditEvent HTTP/1.1
Host: ehmi.medcom.dk
User-Agent: curl/8.4.0
Accept: application/fhir+json
Content-Type: application/fhir+json
Content-Length: 11996
Authorization: Bearer eyJhb ... Dhi6g

{
  "resourceType" : "AuditEvent",
  "id" : " cbcb0de9-105e-470a-8754-ffad3b581ed4",
  "meta" : {
    "profile" : [
       "http://medcomehmi.dk/ig/dk-ehmi-
eds/StructureDefinition/EdsPatientDeliveryStatus"
    ]
  },

...

}
```

EDS adgangskontrol

Forsendelsesstatusservicen tjekker at access tokenet er gyldigt, udstedt til EDS som aftager af tokenet og indeholder de nødvendige scopes til at klienten må foretage registreringer i EDS (her "`EDS system/AuditEvent.c`").

EDS tjekker desuden hvorvidt SOR koden, GLN lokationsnummeret og device_id som den kan uddrage af access tokenet, matcher oplysningerne i den medsendte AuditEvent ressource.

5 EHMI ADRESSERING SERVICE (EAS)

<TBD>

6 EHMI ENDPOINT REGISTER (EER)

<TBD>

7 APPENDIKS: DK-HEALTH OAUTH PROFILE V01

The DK-Health OAuth profile is based on the [iGOV-OAuth] assurance profile and follows the requirements laid out in the assurance profile.

This appendix defines the additional constraints that make out the DK-Health OAuth profile.

Addition to iGOV-OAuth Section 2.3.3. Client Keys

Full clients and direct access clients SHOULD use public/private keypairs derived from [OCES] system certificates.

<TBD>

8 APPENDIKS: ARKITEKTURBESLUTNINGER

I dette kapitel fastholdes begrundelserne for væsentlige arkitekturvalg som ligger til grund from udformning af denne sikkerhedsarkitektur.

8.1 Udgangspunkt for profilering af OAuth 2.0 til sundhedsområdet

Problemstilling	Hvilken OAuth 2.0 sikkerhedsprofil bør profileringen af OAuth 2.0 til det danske sundhedsvæsen basere sig på?
Antagelse	Til det generelle [OAuth] framework findes der lang række yderligere specifikationer (typisk i form af [IETF RFC]'er) som fastlægger token-formater, protokol-flows, klienttype-specifikke udvidelser, sikkerheds-tiltag mod kendte angreb mm. At basere en dansk profilering af OAuth til sundhedsområdet alene på de mange basis specifikationer vil være en kompleks opgave og vanskeligt at overskue for anvenderne. I stedet bør profileringen basere sig på en allerede eksisterende sikkerhedsprofilering af OAuth som bygger på 'best practice' på området.
Muligheder	Følgende sikkerhedsprofileringer er identificeret som mulige kandidater: - [FAPI] - [HEART] - [iGOV-OAuth] - [OIO-OIDC] - [SMART]
Analyse	[FAPI] specifikationen er forankret i OpenID Foundation og har sin oprindelse i bankverdenen og [PSD 2] EU-direktivet, men er i version 2.0 blevet generaliseret til at være bredt anvendelig i alle områder med høje sikkerhedskrav. Specifikationen forudsætte anvendelsen af de nyeste OAuth2 teknologier (som fx [DPOP] og [PAR]), og ser ud til at have noget begrænset tool-support på nuværende tidspunkt. Det norske Helsenett har baseret deres HelseID løsning på FAPI 2.0. Erfaringerne fra Norge har vist, at FAPI 2.0 kan implementeres i praksis, men kræver en del støtte til anvenderne i forhold til de nyeste teknologier og fraværet af bred tool-support. [HEART] som ligeledes er forankret under OpenID Foundation er en decideret sikkerhedsprofilering af OAuth 2.0 til sundhedsområdet, som er skabt til at understøtte andre profiler under HEART initiativet. Det ser ikke ud til at profilen er blevet vedligeholdt siden 2018 og der peges i [HEART] fortsat på flows som jf. 'best practice' på området ikke længere bør anvendes. I [iGOV-OAuth] fra OpenID Foundation er sikkerhedsprofileringen fra [HEART] blevet ajourført og generaliseret. I [iGOV-OAuth] tages der afsæt i velafprøvede og bredt anvendte specifikationer, samt at der åbnes op for nyere teknologier (som [DPOP]) uden at gøre anvendelsen påkrævet. Både Holland og Italien har taget udgangspunkt i specifikationen i deres nationale OAuth profileringer og det engelske NHS har ligeledes baseret deres føderation på [iGOV-OAuth]. Specifikationen fremstår operationel med klart definerede krav til de enkelte aktører, protokolflows og beskeder. Specifikationen suppleres med en profilering af [OIDC] ('identitets-laget' ovenpå [OAuth]), se [iGOV-OIDC]. [OIO-OIDC] er Digitaliseringsstyrelsen bud på en OpenID Connect profilering. Specifikationen har ikke været anvendt i praksis og profileringen går netop kun på

	'identitets-laget' OpenID Connect og angiver ikke hvordan rene system-til-system integrationer kan realiseres. [SMART] er HL7's profilering af OAuth i forhold til adgangsstyring af FHIR-snitflader. Specifikationen tager i høj grad afsæt i FHIR terminologien og fremstår operationel. Udover en general angivelse af hvordan OAuth bør anvendes i en FHIR kontekst, defineres en model for OAuth 'scopes' for adgangsstyring til FHIR ressourcer. I modsætning til de andre fire analyserede specifikationer fremstår SMART mindre stringent (fx mangler der delvis referencer til underliggende standarder/versioner) og har nogle steder mest karakter af en udviklervejledning. Derudover definer SMART sin egen metadata-discovery-mekanisme, i sted for at basere sig på standard mekanismerne.
Beslutning	Profilering af OAuth 2.0 til sundhedsområdet tager udgangspunkt i [iGOV-OAuth] og suppleres med en profilering af [iGOV-OIDC] for at understøtte identitetsbaserede anvendelsesscenarier. I profileringen tillades der derudover, at der i FHIR-baserede anvendelser må benyttes mekanismerne fra SMART ('scope' definitioner, SMART-specifik metadata-discovery-mekanisme) for at være compliant med denne specifikation.

8.2 Selv-indeholdt token eller token reference?

<TBD>

8.3 Indhold af token til EDS: device_id, SOR kode og GLN

<TBD>

9 APPENDIKS: EKSEMPLER

9.1 Komplet JWKS dokument

Nedenstående er et komplet eksempel, hvor modulus (n) og eksponent (e) som definerer public key matcher public key fra certifikatet (x5c).


```
{
  "keys": [
    {
      "kty": "RSA",
      "alg": "RS256",
      "use": "sig",
      "kid": "Region_Midt_AP-key-1",
      "n":
"yiM__9uViZ525zwviXjtuWBwvz2YE7Zo_QYsxrK0NDmVAm4Mso47SqqRGgv8Ezy9H5oUoLK
C2B2RZdIgTSr5jH-
e2QLs8ZVbeoGbX5s180kjY7VT5dy1SX0KigZvLOiK_aFScrMfmlmOlU5T5kzbDj6GGBqoRJ3
jVav05wAankeANnOlDpwEJv1l0r_O9E2kSaYaEZh50UfNdGzqv9NQaa2pYrXMI2hvudYDuS8
4Sh9r0smbmklEgKNqCkGWx6QQpjGHr5ZDZrnGi_MLZwAsGHxETbx1t6OnevClrkEKABN7MSG
223ldx-FeobalsGIfB9V8Xd9epmiKu-
g9urmwrzabH1wmMljoHZ_GrgOtWrw65bnhJWlXVD0lPbSFubusIBB-
wBaVcemxkXOwl7idHhg4zmwPn8nW4QZL7I8ndU2zOVAcJBQmWPYdtM5JzgcnBd4lznG6v_5y
2gu3aeM8f9b2FZDbNKASkRgp6OpXmwIqOuiPpqJo3R4zbx8wzhd5",
      "e": "AQAB",
      "x5c": [
"MIIGrDCCBOCgAwIBAgIUV5sHOblG/Ab9Av36a9FtrX1SRkAwQQYJKoZIhvcNAQEKMDSgDzA
NBglghkgBZQMEAgEFAKEcMBoGCSqGSIb3DQEBCDANBglghkgBZQMEAgEFAKIDAgEgMGsxLTA
rBgNVBAMMJERlbiBEYW5za2UgU3RhdCBPQ0VTIHVkc3RlZGVuZGUtQ0EgMTETMBEGA1UECww
KVGVzdCAtIGN0aTEYMBYGA1UECgwPRGVuIERhbnNrZSBTdGF0MQswCQYDVQQGEwJESzAeFw0
yMjAzMzExNzU0MTdaFw0yNTAzMzAxNzU0MTZaMIGmMS4wLAYDVQQDDCVMYWtlc2lkZSB0ZXN
0IG9yZ2FuaXNhdGlvbnNjZXJ0aWZpa2F0MTcwNQYDVQQFEy5VSTpESy1POkc6OWI5OTZiZTE
tYjQzOS00NWFiLWIyMzktMGM5NWQ4ZTAyYWVlMRUwEwYDVQQKDAxMYWtlc2lkZSBBL1MxFzA
VBgNVBGEMDk5UUkRLLTI1NDUwNDQyMQswCQYDVQQGEwJESzCCAaIwDQYJKoZIhvcNAQEBBQA
DggGPADCCAYoCggGBAMojP//blYmeduc8L4l47blgcL89mBO2aP0GLMaytDQ5lQJuDLKOO0q
qkRoL/BM8vR+aFKCygtgdkWXSIE0q+Yx/ntkC7PGVW3qBm1+bNfNJI2O1U+XctUl9CooGbyz
oiv2hUnKzH5pZjpVOU+ZM2w4+hhgaqESd41Wr9OcAGp5HgDZzpQ6cBCb9ZdK/zvRNpEmmGhG
YedFHzXRs6r/TUGmtqWK1zCNob7nWA7kvOEofa9LJm5pJRICjagpBlsekEKYxh6+WQ2a5xov
zC2cALBh8RE28dbejp3rwpa5BCgATezEhttt5XcfhXqG2pbBiHwfVfF3fXqZoirvoPbq5sK8
2mx9cJjJY6B2fxq4DrVq8OuW54SVpV1Q9JT20hbm7rCAQfsAWlXHpsZFzsJe4nR4YOM5sD5/
J1uEGS+yPJ3VNszlQHCQUJlj2HbTOSc4HJwXeJc5xur/+ctoLt2njPH/W9hWQ2zSgEpEYKej
qV5sCKjroj6aiaN0eM28fMM4XeQIDAQABo4IBojCCAZ4wDAYDVR0TAQH/BAIwADAfBgNVHSM
EGDAWgBR/KJ/ZcZlC4nXn1zV2Lk0IJW12XjB7BggrBgEFBQcBAQRvMG0wQwYIKwYBBQUHMAK
GN2h0dHA6Ly9jYTEuY3RpLWdvdi5kay9vY2VzL2lzc3VpbmcvMS9jYWNlcnQvaXNzdWluZy5
jZXIwJgYIKwYBBQUHMAGGGmh0dHA6Ly9jYTEuY3RpLWdvdi5kay9vY3NwMBoGA1UdEQQTMBG
BD2NoZ0BsYWtlc2lkZS5kazAhBgNVHSAEGjAYMAgGBgQAj3oBATAMBgoqgVCBKQEBAQMHMDs
GCCsGAQUFBwEDBC8wLTArBggrBgEFBQcLAjAfBgcEAIvsSQECMBSGEmh0dHBzOi8vdWlkLmd
vdi5kazBFBgNVHR8EPjA8MDqgOKA2hjRodHRwOi8vY2ExLmN0aS1nb3YuZGsvb2Nlcy9pc3N
1aW5nLzEvY3JsL2lzc3VpbmcuY3JsMB0GA1UdDgQWBBROi7+XwccxFCcVKOvuXDf/OyvbGTA
OBgNVHQ8BAf8EBAMCBeAwQQYJKoZIhvcNAQEKMDSgDzANBglghkgBZQMEAgEFAKEcMBoGCSq
GSIb3DQEBCDANBglghkgBZQMEAgEFAKIDAgEgA4IBgQAltvyXPJkKoMP6KdG24hNqgciv8Bp
Z5xCUsHHzpqfd6sME3tEyW5uhdwccEmBybnJEixDoh3+vHHYPdBwStnkmPFzQTqjem3U8Evv
zKCtT7m3XUcvsJ0LyHXaVIq07d98CuRv7rU8VDStlO9q150g2GLNqruD0LBpgUBTi8FP6peY
OULaVVYhRe6PyOy2AtOzIrn+w3Ovxo7TOGRPcSYXfxPHVj/YxBTMQifU3WCSlt27KHlaDFEw
HlwM+wEDEFkTiBXBpZlB/NNmoWcA8fPWM1lCHOlZiQlcztaVtXXz6wcOFxsQizvVqaWp2SWx
zD3D2YWMwANqUL96WX8jNSqFvon2FYBs7hxk+xnP4khXoMbjm64FaulCqnU5xQcMvtVesLUC
mqe1loZn/GrWRu0+CDDTwT9apgn4TOyDYF/GDuQj01S0iR7fj5WbcyN0vuo1pMvsxQokRUB1
```

```
uE/VBj1AP1npcfaThnAN/RQDLzrVzu8hLFDvlFLLrMb9R4VvFokI="
      ]
    }
  ]
}
```

9.2 JWT client assertion

Nedenstående er et eksempel på en signeret og encoded JWT client assertion:

eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiIwYmEyODRkMS04OTc0LTQyNDEtYmNlMS0wNDk4YmMyZDQ4ZWEiLCJzdWIiOiIwYmEyODRkMS04OTc0LTQyNDEtYmNlMS0wNDk4YmMyZDQ4ZWEiLCJhdWQiOiJodHRwczovL2F1dGhvcml6YXRpb24uc3VuaGVkc2RhdGFzdHlyZWxzZW4uZGsvdG9rZW4iLCJpYXQiOjE3MTA5NDEzMTYsImV4cCI6MTcxMDk0MTM3NiwianRpIjoiNDJhNGIxYzktYTdlYy00NTM4LThmNWEtZWU2ODc1NTM4NzBiIn0.
Tih5FZDvZx0bfuyJXbVusUGfeTJWJAiRTXcvPcadDLJ7gbgnUFGyarKVcf1aaG8iCq5G6u9gPQagHfuY7Om0yPjC2VpdF8Ey0PKL21VNxk3wtVxirwjIGjzV7llvTCkZAUMqxY1M-RDWP5m-PKapQmAVoB_2w3m8rBK-2Aih_V_90JGcfnSaAZDCkTG8s6OsOJvELQRrPt2n7X2-s4bnnYLR9nOXNUM3ePDW-jhvKXePOTsDAQqKOhydTOa6LYG0yb5usOuDAY6OUpo15MsY7d2m1ttZwwG3utKmJogJEQ-rIjl4VngLcLxsMHgMDAfadudmazEduRnSR4yWUm5_IK_qtumMiAw2ARnZnIvf0GOQ4NPd1wZlh8yscZXnLD82Huh1ouFvMsKZmZBgos4oTKu_Yuc2dF6dGkMmcvxieqwi7U0okEGEZb5GYq8mrRM0jrHqyL26voK2lOVODdltGFyIR49vUmV0-42NzBSsCTZb2k_1iOpV7CoNbp3Dhi6g

og kan signaturvalideres med denne public key (fx på https://jwt.io/):

```
-----BEGIN PUBLIC KEY-----

MIIBojANBgkqhkiG9w0BAQEFAAOCAY8AMIIBigKCAYEAyiM//9uViZ525zwviXjtuWBwvz2Y
E7Zo/QYsxrK0NDmVAm4Mso47SqqRGgv8Ezy9H5oUoLKC2B2RZdIgTSr5jH+e2QLs8ZVbeoGb
X5s180kjY7VT5dy1SX0KigZvLOiK/aFScrMfmlmOlU5T5kzbDj6GGBqoRJ3jVav05wAankeA
NnOlDpwEJv1l0r/O9E2kSaYaEZh50UfNdGzqv9NQaa2pYrXMI2hvudYDuS84Sh9r0smbmklE
gKNqCkGWx6QQpjGHr5ZDZrnGi/MLZwAsGHxETbx1t6OnevClrkEKABN7MSG223ldx+Feobal
sGIfB9V8Xd9epmiKu+g9urmwrzabH1wmMljoHZ/GrgOtWrw65bnhJWlXVD0lPbSFubusIBB+
wBaVcemxkXOwl7idHhg4zmwPn8nW4QZL7I8ndU2zOVAcJBQmWPYdtM5JzgcnBd4lznG6v/5y
2gu3aeM8f9b2FZDbNKASkRgp6OpXmwIqOuiPpqJo3R4zbx8wzhd5AgMBAAE=

-----END PUBLIC KEY-----
```

10 REFERENCER

- [DPOP] “OAuth 2.0 Demonstrating Proof of Possession (DPoP)”, https://datatracker.ietf.org/doc/html/rfc9449
- [EHMI] “Ny infrastruktur (EHMI)”, https://medcom.dk/projekter/kommunale-proevesvar-paa-ny-infrastruktur/kommunale-proevesvar-ny-infrastruktur-ehmi/
- [FAPI] ”FAPI 2.0 Security Profile”, https://openid.net/specs/fapi-2_0-security-profile-ID2.html
- [HEART] ”Health Relationship Trust Profile for OAuth 2.0”, https://openid.net/specs/openid-heart-oauth2-1_0.html
- [IETF RFC] ”The Internet Engineering Task Force (IETF) - RFCs”, https://www.ietf.org/standards/rfcs/
- [iGOV-OAuth] “International Government Assurance Profile (iGov) for OAuth 2.0”, https://openid.net/specs/openid-igov-oauth2-1_0.html
- [iGOV-OIDC] “International Government Assurance Profile (iGov) for OpenID Connect 1.0”, https://openid.net/specs/openid-igov-openid-connect-1_0.html
- [JTP-H] “JWT Token Profile for Healthcare (JTP-H)”
- [JWKS] “JSON Web Key (JWK)”, https://datatracker.ietf.org/doc/html/rfc7517
- [JWT] “JSON Web Token (JWT)”, https://datatracker.ietf.org/doc/html/rfc7519
- [JWT-Grant] “JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants”, https://datatracker.ietf.org/doc/html/rfc7523
- [OAuth] “The OAuth 2.0 Authorization Framework”, https://datatracker.ietf.org/doc/html/rfc6749
- [OAuth-DCR] ”OAuth 2.0 Dynamic Client Registration Protocol”, https://datatracker.ietf.org/doc/html/rfc7591
- [OCES] “OCES (Offentlige certifikater til Elektronisk Service)”, https://certifikat.gov.dk/politikker.for.tillidstjenester/ og https://mitid-erhverv.dk/avanceret/certifikater/
- [OIDC] ”OpenID Connect Core 1.0”, https://openid.net/specs/openid-connect-core-1_0.html
- [OIO-OIDC] “OIO Open ID Connect Profiles Version 0.91”, https://digst.dk/media/24669/oio-oidc-profiles-v091.pdf
- [PAR] ”OAuth 2.0 Pushed Authorization Requests”, https://datatracker.ietf.org/doc/html/rfc9126
- [PSD 2] ”PSD 2 Direktiv”, https://www.finanstilsynet.dk/Lovgivning/Ny_EU_lovsamling/PSD-2
- [SMART] ”SMART App Launch 2.1.0”, http://hl7.org/fhir/smart-app-launch/index.html