e-oppslag, modell for autorisering og delegering

# Innledning

## Om dette dokumentet

Dette dokumentet er et saksfremlegg til arbeidet med referansearkitektur for «e-oppslag», tatt frem av en arbeidsgruppe bestående av:

* Jørgen Binningsbø, Difi
* Rolf Rander Næss, Acando AS, for BRREG
* Øyvind Syrstad, Systek AS, for Skattedirektoratet

## Mål og rammer

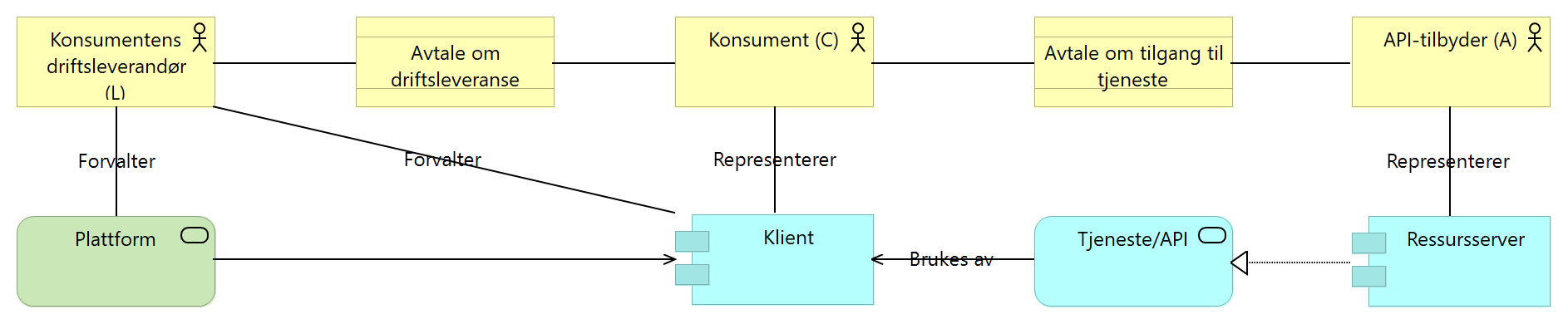
Dette dokumentet definerer en modell for å diskutere autentisering og autorisasjon i forbindelse med synkrone tjenestekall som implementerer system-til-system-kommunikasjon. Videre foreslås en rekke modeller for å implementere dette i en token-basert tjeneste som er tenkt å være felles på tvers av offentlig sektor. Det er antatt at dette vil implementeres som en utvidelse av dagens «maskin-porten» som tilbys av Difi, men modellen forutsetter ikke dette.

Modellen som diskuteres i dette dokumentet er ment å løse et felles behov hos offentlige virksomheter som skal utveksle informasjon med offentlige og private virksomheter gjennom tekniske grensesnitt.

# Generell modell for tilgangsstyring

## Overordnet modell

Figuren under er i «archimate»-notasjon og oppsummerer de aktører og komponenter som er relevante i diskusjonen av tilgangsstyring.



## Aktører

En aktør er notert i archimate som en gul boks med en strekfigur. Vi diskuterte 4 ulike typer aktører, bokstavene i parentes er de vi brukte i figurene på tavla:

* API-tilbyder (A): den part som tilbyr en tjeneste gjennom et API
* Konsument (C): den part som skal bruke tjenesten og juridisk sett er mottaker
* Konsumentens driftsleverandør (L): en separat organisasjon som konsumenten bruker til å forvalte deler av eller hele sin tekniske infrastruktur, samt som i større eller mindre grad er integrert i konsumentens forretningsprosesser.

Modellen sier ikke direkte noe om databehandler eller behandlingsansvarlig, men typisk vil konsument (C) være behandlingsansvarlig for data den henter gjennom API-et. Dersom konsumenten bruker en driftsleverandør (L) vil denne være databehandler. API-tilbyder vil være en databehandler, og kan også være behandlingsansvarlig.

## Avtaler

En avtale i archimate-notasjon er en gul boks med to vannrette striper. Det er to avtaler involvert som er relevant:

* *Driftsavtale:* Avtale mellom konsument (C) og driftsleverandør (L). Dersom L skal opptre på vegne av C (scenario 3 over) er det denne avtalen som delegerer myndighet fra C til L.
* *API-avtale:* Avtale mellom tjenesteeier (A) og konsument (C). Dette er den avtalen som regulerer C sin tilgang på opplysninger/tjenester fra A. I vår sammenheng hvor C og/eller A er offentlige virksomheter vil denne avtalen typisk være strengt regulert i lov og forskrift.

## Applikasjonstjenester og -komponenter

Alle begrep på applikasjonslaget er turkis i archimate-notasjonen. Bokser med runde hjørner er tjenester/API, bokser med to «kontakter» til venstre er komponenter.

* Klient: dette er omtrent det samme som «client» i OAuth, altså den komponenten som ønsker tilgang til en ressurs
* Ressursserver: Applikasjon som implementerer tjeneste, det samme som «resource server» i OAuth.
* Tjeneste/API: grensesnittet som realiseres av «resource server» og som brukes/konsummeres av «client»

Pilene betyr:

* Åpen pil: «brukes av»
* Trekant med stiplet linje: «realiserer»

## Tilgangsstyringsmodell

Noen generelle føringer og antagelser som ligger til grunn:

* Ressursserver trenger vite hvem konsumenten (C) er
* Konsument (C) kan velge å delegere myndighet til å kalle en eller flere spesifikke tjenester til driftsleverandør (L). Klienten vil autentiseres som enten C eller L. Kapittel 3.1 har konkrete eksempler på når klienten typisk vil velge det ene eller det andre.
* Enten klienten autentiserer seg som konsument (C) eller driftsleverandør (L) er det C som er ansvarlig for bruken av tjenesten
* Det er tjenesteeier (A) som bestemmer hvilke konsumenter (C) som får kalle tjenesten. Den juridiske delen av dette er avtalen mellom A og C, men den tekniske er en tilgangspolicy som sjekker at klienten opptrer som C.
* I praksis vil vi ikke styre tilgang til en konkret tjeneste, men til et «API» eller en gruppe API-er eller et subsett av funksjonalitet innenfor et API. Identifikasjon av API-et er uavhengig av API-tilbyder. Hvilke tilganger som er relevante for et gitt API er en egenskap ved dette API-et, og dersom samme API tilbys av flere tilbydere (for eksempel alle banker tilbyr «hent kontoopplysninger») vil identifikator og semantikk for ulike tilganger være felles. Dette er spesielt relevant dersom dette skal kodes inn i et tilgangstoken, fordi man da må ha identifikatorer i tokenet som har en definert semantikk på tvers av tilbydere. Dette vil begrense et tokens gyldighet til å kun gjelde en bestemt type operasjon. Dette reduserer risiko dersom token-et skulle komme på avveie.
* API-tilbyder tilbyr et API, flere API-tilbydere kan tilby samme API, en API-tilbyder vil også typisk tilby flere API. API-tilbyder identifiseres uavhengig av API-et. Dette er også relevant når tilgang styres ved hjelp av token, fordi dette begrenser et tokens gyldighet til å gjelde kun en API-tilbyder.

Avgrensninger, hva vi ikke dekker i denne modellen:

* Krav til ulike sikkerhetsnivåer på autentiseringen. En av aktørene må være en offentlig virksomhet.
* Involvering av personer, verken system som opptrer på vegne av en person, eller en tjeneste som trenger et samtykke fra en person.

Da ender vi opp med en tilgangsstyringsmodell som dette:

|  |  |  |  |
| --- | --- | --- | --- |
| Driftsleverandør L | Konsument, C | API (-gruppe/-funksjon) | API-tilbyder |
|  |  |  |  |

Gitt definisjonene over er det slik at:

* Innholdet i kolonnene konsument, API og API-tilbyder reguleres av API-avtalen
* Innholdet i kolonne driftsleverandør reguleres av Driftsavtalen

Kapittel 3.2 har et praktisk eksempel på en slik tabell. Merk at for de fleste tjenester vil det bare finnes en API-tilbyder, og den kolonnen vil dermed være mindre interessant for tilgangsstyringen (API-tilbyderen vil være gitt fra API-et).

Begrepene «API(-gruppe/-funksjon)» og «API-tilbyder» i denne tabellen tilsvarer OAuth2-begrepene «scope» og «audience», men disse begrepene i modellen trenger ikke nødvendigvis implementeres som scope/audience i OAuth2. Her vil det være praktiske og tekniske begrensninger i løsningen.

# Implementasjonsalternativer

Noen antagelser og føringer som ligger til grunn for den videre diskusjonen:

* Løsningen skal gjelde for synkrone tjenestekall over HTTPS, typisk gjennom et REST-pattern
* Autentisering av klienten opp mot ressursserver gjøres med et JWT-token som kan legges i en http-header
* Dersom konsumenten (C) velger å bruke en driftsleverandør (L) for å bygge slike integrasjoner antar vi at konsumenten ikke har mulighet til å kalle noe API for å konfigurere tilgangsstyring.
* Derimot antas det at API-tilbyder (A) har mulighet til å kalle et API for å konfigurere tilgangsstyringen.

## Komponenter i løsningen

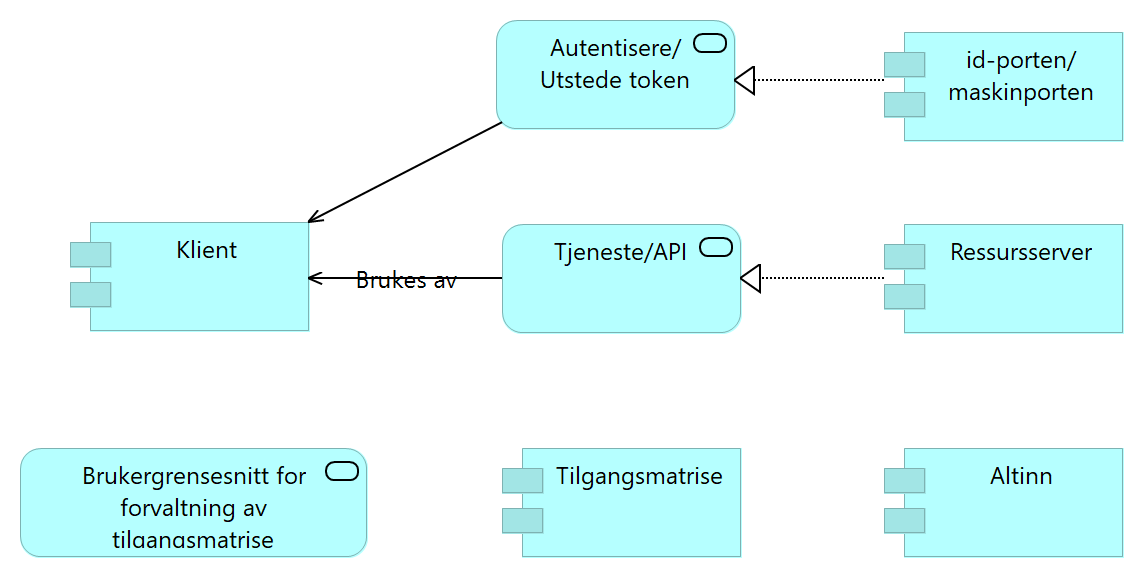
I tillegg til komponentene vist i figuren i kapittel 2.1 trenger vi noen ekstra:

* Tokentjeneste (id-porten/maskin-porten): tjeneste som autentiserer klienten gjennom en tilgjengelig mekanisme (for eksempel virksomhetssertifikat) og utsteder et token.
* Altinn: en sentral funksjon i dagens altinn er at en person kan logge på som seg selv og gjennomføre tjenester på vegne av en virksomhet han/hun representerer

De sentrale spørsmålene i diskusjonen om løsning er:

* Hvor skal tilgangsmatrisen lagres?
* Hvordan skal tilgangsmatrisen forvaltes? Og da spesielt spørsmålet om å autorisere at en driftsleverandør gjør kall til en bestemt tjeneste på vegne av en konsument. Formelt sett er det konsumenten som må godkjenne dette, men vi har antatt over at konsumenten ikke har mulighet til å kalle et API for å gjøre en slik godkjenning, så det må være en eller annen form for brukergrensesnitt.

Etter en utvidelse av den opprinnelige modellen har vi nå følgende komponenter:



For implementering av brukergrensesnitt og tilgangsmatrise finnes tre alternativ:

## Implementering av tilgangsmatrise

|  |  |  |
| --- | --- | --- |
| I komponent/ aktør | Fordeler | Ulemper |
| API-tilbyder | API-tilbyder må uansett ta ansvar for mye av tilgangsstyringen, og det vil dermed være enkelt å legge denne tabellen nær ressursserveren. | Mye av logikken rundt forvaltning av tilgangsmatrisen blir duplisert på tvers av API-tilbydere.  Man får ikke mulighet til å gjøre grovkornet filtrering av kall (basert på tilgangsmatrisen) i maskinporten |
| Difi/ maskin­porten | Sentralisering av funksjonalitet alle trenger.  Mulighet for å legge autoriserings-informasjon inn i token, samt la være å utstede tokens for klienter som forsøker å kalle tjenester den ikke skal ha tilgang til | Faktisk policy for hvem som skal ha tilgang til hva er forvaltet gjennom avtaler som Difi ikke er en del av, og det må implementeres synkronisering av konfigurasjon for å oppdatere tilgangsmatrisen. |
| Altinn | Dette ligger nært opp til funksjonalitet Altinn allerede har, samt vil også muliggjøre mer sofistikerte scenarier med altinn som en «token-exchange» eller kombinasjon med samtykke. | Dette er ikke funksjonalitet som finnes i dag, og det må i så fall inn i altinn sin strategi frem mot 2021.  I tillegg involverer dette ytterligere en komponent i tilgangsstyringen som øker risiko for nedetid og øker responstiden for tjenestekallet (pga ytterligere nettverkshopp) |

## Konsumentens forvaltning av tilgangsmatrise

Det er antatt at for alle alternativene listet over, kan tilgangsmatrisen tilby et API som API-tilbyder kan bruke for å styre «sine» deler av konfigurasjonen. Diskusjonen her gjelder hvordan konsumenten kan styre delegering av myndighet til å kalle et gitt API på vegne av seg selv til en driftsleverandør.

|  |  |  |
| --- | --- | --- |
| I komponent/ aktør | Fordeler | Ulemper |
| API-tilbyder, altså at konsumenten henvender seg til API-tilbyder som oppdaterer tilgangsmatrisen på vegne av konsumenten. | API-tilbyder må allerede ha funksjonalitet for å forvalte deler av tilgangsmatrisen, så dette er teknisk sett en enkel utvidelse. | Menneskelige rutiner rundt forvaltningen vil være omfattende, og dette vil dupliseres for alle API-tilbydere. Det vil neppe være aktuelt å bygge en selvbetjeningsløsning for dette, fordi ingen API-tilbydere alene vil ha nok volum til at dette lønner seg. |
| Difi/maskinporten, altså at maskinporten utvides med GUI for konfigurasjon av delegering | Maskinporten kjenner allerede modellen for tilgangsstyring og det er naturlig å legge konfigurasjonen av dette i nærheten av der den brukes. | Utfordringen er å håndtere at en person skal kunne gjøre en slik konfigurasjon på vegne av en virksomhet. |
| Altinn, et skjema i Altinn hvor en person som representerer en virksomhet (for eksempel daglig leder) kan registrere at en driftsleverandør får lov å kalle en gitt tjeneste på sine vegne. | Altinn er bygget for nettopp denne typen dialog med brukere, hvor en person setter opp konfigurasjon på vegne av virksomheten han/hun representerer. | Dette innebærer å utvikle brukerdialogen for godkjenning av driftsleverandør som en altinn-tjeneste, som er ytterligere en komponent som må forvaltes. |
| Driftsleverandør forvalter dette på vegne av sine kunder | Dette gir mulighet for en driftsleverandør å tilby en «total-løsning» hvor alle tekniske løsninger og fordeling av ansvar kan oppsummeres i en og samme avtale. Det er også greit å implementere, i og med at driftsleverandør allerede har mulighet til å kalle API-er. | Dette gir driftsleverandør mulighet til å operere på vegne av hvem som helst, uten at noen av de andre involverte aktørene kan verifisere hvorvidt denne leverandøren faktisk har juridisk myndighet. Denne løsningen bør derfor støttes av kompenserende tiltak som at:   * Det er bare noen driftsleverandører som får lov til dette * Difi eller API-tilbyder får tilgang til dokumentasjon av avtaler mellom konsument og driftsleverandør for å kunne revidere at tilgangen faktisk er hjemlet i en avtale. |

## Realistiske kombinasjoner

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Lagring | | |
|  |  | API-tilbyder | Maskinporten | Altinn |
| Forvaltning | API-tilbyder | 4 | 5 |  |
| Maskinporten |  | 6 |  |
| Altinn |  | 1 | 2 |
| Driftsleverandør |  | 3 |  |

1. Lagring hos maskinporten, forvaltning gjennom altinn
2. Lagring hos altinn, forvaltning gjennom altinn
3. Lagring hos maskinporten, forvaltning gjennom driftsleverandør
4. Lagring og forvaltning hos API-tilbyder
5. Lagrings hos maskinporten, forvaltning gjennom API-tilbyder
6. Lagring og forvaltning hos maskinporten

Merk at 1, 3, 5 og 6 alle innebærer at hele tilgangsmatrisen lagres i maskinporten, at det tilbys et API for å forvalte denne, men at forvaltningen gjøres fra ulike parter. Det vil være mulig å støtte alle disse i kombinasjon, gjennom å la flere parter bruke API-et, evt med ulike forutsetninger og tilganger for hva de får lov å gjøre.

# Eksempler og scenarier

## Eksempel på forhold mellom «L» og «C»

1. Et selskap har kjøpt regnskapssystem av Visma og installerer dette på egen HW. Regnskapssystemet gjør et oppslag i BR på vegne av selskapet. I denne sammenhengen er «selskap=C» og det finnes ingen L. Regnskapssystemet autentiseres som C.
2. Et selskap har kjøpt regnskapssystem av Visma og installerer dette i Azure. Her er «Selskap=C» og «Microsoft=L». Jeg tror dette innebærer at Microsoft er databehandler og må ha en databehandleravtale med Selskap. Men regnskapssystemet autentiseres fortsatt som C.
3. Selskapet har ikke kjøpt en egen lisens på Visma, men kjører «visma-as-a-service», altså at regnskapssystemet er installert på Visma sin hardware i en delt løsning (samme installasjon betjener mange andre visma-kunder), men selskapet fører regnskapet sitt selv, Visma leverer bare software (as-a-service). Her er «selskap=C» og «visma=L». I denne sammenhengen er det et valg om regnskapssystemet autentiseres som Visma (L), eller om det har egne nøkler/sertifikat for hvert selskap og autentiserer som C, men jeg mener det riktige her er at regnskapssystemet autentiserer som Visma (L)
4. Selskapet har ikke kjøpt programvare i det hele tatt, men leier Visma til å føre regnskapet for seg. Her er fortsatt «selskap=C» og «visma=L» men jeg mener det er helt åpenbart at her må regnskapssystemet autentiseres som Visma (L). Selskapet (C) har ingen befatning med programvaren.

## Eksempel på tilgangsmatrise

Tilgangsmatrisen ser slik ut (med et par eksempler fylt ut):

|  |  |  |  |
| --- | --- | --- | --- |
| Konsument, C | Driftsleverandør L | Tjeneste, API | API-tilbyder |
| Valle sparebank | Eikagruppen | Hente ligningsopplysninger | Skattedirektoratet |
| DNB |  | Hente ligningsopplysninger | Skattedirektoratet |
| Bisnode |  | Løsøreregisteret | Brreg |
| Skattedirektoratet |  | Kontoopplysninger | BITS (for lille leikanger sparebank) |
| Skattedirektoratet |  | Kontoopplysninger | Storbanken AS |
| Leikanger kommune | Visma Flyt barnehagesystem | Ledige barnehageplasser | Private barnehagers landsforbund (for Hermansverk barnehage AS) |

I eksempelet er det formelt sett slik at:

* Det er skattedirektoratet som kan bestemme at systemer hos hhv «valle sparebank» og «dnb» kan kalle «hente ligningsopplysninger»
* Det er valle sparebank som kan bestemme at eikagruppen skal kunne kalle tjeneste-X på vegne av valle sparebank, men når eika sitt system autentiserer seg mot difi-maskinporten, blir systemet autentisert som «eikagruppen» (ikke «valle sparebank»)

Merk at dette gjelder bare hvem som har myndighet til å ta hvilke beslutninger, ikke hvor dette lagres, vedlikeholdes eller håndheves. Merk også at det kan være mer finkornede regler for tilgangsstyring internt i serveren enn hva som fremkommer i tabellen