Direktoratet for e-helse

Veileder for utvikling av datadelingsgrensesnitt

HITR 1221:2019

Publikasjonens tittel:

Veileder for utvikling av datadelingsgrensesnitt

Rapportnummer
HITR 1221:2019

Utgitt:
03/2019

Utgitt av:
Direktoratet for e-helse

Kontakt:
postmottak@ehelse.no

Besøksadresse:
Verkstedveien 1, 0277 Oslo
Tlf.: 21 49 50 70

Publikasjonen kan lastes ned på:
www.ehelse.no

Forord

Dagens samhandling baserer seg først og fremst på meldingsutveksling. På ett døgn sendes det ca. 1,7 mill. elektroniske meldinger i helsenettet. Samtidig er det et økende behov for å ta i bruk nye samhandlingsformer, slik som data- og dokumentdeling, for å dekke samhandlingsbehovene i helse- og omsorgssektoren.

For å unngå behovet for å "rydde opp" om noen år ble prosjekt FIA Data- og dokumentdeling prioritert i 2018 for å tilrettelegge for at aktørene i helse- og omsorgssektoren skal kunne ta i bruk data- og dokumentdeling på en enhetlig måte. Som et ledd i dette arbeidet ble *veileder for utvikling av datadelingsgrensesnitt* utarbeidet.

Stadig flere virksomheter tar i bruk datadeling i samhandlingen. Det er i den forbindelse behov for nasjonalt anbefalte kommunikasjonsrammverk for datadeling. Aktørene i helse- og omsorgssektoren har i dag ulikt syn på hvilke kommunikasjonsrammeverk som skal brukes i sektoren. Manglende nasjonale anbefalinger åpner opp for at aktørene i større grad kan fremme egne preferanser. Dette bidrar til lavere grad av standardisering og som en konsekvens kan det lede til forsinkelse i utviklingen av nye tjenester og økt kompleksitet i IKT-landskapet.

Denne veilederen presenterer nasjonalt anbefalte kommunikasjonsrammeverk for datadeling og har som mål å bidra til at sektoren skal kunne ta i bruk datadeling på en mer enhetlig måte. Prosjektet FIA Data- og dokumentdeling har som mål å bidra til å bygge opp under sektorens strategiske mål; Helsepersonell skal ha enkel og sikker tilgang til pasient- og brukeropplysninger, innbyggere skal ha tilgang på enkle og sikre digitale tjenester, og data skal være tilgjengelig for kvalitetsforbedring, helseovervåking, styrking og forskning.

Innhold

1 Innledning

Erfaringer fra tidligere prosjekter er at når samhandlingsbehov skal implementeres mellom virksomheter, diskuteres ofte teknologivalg uten å ha fokus på hvilke behov man skal løse. For å løfte diskusjonene opp fra teknologilaget har Direktoratet for e-helse definert noen generiske samhandlingsmønstre som er gjenkjennbare for ulike interessenter som jobber med samhandling mellom virksomheter i helse- og omsorgstjenesten. Disse mønstrene kan enklere knyttes til de forretningsmessige behovene for samhandling gjennom å analysere egenskapene til samhandlingsbehovene og vurdere hvilket samhandlingsmønster som dekker egenskapene best.

Samhandlingsmønstrene har vi valgt å kalle for samhandlingsmodeller og disse er identifisert:

- Meldingsutveksling – overføring av strukturerte data til kjent mottaker (som en del av en automatisk prosessering)
- Dokumentutveksling– overføring av godkjent, lesbart dokument, med varierende grad av struktur. I Norge har vi ofte omtalt dokumentutveksling som meldingsutveksling, og en elektronisk melding kan derfor dekke både meldingsutveksling og dokumentutveksling
- Dokumentdeling – deling av godkjent, lesbart dokument gjennom felles infrastruktur/tjenester
- Datadeling - deling av eller samarbeid om strukturert informasjon gjennom felles ressurser/tjenester
- Direkte tilgang - (Innsyn) til fellesløsning eller løsning i en annen virksomhet
- Samhandling i felles kjernesystem

Samhandlingsmodellene kjennetegnes ved at de dekker ulike generiske samhandlingsbehov i helsesektoren, både mellom virksomheter og mellom virksomheter og innbyggere. Samhandlingsmodellene stiller krav til hvordan virksomhetene må integrere sine løsninger seg imellom eller mot nasjonale fellesløsninger for å oppnå god samhandlingsevne. Det er ikke nødvendigvis klare skiller mellom de ulike samhandlingsmodellene. Flere nasjonale- og internasjonale standarder og -teknologier vil kunne benyttes i flere av samhandlingsmodellene.

Denne veilederen omhandler samhandlingsmodellen datadeling. Ved bruk av datadeling forventer brukere at samhandlingen skjer i sanntid. Datadeling realiseres gjennom å etablere datadelingsgrensesnitt som en helseaktør tilgjengeliggjør for andre helseaktører.

Datadelingsgrensesnitt gjøres tilgjengelig for andre aktører gjennom bruk av webteknologi. Datadelingsgrensesnitt må støtte både lesing, registrering, sletting og oppdatering av helse- og personopplysninger. Et datadelingsgrensesnitt kan realiseres som et web API.

1.1 Bakgrunn

Historisk har web API vært synonym med SOAP baserte webservices (Simple Object Access Protocol), men den senere trend har medført at mange har gått fra å bruke SOAP og

tjenestebasert arkitektur (SOA) til direkte «representational state transfer» (REST) baserte webservices (RESTful API-er) og «Resource Oriented Architecture» (ROA). Dette er nærmere beskrevet i kapittel om "ressursorientert arkitektur".

Selv om SOAP-baserte grensesnitt er profilert nasjonalt, foreligger det ikke en nasjonal anbefaling for bruk av kommunikasjonsrammeverk til datadeling.

Denne veilederen skal gi en beskrivelse av anbefalinger ved implementering av datadelings-grensesnitt i helsesektoren. Dette har vært særlig etterspurt i helsesektoren og da spesielt når det gjelder bruk av REST.

FIA Arkitekturstyring gjennomført i 2017 en kartlegging av aktuelle kommunikasjonsramme-verk for datadeling. Kartleggingen resulterte i en rekke kandidater som det var behov for å utdype, vurdere og gjøre et nedvalg av sammen med sektoren. *Fastsetting av nasjonalt anbefalte kommunikasjonsrammeverk for datadeling* ble etablert som en av leveransene til prosjektet FIA Data- og dokumentdeling. Veilederen er leveransen fra denne aktiviteten og er en anbefaling for hvilke kommunikasjonsrammeverk helsesektoren bør benytte ved realisering av datadelingsgrensesnitt.

1.2 Om veilederen

Veileder for utvikling av datadelingsgrensesnitt er utarbeidet av prosjektet FIA Data- og dokumentdeling. Veilederen understøtter FIA Data- og dokumentdelingsprosjektets resultatmål 3: *Etablere praktisk bruk av referansearkitektur for data- og dokumentdeling, nasjonale retningslinjer og internasjonale standarder ved innføring av løsninger for data- og dokumentdeling*. Veilederen ble utarbeidet som en del av arbeidet med å *tilrettelegge for at sektoren kan ta i bruk data- og dokumentdeling på en enhetlig måte.*

Arbeidet med nedvalg av kommunikasjonsrammeverk og diskusjon av anbefalingen ble utført sammen med prosjektets arbeidsgruppe som består av representanter fra helsesektoren. Basert på dette arbeidet beskriver veilederen en anbefaling av REST/RESTful, FHIR-basert REST og SOAP basert WS grensesnitt som de nasjonalt anbefalte tekniske standardene som bør velges i nye prosjekter.

1.3 Målgruppe

Målgruppen for veilederen er leverandører og prosjekter i helsesektoren som planlegger eller gjennomfører realisering av datadelingsgrensesnitt.

2 Kommunikasjonsrammeverk for datadeling

Et kommunikasjonsrammeverk er tilpasset de samhandlingsbehov den skal støtte. Et kommunikasjonsrammeverk for datadeling må støtte overføring av informasjon i sanntid slik at brukere ikke opplever forskjell på å arbeide med lokalt lagret informasjon og informasjon lagret hos andre virksomheter.

Overføring av informasjon i sanntid har normalt færre feilsituasjoner som en løsning må håndtere enn andre former for kommunikasjon, slik som asynkron meldingsutveksling. Kommunikasjonsrammeverk for datadeling er derfor mindre kompleks enn for eksempel et kommunikasjonsrammeverk for meldingsutveksling.

Datadeling er nært knyttet til det historiske konseptet "remote procedure call" som er en type samhandling mellom to prosesser som kjører på to ulike systemer. Ved "remote procedure call" kreves det at prosessene må ha et felles språk for å sende og motta kommandoer og informasjon. Kommandoen kan sees på som formålet med hvorfor den ene prosessen kontakter den andre. Gjennom historien har det eksistert mange ulike språk for å oppnå dette, men mange av språkene krevde sikkerhetsmessige tiltak for å tillate prosessene i å snakke sammen. Etter hvert ble protokollen http svært populær da protokollene krevde få sikkerhetsmessige tiltak for å tillate prosessene i å snakke sammen.

Et kommunikasjonsrammeverk for datadeling må også ha støtte for å sende og motta kommandoer og informasjon. Basert på erfaringer og utbredelse er det en forutsetning for kommunikasjonsrammeverkene for datadeling at de er basert på protokollen http.

Datadeling er en samhandlingsform som skjer mellom to systemer i to forskjellige virksomheter. Kommunikasjonen foregår over nettverk som ikke virksomhetene har kontroll over og må skje på en sikker måte. Et kommunikasjonsrammeverk må ha støtte for å sikre konfidensialitet, integritet og tilgjengelighet over slike usikrede nettverk.

For kommunikasjon over http så finnes det to utbredte kommunikasjonsrammeverk, REST og SOAP over http.

REST

REST er ingen standard i seg selv, men består av ulike prinsipper som former en arkitekturstil for samhandling [1]. REST er normalt knyttet til bruk av protokollen http, men er i teorien ikke begrenset til dette.

SOAP over http

SOAP over http er en standard som defineres som en enkel protokoll beregnet for utveksling av strukturert informasjon i et desentralisert, distribuert miljø. Den bruker XML-teknologier til å definere et utvidbart rammeverk som gir en meldingskonstruksjon, og som kan utveksles over protokollen http. Rammeverket er utformet for å være uavhengig av en bestemt programmeringsmodell og annen implementeringsspesifikk semantikk.

2.1 Ressursorientert arkitektur versus tjenesteorientert arkitektur

Datadelingsgrensesnitt kan realisere både i en tjenesteorientert arkitektur (SOA) og i en ressursorientert arkitektur (ROA).

I mange tilfeller er ofte et for stort fokus på det tekniske ved design og implementering av web API-ene. Likevel er det slik at SOA og ROA er to ulike arkitekturmønstre som ikke bare påvirker design av API-ene, men også hele løsningsarkitekturen. Det anbefales derfor å ha et mer fokus på dette valget og hvilke konsekvenser valget har for løsningsarkitekturen.

SOA velges når en virksomhet ønsker å tilby forretningstjenester og tilgjengeliggjøre de for andre som web API-er. Dette er også omtalt som RPC - Remote Procedure Calls. Eksempler på forretningstjenester: "hent ledige timer", "bestill time", "avbestill time".

I en ROA arkitektur ønsker en virksomhet å samarbeide med andre om felles ressurser (informasjon). Virksomheten som deler vil da tilby API-er for å behandle ressursene.

Eksempel på et samarbeid om felles ressurs er at en virksomhet tilbyr andre virksomheter å behandle ressursen "Avtale" gjennom en forhåndsdefinert liste med operasjoner: les (GET) som normalt henter en liste av ressurser; opprett (POST) som oppretter en ny representasjon av en ressurs; oppdater (PUT) oppdaterer en ressurs med egen id; og slett (DELETE) som sletter en gitt ressurs.

REST er som nevnt ikke en standard. Det er en kjent utfordring internasjonalt at begrepet benyttes i tekniske miljøer i en videre betydning. Det kan dermed være en utfordring at det er uklart hva bruk av REST innebærer for en gitt løsning. Det er i den sammenheng utarbeidet en modenhetsmodell for REST-baserte API-er [2] som definerer fire nivåer.

Det laveste modenhetsnivået i modellen er anvendelser av REST hvor det benyttes http kun som en transportkanal og det benyttes "forretningstjenester" som kommandoer mellom løsningene som samhandler. Dette nivået kan sies å være knyttet til arkitekturvalget tjenestebasert arkitektur (SOA).

Fra det andre modenhetsnivået i modellen benyttes kommandoene som er innebygget i http og dette tvinger de samarbeidende løsningene å samarbeide om ressursene (ROA).

Ved bruk av REST anbefales det derfor at valg av nivå i modenhetsmodellen er i samsvar med valg av arkitekturstil (SOA eller ROA).

Fordeler og ulemper med ROA og SOA

Ulikhetene mellom ROA/REST og SOA/SOAP baserte arkitekturer:

1. REST benytter forutsigbare kommandoer for å manipulere informasjon på.
    a. Med REST kan du hente, lage nye, oppdatere og fjerne informasjon
    b. Denne måten å manipulere informasjon på er svært utbredt og enkelt for utviklere å implementere.
    c. De forutsigbare kommandoene medfører også en begrensning av mulighetsrommet til utviklere og tvinger designet av samhandlingsløsninger til å tenke ROA. Dette gir enklere integrasjoner som er lettere å vedlikeholde.
2. ROA gir en arkitektur hvor klientene har mer styring over selve samhandlingslogikken
    a. Når et web API fullt ut implementerer REST, vil det være mulig for klientene å manipulere alle typer ressurser som de har tilgang til. Det vil være opp til klientene hva de ønsker å støtte.
    b. En SOA tjeneste må ofte tilpasses behov som kun fåtall av klientene har. Web API-er må i liten grad tilpasses slike behov og det er derfor enklere å legge til nye klienter.
3. ROA gir mindre behov for dobbellagring
    a. REST baserte web API-er gir klientene små, men presise mengder informasjon som ikke gir behov for å lagre dataene lokalt hos klientene.
4. REST baserte web API-er kan støtte flere formattyper for innhold enn SOAP, slik som XML og JSON.

   a. Det er enklere å utvikle grensesnitt som benytter JSON.
   b. Flere applikasjoner på mobile enheter foretrekker kombinasjonen REST+JSON på grunn av raskere overføring og parsing.

5. SOAP over http baserte web API-er håndterer store strukturerte dokumenter mer effektivt. Ved bruk av REST er det normalt nødvendig å sette sammen mange ressurser til et dokument og dette er ofte ikke REST tjenere optimalisert for.
6. I en SOA-arkitektur kan forretningstjenester designes slik at mange operasjoner eksekveres i et kall. ROA er optimalisert for kommunikasjon med små mengder data og dette vil medføre at klientene må spørre mange ganger for å oppnå ønsket måltilstand.
7. I en SOA basert arkitektur overlates orkestrering av en tjeneste som består av mange operasjoner til tjenerne. I ROA så overlates "orkestreringen" til klientene. Klienten må selv implementere logikk for å håndtere konversasjoner (flere kall for å oppnå ønsket måltilstand). Det er derfor høyere risiko for at ulike klienter som benytter samme REST API-ene gjør det på feil måte eller på en ikke ensartet måte som blant annet kan gå ut over datakvaliteten.
8. Med "contract-first" mønster og bruk av XML er SOAP over http lettere å standardisere. REST er mer fleksibelt og det kan være behov for å bruke mer tid på standardisering. REST gir utviklere stor frihet til hva de implementerer av støtte for operasjoner (GET, PUT, POST, DELETE) og hvilke responser og responskoder som en tjener benytter. Ved liten grad av standardisering er det større risiko for at klienters feil bruk av et REST-API kan få uforutsette konsekvenser. Nasjonal bruk av REST baserte API-er krever standardisering. Den internasjonale standarden HL7 FHIR har langt på vei standardisert bruken av REST. Bruk av OpenAPI for å spesifisere REST-API-er gir bedre standardisert bruk av API-ene, mer om dette i kapittel 9.
9. Ved behov for implementering av forretningstjenester er bruk av SOAP over http mer egnet enn REST. REST har en sterk kobling til http-protokollen (GET, PUT, POST, DELETE) og dette kan i noen kontekster være begrensende. I de tilfeller hvor det er behov for å designe tjenesteorienterte grensesnitt og det ikke er ønskelig å velge annet enn REST, må grensesnittet tilby tjenester (laveste nivå i modenhetsmodellen til REST). Det eksisterer flere alternative metoder for implementering av tjenester med REST. I kapittel 8 har vi beskrevet anbefalte metoder.

3 Anbefaling på bruk av kommunikasjonsrammeverk

Det er identifisert seks internasjonale kommunikasjonsrammeverk som kan benyttes for datadeling. Direktoratet for e-helse har i samarbeid med sektoren foretatt et nedvalg av antall anbefalte kommunikasjonsrammeverk for å forenkle oppnåelsen av teknisk samhandlingsevne mellom virksomheter i sektoren, samtidig som virksomhetene får færre standarder å forholde seg til.

I tabellen under er de anbefalte kommunikasjonsrammeverkene for etablering av datadelingsgrensesnitt oppsummert.

Anbefalte kommunikasjonsrammeverk	Beskrivelse
REST/RESTful	Bygger på HTTP. Bruke alle tilgjengelige HTTP verb til å overføre XML/JSON baserte ressurser over http protokoll, og med unik identifiserbare ressurser i URLen. Minst nivå 2 på REST maturity model [2]
FHIR-basert REST	Bygger på REST/RESTful gruppen, men legger til FHIR-baserte utvidelser knyttet til teknisk samhandlingsevne, slik som: • operations [3] i URL, • bruk av bundles (sette sammen ressurser til en melding) • støtte for å håndtere endringer på flere ressurser i en enkel transaksjon [4]. • kapabilitetstjeneste som forteller klienter hvordan serveren støtter ressurser [5].
SOAP baserte webservice	Bygger på HTTP. Standard webservices ved hjelp av SOAP standarden, samt utvidelse for å håndtere konfidensialitet og integritet (ws-security)

Anbefaling for bruk at REST/RESTful

Direktoratet for e-helse anbefaler at REST på minimum nivå 2 på REST maturity model [2] kan brukes i helsesektoren. REST på nivå 0 hvor det ikke benyttes ressurser anbefales ikke.

Det oppfordres til å følge anbefalingene som er beskrevet i denne veilederen samt i Normens Faktaark 24 [6].

Anbefaling for bruk av FHIR-basert REST

FHIR-basert REST som er en utvidelse av REST kan brukes fritt i helsesektoren – men det oppfordres til å følge standarden samt anbefalingene som er beskrevet i denne veilederen og i Normens Faktaark 24 [6].

Anbefaling for SOAP baserte WS (SOAP over http)

SOAP over http oppfordres til å ikke tas i bruk på nye anvendelsesområder dersom bruken ikke er knyttet til nasjonale eller internasjonale standarder og/eller nasjonale anvendelser. IHE XDS er et eksempel på en internasjonal standard som benytter SOAP og Reseptformidlerens grensesnitt er et eksempel på en nasjonal anvendelse.

4 Krav til transportsikkerhet og bruk av TLS

Ved overføring av helse- og personopplysninger over åpne nett skal opplysningene sikres mot at uvedkommende får kjennskap til opplysningene. I tillegg skal overføringen være sikret mot utilsiktet eller uautorisert endring eller sletting. Normens kapittel. 5.5.2 "Sikring av nettjenester" sier:

Minst to uavhengige, tekniske tiltak skal iverksettes slik at personer utenfor virksomheten ikke skal kunne få uautorisert tilgang til og/eller kunne endre eller slette helse- og personopplysninger

All overføring av helse- og personopplysninger ved bruk av datadeling over åpne nett må derfor alltid krypteres slik at innholdet i overføringen alltid er uleselig for andre enn mottaker av opplysningene.

Som et minimum skal transportkanalen krypteres. Dette er beskrevet mer i detalj i Normens faktaark 24 som viser til Nasjonal Sikkerhetsmyndighet sin anbefaling om bruk av TLS (Transport Layer Security) [7]:

"*Nasjonal Sikkerhetsmyndighet anbefaler at TLS benyttes i størst mulig grad, da protokollen er tilgjengelig i mange ulike systemer og tjenester."*

TLS er benyttes for å sette opp en sikker kommunikasjonskanal mellom klient og server for overføring av informasjon. NSM anbefaler bruk av siste versjon av TLS (som per juni 2018 er versjon 1.3). Andre metoder for å kryptere kommunikasjonskanaler er VPN (Virtuelt Privat Nettverk og IPSec (Internet Protocol Security).

Bruken av TLS-protokollen deles opp i to faser [8]:

1) Etablering av sikker forbindelse mellom klient og tjener

a. Valg av protokoll og kryptografiske algoritmer
b. Autentisering av server (og eventuelt klient) basert på sertifikat
c. Utveksling av kryptografiske parametere
d. Etablering av krypteringsnøkkel for sesjonen

2) Utveksling av applikasjonsdata over den sikre forbindelsen

Første fase av oppkoblingen skjer hovedsak i klartekst. Det betyr at i tillegg til informasjon som IP-adresser, kan den offentlige delen av sertifikater som utveksles i etableringsfasen avlyttes.

Når TLS er etablert for en tjeneste, vil informasjonen overføres uten innsyn for andre. TLS gir samme beskyttelse mot avlytting på lokale nett, som mot avlytting på Internett.

Bruk av klientsertifikat kan være krav hos enkelte tjenestetilbydere. En tjenestetilbyder må kunne knytte en klient til en virksomhet og ha logikk (selv eller hos en tiltrodd tredjepart slik som f.eks. HelseID) for å kontrollere om det foreligger avtale med virksomheten. Et klientsertifikat kan ha en slik funksjon.

HelseID tilbyr en tjeneste for sikring av datadelingsgrensesnitt. Som en del av denne tjenesten må HelseID ha en oversikt over hvilke klienter som har anledning til å bruke ulike grensesnitt. Når en EPJ på et legekontor skal integreres med nasjonal kjernejournal brukes HelseID for å utstede sikkerhetsbilletter alle kan stole på. Dette gjør det mulig å realisere f.eks. tett integrasjon mellom nasjonal kjernejournal og EPJ. Veiledning for hvordan dette virker finnes her.

5 Krav til tilgjengelighet

I henhold til Personvernforordningens artikkel 32 skal det gjennomføres egnede *tekniske og organisatoriske tiltak for å oppnå et sikkerhetsnivå som er egnet med hensyn til risikoen*. Dette for blant annet å sikre evnen til vedvarende tilgjengelighet til helse- og personopplysninger. Det skal også gjennomføres tiltak for å gjenopprette tilgjengeligheten og tilgangen til helse- og personopplysninger i rett tid dersom det oppstår en fysisk eller teknisk hendelse. Artikkel 32 sier også at det skal opprettes en prosess for regelmessig testing, analysering og vurdering av hvor effektive behandlingens tekniske og organisatoriske sikkerhetstiltak er.

Bruk av sanntidskommunikasjon, slik som datadeling, er sårbar for utilgjengelighet og brukere forventer tilgjengelige systemer. Det er viktig å sørge for at løsninger som tilbyr tilgang til helse- og personopplysninger til andre virksomheter er robuste og har høy oppetid slik at brukerne og virksomhetene har tillit til tjenesten. Når mange klienter benytter en tjeneste, er det også behov for innføring av flere tiltak. Fellestjenester med høy utbredelse krever svært høye krav til tilgjengelighet.

En robust tjeneste med høy tilgjengelighet kan oppnås på mange måter. Eksempler på tiltak kan være:

- Redundante servere, databaser, applikasjonsservere osv.
- Overvåking av komponenter og trafikk med gode rutiner for å rette feil.
- Gode ikke-funksjonelle testrutiner slik som gjennomføring av volum-, stress-, avbruddstesting osv som tester robusthet til systemet.
- Oversikt over hvem som bruker tjenesten og ha gode rutiner for varsling av planlagt vedlikehold, endringer i tjenesten og øvrige driftsvarsler.

Det er ikke bare de virksomheter som tilbyr tjenester som må innføre tiltak for å sikre høy tilgjengelighet. Virksomheter som benytter andre virksomheters datadelingsgrensesnitt må ha rutiner for hvordan brukere skal forholde seg til utilgjengelighet av eksterne tjenester. Eksempler på vurderinger som disse virksomheter kan gjøre er:

- Hvor kritisk er det for brukerens arbeidsoppgaver dersom tjenesten er utilgjengelig?
    - Skal brukeren kunne fortsette sine oppgaver dersom tjenesten er utilgjengelig?
- Bør det etableres reserveløsninger dersom utilgjengelighet ikke kan aksepteres, eller holder det med manuelle rutiner?
- Hvordan skal rutiner for overvåking og oppfølging av utilgjengelighet være?

6 Bruk av pasient ID i webadresser

Webadresse, eller en URL er en referanse til en web basert ressurs. REST-basert kommunikasjon benytter URL-er til å adressere ressurser. I en helseorientert applikasjon som benytter REST, kan f.eks. pasient være en ressurs. En URL som søker etter en bestemt Pasient-ressurs kan være slik: (forenklet):

GET https://SystemA.sykehus.no/FHIR/Patient?identifier=12345678901
(hvor "12345678901" er et fødselsnummer)

Dersom Pasient-ressursen med det forespurte fødselsnummer finnes, returneres ressursen. Ved bruk av REST har normalt alle ressurser egne id-er (ressurs-id) som må benyttes ved manipulering av en ressurs. Et søk på om en pasient har en bestemt diagnose kan også forekomme som et kall slik som dette:

GET https://SystemA.sykehus.no/FHIR/Patient/a1234-b1234-c1234/_search?subject="Krefttype"

Dersom koblingen mellom ressurs-id og fødselsnummeret er kjent, kan dette medføre i ytterste tolkning at innholdet i disse to URL-er er personsensitiv informasjon.

Ved bruk av REST og transportsikring (TLS) vil URL-ene være kryptert. Ved etablering av en TLS sesjon vil DNS-navene på klient og en tjener bli utvekslet i klartekst, men ikke selve URL-en. Det er derfor liten risiko for at personopplysninger i URL-ene vil kunne leses på veien.

Det er derimot vanlig å benytte mellomtjenere i flere ledd. Hver av mellomtjenere har som standard logger som logger URL-ene som aksesseres. Tilgang til disse loggene må sikres slik at uvedkommende ikke får aksess til dem.

Basert på forskrift om informasjonssikkerhet §5-5 så anbefaler Nasjonal Sikkerhetsmyndighet at prinsippet om minimalisme bør gjelde for bruk av fødselsnummer i

URL og deres anbefaling er:

Dersom det ikke er en tungtveiende grunn til at fødselsnummer må benyttes i URL-er, bør det unngås (NSM).

Direktoratet for e-helse anbefaler:

Dersom risikoen for at sensitive personopplysninger i tekniske logger til mellomtjenere kan komme på avveie er stor, bør fødselsnummer ikke benyttes i URL.

Det anbefales to alternative tiltak for å redusere denne risikoen:

1. Unngå å benytte fødselsnummer i en GET forespørsel til en REST tjeneste, ved å skrive om tjenesten slik at det er mulig å bruke POST. Da vil fødselsnummer overføres som en del av payloaden til http forespørselen. Dette er støttet i FHIR [14]

Dersom det allikevel er nødvendig å benytte fødselsnummer i URL, så kan følgende tiltak redusere risikoen for at sensitive personopplysninger kommer på avveie:

1. Unngå logging av "søkestreng". Når fødselsnummer benyttes i en URL, vil nummeret overføres som en del av "søkestrengen" (Query String). Mange mellom- og webtjenere har mulighet til å slå av logging av "søkestrengen".
2. Alternativt kan pseudonym for fødselsnummer benyttes. Personvernforordningen omtaler psedonymisering og Datatilsynet har gitt ut en veileder om anonymisering av personopplysninger [9] hvor Pseudonymisering er omtalt og definert slik: *«**Pseudonymisering**» vil si at enkelte direkte identifiserende parametere erstattes med pseudonymer, som fremdeles vil være unike indikatorer".*

7 Transaksjonshåndtering

En transaksjon er et sett med operasjoner som anses som den minste enheten for arbeid for et system. En transaksjon skal enten fullføre alle sine endringer på systemet (commit) eller avbryte (abort, rulle tilbake) om transaksjonen opplevde problemer. Vi sier en transaksjon er atomisk, enten fullføres den i sin helhet - eller ikke i det hele tatt.

Håndtering av flere ressurser i et kall

API-er innad i en applikasjon har normalt støtte for transaksjoner. Datadelingsgrensesnitt som benytter http som transportkanal har ikke samme transaksjonsstøtte. Det mangler støtte for å automatisk koordinere klientens transaksjoner med transaksjonene til tjeneren (distribuerte transaksjoner).

I webservice-verden finnes noen standarder slik som WS-transaction, men det er lite utbredt.

FHIR har et konsept hvor klienten kan samle flere operasjoner i en Bundle [10] hvor tjeneren håndterer alle operasjonene som en transaksjon.

Ellers i REST-verden finnes ingen standarder for transaksjonshåndtering. Dette vil si at klientene selv må implementere kompenserende logikk når feil i kall til datadelingstjenester oppstår. Dette er særlig en utfordring når klienter må gjøre mange kall på ulike ressurser for å oppdatere tilstanden til tjeneren. Dersom disse oppdateringene ansees av klienten til å være en atomisk operasjon, må klienten selv håndtere koordineringen av alle feilsituasjoner. Et eksempel er når en klient må oppdatere tre ressurser og klienten anser oppdateringene som en transaksjon (en atomisk operasjon). Dersom siste kallet som oppdaterer ressurs nr. 3 feiler, er det klientens ansvar å sørge for at denne blir oppdatert senere eller sørger for å tilbakestille de andre oppdateringene.

Et løsningsalternativ er å benytte samme teknikk som i FHIR. Tjeneren åpner opp for at klienter kan sende inn flere ressursoppdateringer i samme kall (Bundle i FHIR [10]) slik at tjeneren håndterer alle ressursoppdateringene i en transaksjon.

Idempotente tjenester

En vanskelig feilsituasjon innen datadeling er når en klient mottar et tidsavbrudd på sitt kall til en tjener. Utfordringen med dette er at klienten da ikke vet om forespørselen har gått igjennom eller ikke. Det er normalt å implementere at klienten sender inn samme forespørsel på nytt. Dersom forrige forespørsel som "timet" ut gikk igjennom, og det var svaret tilbake til klienten som feilet, vil det oppstå en uavklart tilstand mellom klient og tjener når klienten sender samme forespørsel på nytt.

Et eksempel på dette er når en klient sender inn en forespørsel om å opprette en ny ressurs. Dersom forespørselen timer ut vil det være naturlig at klienten prøver på nytt ved å sende samme forespørsel på nytt. Dersom tjeneren faktisk opprettet ressursen i det første kallet, vil nå tjeneren sende en feilmelding om at ressursen allerede finnes. Da blir klienten satt i en situasjon hvor den ikke vet om den faktisk har fått utført sin forespørsel eller om det er andre som har opprettet denne ressursen.

Et løsningsalternativ er at tjeneren implementerer sine tjenester slik at den oppdager når den mottar en duplikat-forespørsel og sørger for å svare likt på alle duplikat-forespørsler. Dette kalles idempotente tjenester. I eksemplet beskrevet tidligere så må tjeneren svare det samme på forespørsel nr. 2 som den ville ha svart på forespørsel nr. 1. Da vil klienten vite at ressursen er blitt opprettet.

8 Operasjoner - utvidelse av REST

REST (Representational State Transfer) er en arkitekturstil som benytter seg av HTTP. REST baserer seg på at klienter benytter HTTP-funksjonene *GET, POST, PUT* og *DELETE* for å dekke basisfunksjonene *create, read, update, delete* (CRUD) for persistent lagring. Dette kalles et RESTful API.

Det kan være behov for utvidede operasjoner, på samme måte som RPC og SOAP-baserte Web Services. En slik utvidelse av REST omtales som *execute* (E), slik at en REST-tjeneste da støtter "*CRUDE*".

Ved å ta i bruk operasjoner i et RESTful API vil klienter kunne utføre mer spesialtilpassede tjenester ut over kun CRUD.

Erfaring har vist at det er svært utbredt å utvide RESTful API med operasjoner, men på en ustandardisert måte.

Direktoratet for e-helse anbefaler:
RESTful API-er som ikke benytter FHIR og har behov for operasjoner bør følge FHIR sin standardiserte måte å utvide REST med operasjoner på.

Kjennetegn for slike operasjoner:

- Hver operasjon har et navn
- Hver operasjon har en liste med inn- og ut-parametere
- Parametere er enten ressurser, datatyper eller søkeparametere
- Operasjoner har de samme sikkerhetskrav som RESTful API for øvrig
- URIene for operasjonen endepunkt følger samme adressering som API-et for øvrig
- Operasjoner kan benytte datalageres eksisterende ressursdefinisjoner
- Operasjoner kan utføres på en spesifikk ressurs, en ressurstype eller hele systemet

For nærmere detaljer om FHIR operasjoner, se Extended Operations on the RESTful API [11].

9 API-spesifikasjoner

Standard REST har ingen standardiserte teknikker for å dokumentere en tjeners kapabiliteter på. Ved bruk av SOAP brukes WSDL som en standard måte å beskrive operasjonene som en tjener tilbyr. Den benyttes også ofte som en teknisk kontrakt mellom klienten og tjeneren.

Capability Statement
I FHIR REST er det krav om at alle FHIR servere skal støtte Capability Statement [12]. Capability Statement brukes for at FHIR servere kan dokumentere og uttrykke hvilken funksjonalitet og regler den støtter av FHIR standarden. Dette kan i teorien gi støtte for automatisk konfigurasjon samt tilpasning av klienters oppførsel. Capability Statement bør sees på som et trinn i oppnåelsen av samhandlingsevne mellom klient og tjener.

OpenAPI Specification
OpenAPI initiative jobber for å standardisere hvordan REST API-er er beskrevet på et leverandøruavhengig format. Denne standarden kalles Open API Spec [13]. OpenAPI-spesifikasjonen krever at funksjonene til tjenesten beskrives med strukturen som er definert i standarden. Med OpenAPIs deklarative ressursspesifikasjon kan klienter forstå og konsumere tjenester uten kjennskap til serverimplementering eller tilgang til serverkoden. Ikke alle tjenester kan beskrives med OpenAPI. Spesifikasjonen er ikke ment å dekke alle mulige stiler til REST APIs, spesifikasjonen er uavhengig av programmeringsspråk. Den er også utvidbar til nye teknologier og protokoller utover http.

OpenAPI-spesifikasjonen krever ***ikke***:

1. omskrivning av eksisterende API-er.
2. at det skal brukes som en binding mellom programvare og et API (ala WSDL i SOAP verden)
3. at tjenesten som beskrives eies av den som har utformet beskrivelsen.
4. en spesifikk utviklingsprosess (slik som design-first eller code-first)

Direktoratet for e-helse anbefaler:

Bruk standardiserte metoder for å dokumentere API-er.

For FHIR-baserte REST tjenester må Capability Statement benyttes.

For andre REST tjenester anbefales bruk av enten Capability Statement eller OpenAPI Specification

10 Bruk av feilkoder når tilgang nektes

Både ved bruk av REST og FHIR med REST er det viktig at svar fra en tjener følger standardiserte HTTP status koder. HTTP standarden definerer følgende status kodeserier:

- 1xx-serien er reservert for lavnivå HTTP-tekniske protokollfunksjonalitet.
- 2xx-serien er reservert for vellykkede kall der alt går som planlagt.
- 3xx-serien er reservert for omadressering av trafikk.
- 4xx-serien er reservert for å svare på feil gjort av klienten, f.eks. feil inputdata eller at den ber om ting som ikke eksisterer. Forespørslene skal alltid gi samme svar, og skal ikke endre tjenerens tilstand.
- 5xx-serien er reservert for svar når tjeneren gjør en feil. Ofte blir disse feilene returnert av applikasjonene og normalt utenfor utviklerens rekkevidde. Dette er for å sikre at klientene faktisk mottar et svar. Klienten kan ikke vite tilstanden til serveren når et 5xx-svar mottas, og derfor bør disse unngås.

Når klienter nektes tilgang, må en tjener gi nok informasjon slik at det dekker klientens behov for en god brukervennlighet. Tjeneren må returnere en http status kode som klienten kan presentere for brukeren sin. Nekting av tilgang kan skyldes flere forhold, for eksempel at bruker ikke er autorisert til å benytte API-et, bruker er ikke autorisert til å få tilgang til bestemte data eller at innbygger har sperret tilgang til dataene for brukeren.

Det er viktig når klienter nektes tilgang at svaret fra en tjener blir valgt med omhu. Dersom tjeneren returnerer for mye informasjon, kan dette avsløre sensitiv informasjon. For eksempel kan svaret fra en spørring etter en pasient sin diagnose som brukeren ikke har tilgang til, kunne gi opplysninger om at data om diagnosen faktisk finnes hos tjeneren. Dette kan brukeren sette sammen med annen informasjon som kan resultere i at brukeren kan forstå at pasienten har diagnosen.

FHIR-standarden har noen anbefalinger rundt håndtering av dette som vi ønsker å ta med som generelle anbefalinger for bruk av REST.

Direktoratet for e-helse anbefaler:

Resultatet som gis av tjener når tilgang til data nektes må tilpasses basert på konteksten tilgangen nektes. Eksempler på en slik tilpasning:

1. Dersom det finnes data som bruker ikke skal vite at finnes, så må det ikke skilles mellom "ikke funnet" og "ikke tilgang".
   Ved benytte status kode 404 "not found" når bruker ikke har tilgang, så kan ikke bruker forstå at dataene finnes.
2. Dersom bruker ikke skal vite at pasient har sperret dataene, så må ikke bruker vite når han/hun har fått og ikke fått tilgang.
   Ved å benytte status kode 404 "not found" når pasienten har sperret brukerens tilgang til dataene, så kan ikke bruker forstå at det finnes en sperring.

3. Dersom det er OK at bruker vet at han/hun ikke har tilgang, men ikke skal forstå hvordan han/hun skal kunne få tilgang, så bør kode 403 "forbidden" benyttes fremfor koden 401 "unauthorized" når tilgang nektes.
4. Dersom en tjener tillater å returnere lister med ressurser anbefales det å returnere en tom liste som et vellykket kall når bruker ikke har tilgang til å se listen. Da kan bruker ikke skille mellom der det ikke finnes data og der brukeren ikke har tilgang (og det finnes data).

11 Vedlegg

11.1 Sentrale begreper

Begrep	Beskrivelse
API	Begrepet API betegner i all enkelthet et grensesnitt i en programvare hvor spesifikke deler av denne kan aktiveres (kjøres) fra en annen programvare gjennom kall til grensesnittet. I dette dokumentet er API brukt i en kontekst hvor en virksomhet tilgjengeliggjør et grensesnitt i en programvare for andre aktører via web. Dette er også kalt Web API. API benyttes i dokumentet som både SOAP-baserte API-er og REST-baserte API-er.
Autentisering	Bekrefte en person eller et systems identitet.
Datadeling	Et samhandlingsmønster hvor virksomheter deler eller samarbeider om strukturert informasjon gjennom felles ressurser/tjenester.
Datadelingsgrensesnitt	Et grensesnitt for deling av data som gjøres tilgjengelig for andre aktører gjennom bruk av web. Datadelingsgrensesnitt kan dekke både lesing og registrering/oppdatering av helse- og personopplysninger og dette krever tilgangsstyring på tvers av virksomheter i tillegg til høy tilgjengelighet. Et datadelingsgrensesnitt kan realiseres som et API.
FHIR	FHIR er en ny standard som er under utarbeidelse innen HL7 (release 3 STU publisert mars 2017, release 4 er planlagt i desember 2018), for strukturert utveksling av helsedata mellom systemer og applikasjoner. FHIR er basert på teknologier som er enkle å utvikle web-applikasjoner på, som REST og JSON.
Klient	Det programmet som anvendende aktør benytter og som håndterer forespørsel til API på vegne av den anvendende aktør.
Ressurs	I henhold til [RFC2396] så kan en Ressurs være alle former for elektronisk lagret informasjon så lenge det har en id. I vår kontekst så kan man si at ressurs er en fellesbetegnelse for en

Begrep	Beskrivelse
	avgrenset mengde informasjon med egen identifikator som deles med andre virksomheter/personer, og som er representert med et navn, har en eier og kan kontrolleres gjennom et API. Eierskapet er sterkt knyttet til retten til å bestemme tilgangsreglene til en ressurs.
Ressurstjener	Tjener som har ansvaret for å tilby operasjoner på ressursen samt å beskytte det.
URL	Forkortelse for Unified Resource Locator, men ofte kalt for webadresse.URL er en referanse til en webbasert ressurs.
Web API	API som kan nås via web.
Webservice	En webservice er et API som kalles over web og utfører sammensatte funksjoner bestående av et sett med handlinger. I motsetning til en ressurs, så er ikke en webservice knyttet til en representasjon, men til funksjonelle oppgaver som klienter ønsker å få utført for å oppnå en måltilstand.

11.2 Risikovurdering

Alle virksomheter som tilbyr et datadelingsgrensesnitt må sørge for at nødvendige risikovurderinger er gjennomført. Kravet kommer fra artikkel 35 i personvernforordningen som er en del av personopplysningsloven samt Forskrift om tilgang til helseopplysninger mellom virksomheter §5. Se *Normen*[1] kapittel 3.3: (siste avsnitt)

> *Risikovurderingen skal dokumenteres. Konklusjonene fra vurderingen skal sammenlignes med fastlagt nivå for akseptabel risiko. Er risikoen høyere enn fastsatt nivå for akseptabel risiko skal det iverksettes tiltak (nye/endrede) for å oppnå akseptabel risiko. Dersom tekniske tiltak for å oppnå akseptabel risiko ikke innføres umiddelbart, kan det i en overgangsperiode benyttes administrative tiltak, f.eks. i form av prosedyrer.*

Se også Normens faktaark nr. 5 *Nivå for akseptabel risiko* og faktaark nr. 7 – *Risikovurdering*. I sistnevnte finnes eksempler på skjema for risikovurdering.

Temaer som er gjenstand for risikovurdering er blant annet:

- Integritet
- Konfidensialitet
- Tilgjengelighet
- Tilgangskontroll
- Logging
- Pasientsikkerhet

[1] Norm for informasjonssikkerhet versjon 5.3, 20 juli 2018

- Helseopplysninger på avveie
- Omdømme

Under følger noen eksempler på momenter som kan tas med i en risikovurdering.

Integritet

- Data kan endres i mellomtjener med overlegg
- Data kan endres i mellomtjener ved tilfeldighet
- Data lagres og leses ikke korrekt i tjenesten

Konfidensialitet

- Ukrypterte helseopplysninger på mellomtjener kommer på avveie
- Helseopplysninger kan leses under transport
- Tekniske logger inneholder for mye informasjon (helseopplysninger)

Tilgjengelighet

- Tjenesten er nede
- Innlogging med *single sign-on* (SSO) feiler
- Nettverk/bredbånd er utilgjengelig
- Tilgangskontrollen er for streng

Eksempel på en forenklet risikovurdering

Forhold som er vurdert	Tekniske logger inneholder for mye informasjon (helseopplysninger)
Sannsynlighet	3 Mulig
Konsekvens	3 Alvorlig (pasientsikkerhet, omdømme)
Risikonivå	9 Rødt
Analyse	Store mengder enkeltinformasjon kan danne et bilde av en pasients helsesituasjon.
Forslag tiltak	Tekniske logger på en tjeneste må ligge under en dataansvarlig og tilgangsstyres. Tekniske logger bør slettes så fort som mulig uten at det går ut over tjenestens kvalitet, for eksempel etter 24 timer.

11.3 Evalueringskriterier for valg av anbefalte kommunikasjonsrammeverk

Vi har identifisert 6 grupper av internasjonale kommunikasjonsrammeverk som vi har valgt å evaluere.
Tabellen under beskriver disse.

Alternative grupper	Beskrivelse
HTTP	Benytte HTTP protokollene til å overføre XML eller JSON innhold. Kun GET (be om data) og POST (oppdatere data) benyttes. Bruk av URL-er som inneholder tjenestebaserte kall (Remote Procedure Calls - mønster)
REST/RESTful	Bygger på HTTP. Bruke alle tilgjengelige HTTP verb til å overføre XML/JSON baserte ressurser over http protokoll og med unik identifiserbare ressurser i URLen. Minst nivå 2 på REST maturity modell
FHIR-basert REST	Bygger på REST/RESTful gruppen, men man legger til FHIR-baserte utvidelser knyttet til teknisk samhandlingsevne slik som operations i URL, bruk av bundles (sette sammen ressurser til en melding) og støtte for å håndtere endringer på flere ressurser i en enkel transaksjon. Samt også kapabilitetstjenesten som forteller klienter hvordan serveren støtter ressurser.
SOAP baserte webservice	Bygger på HTTP. Standard webservices ved hjelp av SOAP standarden, samt utvidelse for å håndtere konfidensialitet og integritet (ws-security)
SOAP baserte webservice++	Bygger på SOAP baserte webservice, men utvidet med aktuelle ws*-standarder for økt funksjonalitet.
ebXML/ebMS 3.0 over SOAP	Benytte ebMS 3.0 over SOAP. ebMS 3.0 er mer i samsvar med SOAP og WS-I profilene enn ebMS 2.0 og en av motivene bak versjon 3.0 er også å kunne inkludere andre WS* standarder. ebMS 3.0 er i liten bruk i helse- og omsorgstjenesten, eneste kjente anvendelse er XDS baserte grensesnitt som bygger på ebMS 3.0. Difi har valgt denne standarden til bruk i dialog med Digital sikker meldingsboks samt meldingsutveksling mellom offentlige etater. EU benytter ebMS 3.0 til meldingsutveksling mellom EU land (eDelivery).

For å evaluere ulike grupper av internasjonale kommunikasjonsrammeverk ble det benyttet kriterier som også er benyttet i forbindelse med evaluering av internasjonale standarder.

Evalueringen av rammeverkene er foretatt med farger for God/Nøytral/Dårlig:

God	Nøytral	Dårlig

Tabellen under vise selve evalueringen.

Teknologi/kapabilitet	**Kommentar**	**REST**	**FHIR-basert REST**	**SOAP basert WS**	**SOAP basert WS++**	**HTTP**	**ebMS over WS**
Interne faktorer							
Behovsoppnåelse	I hvilken grad møter standarden kjente og relevante behov i helse- og omsorgstjenesten? Kan inkludere både nåtid og fremtid, i den grad dette er kjent.						
Innovasjon/nytenkning	I hvilken grad er standarden innovativ i måten å løse utfordringer på?						
Modenhet	I hvilken grad er standarden gjennomprøvd og stabil? Kan man forvente ofte og/eller store endringer i standarden i nærmeste fremtid?						
Kompleksitet	Innebærer standarden en økning eller reduksjon i kompleksitet sammenlignet med andre relevante standarder?						
Fleksibilitet	I hvilken grad kan standarden benyttes til å løse ulike problemer og i hvilken grad kan den tilpasses konkrete problemer? For						

	eksempel ved å begrense eller utvide innhold og funksjonalitet.						
Implementasjonsvennlighet	I hvilken grad antas det at standarden er lett å realisere i løsninger? Krever det stor innsats/ressurser og/eller spesifikk kompetanse for å realisere den?						
Sammenheng/avhengighet med andre standarder	I hvilken grad er standarden avhengig av andre standarder? Forutsetter den samtidig bruk av andre standarder?						
Eksterne faktorer							
Teknologiutvikling	I hvilken grad er standarden i tråd med den generelle teknologiutviklingen innen IKT (inkludert øvrige sektorer)?						
Implementasjoner hos leverandører	I hvilken grad finnes det tilgjengelige kommersielle løsninger basert på standarden?						
Relevant kompetanse i sektoren	I hvilken grad finnes det kompetanse i sektoren på standarden, for eksempel aktører som jobber med systemintegrasjon eller utarbeider føringer for bruk av standarder?						
Erfaringer	I hvilken grad er det kjente erfaringer med bruk av standarden,						

	og hva sier disse erfaringene?						
Støtte/interesse i marked	I hvilken grad vurderes det at standarden har støtte og interesse i helsesektoren og blant nasjonale og internasjonale markedsaktører?						
Politiske føringer, strategier og planer i sektoren	Er standarden i tråd med eller i konflikt med gjeldende politiske føringer, strategier og planer for bruk av standarder og/eller løsninger i sektoren?						

11.4 Referanser

[1] https://spring.io/understanding/REST

[2] REST maturity model: https://martinfowler.com/articles/richardsonMaturityModel.html

[3] Operations: http://hl7.org/implement/standards/fhir/operations.html

[4] http://hl7.org/implement/standards/fhir/http.html#transaction

[5] http://hl7.org/implement/standards/fhir/http.html#capabilities

[6] Faktaark 24: Kommunikasjon over åpne nett: https://ehelse.no/personvern-og-informasjonssikkerhet/norm-for-informasjonssikkerhet/normen/faktaark-24-kommunikasjon-over-apne-nett

[7] Veiledning for systemteknisk sikkerhet, Nasjonal sikkerhetsmyndighet: https://www.nsm.stat.no/publikasjoner/regelverk/veiledninger/veiledning-for-systemteknisk-sikkerhet/sikring-av-kommunikasjon-med-tls/

[8] Sikring av Windows TLS, Nasjonal sikkerhetsmyndighet: https://www.nsm.stat.no/globalassets/dokumenter/veiledninger/systemteknisk-sikkerhet/u-03_sikring_av_windows_tls.pdf

[9] Veileder om anonymisering av personopplysninger, Datatilsynet: https://www.datatilsynet.no/globalassets/global/regelverk-skjema/veiledere/anonymisering-veileder-041115.pdf

[10] http://hl7.org/implement/standards/fhir/bundle.html

[11] Extended Operations on the RESTful API: https://www.hl7.org/fhir/operations.html

[12] Capability Statement: https://www.hl7.org/fhir/capabilitystatement.html

[13] OpenAPI Spec: https://github.com/OAI/OpenAPI-Specification

[14] search in fhir - http://hl7.org/fhir/2018Sep/http.html#search

Direktoratet for e-helse

Besøksadresse
Verkstedveien 1
0277 Oslo

Postadresse
Postboks 6737
St. Olavs plass
0130 OSLO