Implementasjonsguide SMART App Launch Framework

E-helse Arkitektur

Exported on Mar 05, 2019

Table of Contents

1 Introduksjon

SMART sørger for en pålitelig og sikker autorisasjonsprotokoll for en rekke applikasjonsarkitekturer og kan benyttes for å støtte oppunder applikasjoner som benyttes av klinikere og/eller pasienter.

SMART gir tredjepartsapplikasjoner autorisert tilgang til data i elektroniske pasientjournaler via en pålitelig og sikker autorisasjonsprotokoll. Applikasjonene kan starte som en del av eller utenfor brukergrensesnittet til et EPJ-system. Rammeverket støtter per i dag fire bruksscenarioer beskrevet i Argonaut prosjektets fase 1. Argonaut prosjektet tar for seg følgende bruksscenarioer:

1. Applikasjoner for pasienter som kan starte frittstående
2. Applikasjoner for pasienter som kan starte fra en portal
3. Applikasjoner for klinikere som kan starte frittstående
4. Applikasjoner for klinikere som kan starte i en EPJ eller portal

Dette dokumentet vil kun gå i detalj på bruksscenario 4.

SMART benyttet sammen med FHIR beskrives som SMART on FHIR og beskriver to deler, en autorisasjonsprotokoll basert på OAuth der en web-applikasjon registrert som klient i en EPJ gis autorisert tilgang til lokale helsedata. I tillegg vises det til spesifikasjonen og den voksende standarden FHIR som beskriver ett sett med ressurser og interaksjoner for å utveksle data.

2 Formål med dokumentet, målgruppe og avgrensning

2.1 Formål

Formålet med dokumentet er å tydeliggjøre premisser for teknisk tilrettelegging av EPJ for SMART App Launch Framework.

2.2 Målgruppe

Målgruppen for dokumentet er tekniske ressurser, testpersonell og prosjektledelse med ansvar for å tilrettelegge EPJ for SMART App Launch Framework.

2.3 Avgrensning

Dokumentet er avgrenset til de tjenestene og behovene som skal understøtte en implementasjon av SMART App Launch Framework for bruksscenario 4 beskrevet i introduksjonen av dokumentet.

Dokumentet omfatter ikke

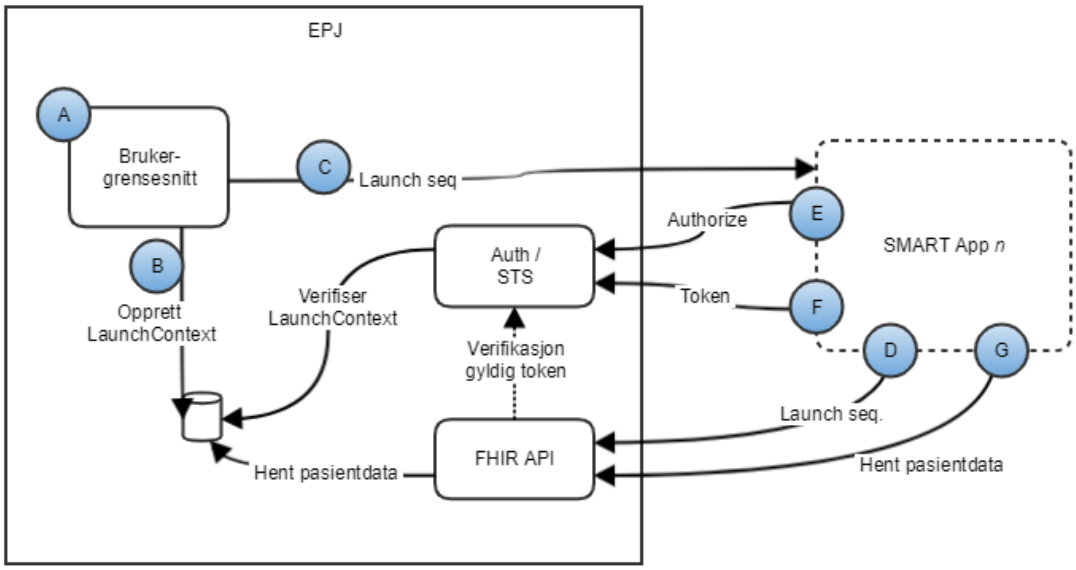
- etablering av driftsrutiner
- etablering av overvåkningsrutiner

2.4 Definisjoner

- Helsepersonell – en person autorisert som helsepersonell i henhold til helsepersonelloven § 48. Rettighetene til ulike autorisasjoner er forskjellig, f.eks. kan kun leger verifisere kritisk informasjon.
- Innbygger – en person som bor innen et spesifikt område. Et område her kan for eksempel være et land, fylke, kommune, by og sted.
- Pasient – en person som får behandling av helsepersonell for psykisk og/eller fysisk sykdom/lidelse eller skade, på sykehus/klinikk og/eller andre behandlingsinstitusjoner.
- EPJ – Elektronisk pasientjournal
- EHR – Electronic Health Record
- SMART – Substitutable Medical Applications
- FHIR – Fast Healthcare Interoperability Resources
- SMART App Launch Framework – Standardbasert applikasjonplattform for elektroniske pasientjournaler
- Tilgangstoken – Akkreditiver benyttet for å aksessere beskyttede ressurser, se seksjon 1.4 i RFC6749
- Autorisasjonskode – En kode som tilegnes i forbindelse med autorization code grant flyten, denne benyttes til å tilegne seg et tilgangstoken, se seksjon 1.3 og 1.3.1 i RFC6749
- Refresh token – Akkreditiver benyttet for å tilegne seg et nytt tilgangstoken når det gjeldende tilgangstokenet utgår på tid eller blir ugyldig, seksjon 1.5 i RFC6749

3 Løsningsbeskrivelse

3.1 SMART App Launch Framework



3.1.1 A. Integrert nettleser for å starte webapplikasjon

EPJ benytter integrert nettleser for å starte en webapplikasjon med SMART on FHIR-støtte

3.1.2 B. LaunchContext opprettes

EPJ oppretter en LaunchContext som tildeles en unik identifikator og assosieres med client_id for SMART-applikasjonen. Konteksten består av:

Parametere		
patient	PÅKREVET	Den logiske logiske ressursidentifikatoren for pasienten
practitioner	VALGFRI	Den logiske ressursidentifikatoren for helsepersonellet som benytter applikasjonen
encounter	VALGFRI	Den logiske ressursidentifikatoren for konsultasjonen

3.1.3 C. Launch-sekvensen initieres

I praksis er dette en URL til webapplikasjonen.
Eksempel: `Location: https://app/launch?iss=https%3A%2F%2Fehr%2Ffhir&launch=xyz1`

Parametere		
iss	PÅKREVET	Identifiserer EPJens FHIR endepunkt. Web-applikasjonen benytter dette endepunktet for å skaffe ytterligere detaljer EPJen, inkludert URLen til autorisasjonserveren
launch	PÅKREVET	Ikke-transparent identifikator for denne spesifikke oppstartsekvensen. Dette parameteret skal kommuniseres tilbake til EPJen på

Parametere		
		autorisasjonstidspunktet og legges ved som et launch-parameter (se eksempel ovenfor).

3.1.4 D. Applikasjonen mottar launch-notifikasjon

Ved mottak av launch-notifikasjonen skal applikasjonen forespørre utstederen (iss) sitt /metadata/ endepunkt eller .well-known/smart-configuration.json endepunkt som inneholder URLene til EPJen sitt authorize og token endepunkt.

3.1.5 E. Applikasjonen utfører en forespørsel mot authorize endepunktet

Applikasjonen skal deretter gjøre en forespørsel mot authorize endepunktet med følgende parametere:

Parametere		
response_type	PÅKREVET	Fiksert verdi code.
client_id	PÅKREVET	Klientens identifikator.
redirect_uri	PÅKREVET	Må samsvare med en av de forhåndsregistrerte redirect URLene for klienten.
launch	PÅKREVET	Må samsvare med den mottatte launch-parameteren fra EPJ.
scope	PÅKREVET	Applikasjonen angir ved hjelp av parameteren scope hvilken aksess den trenger til helsedata. Dette inkluderer kliniske scope som patient/*.read, openid, fhirUser, samt et launch scope som indikerer at applikasjonen ønsker den allerede etablerte launch-konteksten fra EPJen.
state	PÅKREVET	Ikke-transparent verdi satt av klienten for å opprettholde tilstanden mellom forespørsel og responsen. Autorisasjonsserveren inkluderer verdien når den dirigerer nettleseren tilbake til klienten. Parameteren skal benyttes for å forhindre cross-site request forgery og session fixation attacks.
aud	PÅKREVET	URL til EPJen sin ressursserver (FHIR-endepunkt). Denne parameteren skal motvirke lekkasje av genuine bearer tokens til en falsk ressursserver. I kontekst av gjeldende bruksscenario skal verdien i aud parameteren være den samme som var satt i iss parameteren.

Applikasjonen skal bruke en uforutsigbar verdi for state parameteren med minimum 122 bit entropi, en korrekt konfigurert tilfeldig uuid vil dermed være egnet. Applikasjonen skal deretter validere verdien av state parameteren ved mottak til redirect URL endepunktet og skal sikre at verdien er bundet opp mot brukerens gjeldende sesjon, for eksempel ved å relatere verdien til en sesjonsidentifikator utstedt av applikasjonen. Applikasjonen burde begrense tilgangsnivå og tidsperiode til det absolutte minimum.

Dersom applikasjonen trenger å autentisere identiteten til brukeren skal den inkludere to OpenID Connect scope: openid og fhirUser. Når applikasjonen forespør disse to scopene, og forespørselen godkjennes vil applikasjonen motta et id_token i tillegg til et tilgangstoken. Se seksjonen SMART launch kontekst parametere.

Som et eksempel på hvilke scope en applikasjon vil kunne be om la oss anta at en applikasjon trenger demografi og observasjoner for en pasient, i tillegg til informasjon om innlogget bruker kan applikasjonen be om:

- patient/Patient.read
- patient/Observation.read
- openid fhirUser

For bruksscenario 4 som denne implementasjonsguiden fokuserer på skal applikasjonen alltid angi scopet launch og URL-parameteren launch={launch_id}, launch_id er den samme verdien som applikasjonen mottok i steg C og denne vil bli assosiert med EPJens kontekst for denne launch-notifikasjonen.

En typisk URL til authorize endepunktet vil se slik ut:

```
Location: https://ehr/authorize?
  response_type=code&
  client_id=app-client-id&
  redirect_uri=https%3A%2F%2Fapp%2Fafter-auth&
  launch=xyz123&

scope=launch+patient%2FObservation.read+patient%2FPatient.read+openid+fhi
rUser&
  state=98wrghuwuogerg97&
  aud=https://ehr/fhir
```

Autorisasjonen avgjøres av EPJens autorisasjonsserver som potensielt kan kreve at sluttbrukeren samtykker til autorisasjonen. Autorisasjonsserveren håndhever tilgangsregler basert på lokale tilgangsregler og ev. input fra sluttbruker. EPJen avgjør til slutt om applikasjonen skal tillates eller nektes tilgang. Avgjørelsen kommuniseres tilbake til applikasjonen ved at EPJens autorisasjonsserver godkjenner forespørselen og returnerer en autorisasjonskode, eller nekter tilgang og returner en feilkode/-respons, se seksjon 4.1.2.1 i RFC6749. Autorisasjonskoder har kort levetid, vanligvis utløper de i løpet av ett minutt. Koden sendes til applikasjonen når EPJens autorisasjonsserver navigerer til applikasjonens redirect_uri med følgende parametere:

Parametere		
code	**PÅKREVET**	Autorisasjonskoden generert av autorisasjonsserveren. Autorisasjonskoden skal utløpe kort tid etter at den er utstedt for å begrense risikoen for lekkasjer
state	**PÅKREVET**	Eksakt verdi slik den var mottatt fra klienten

Basert på client_id, gjeldende EPJ-bruker, konfigurerte tilgangsregler, og ev. direkte input fra brukeren vil EPJen godkjenne eller avvise forespørselen. Denne avgjørelsen kommuniseres tilbake til applikasjonen ved å navigere nettleseren til applikasjonens registrerte redirect_uri:

```
Location: https://app/after-auth?
  code=123abc&
  state=98wrghuwuogerg97
```

3.1.6 F. Applikasjonen veksler inn autorisasjonskoden i et tilgangstoken

Etter at applikasjonen har mottatt en autorisasjonskode veksles denne inn i et tilgangstoken via et HTTP POST kall til EPJens autorisasjonsserver token-endepunkt, nødvendige parametere legges ved som innhold i forespørslen og content-type settes til application/x-www-form-urlencoded som beskrevet i seksjon 4.1.3 av RFC6749.

For såkalte public apps er autentisering ikke mulig siden klienten mangler en secret og kan derfor ikke bevise sin identitet når den utfører et kall. Ende-til-ende systemet kan fortsatt være sikkert siden klienten aksesseres fra et kjent HTTPS beskyttet endepunkt håndhevet av redirect_uri parameteret i konfigurasjonen. For såkalte confidential apps er det påkrevet å sette en Authorization header som benytter HTTP Basic authentication, brukernavnet vil være applikasjonens client_id og passordet er applikasjonens client_secret, se eksempel.

Parametere		
grant_type	PÅKREVET	Fiksert verdi: authorization_code
code	PÅKREVET	Autorisasjonskoden applikasjonen mottok fra autorisasjonsserveren sitt authorization-endepunkt
redirect_uri	PÅKREVET	Den samme redirect_uri som ble benyttet i den initiale autorisasjonsforespørslen
client_id	BETINGET	Påkrevet for public apps, utelates for confidential apps

EPJens autorisasjonsserver skal returnere et JSON-objekt som inkluderer et tilgangstoken eller en beskjed som indikerer at tilgangsforespørslen ikke ble godkjent. JSON-strukturen inkluderer følgende parametere:

Parametere		
access_token	PÅKREVET	Tilgangstoken utstedt av autorisasjonsserveren
token_type	PÅKREVET	Fiksert verdi: bearer
expires_in	ANBEFALES	Tilgangstokenet sin levetid i sekunder, etter dette vil tilgangstokenet ikke bli akseptert av ressursserveren
scope	PÅKREVET	Aksesstilgang som er autorisert, legg merke til at dette kan være forskjellig fra det som applikasjonen initialt ba om
id_token	VALGFRI	Dersom forespurt inneholder informasjon om pålogget bruker
refresh_token	VALGFRI	Et token som kan benyttes til å forespørre autorisasjonsserveren om et nytt tilgangstoken med det samme nivået eller subsett av den opprinnelige autorisasjonstildelingen

Dersom applikasjonen startes i en pasientkontekst skal parametere for å kommunisere kontekstverdiene inkluderes. For eksempel en parameter som "patient": "123" indikerer FHIR ressursen https://ehr/fhir/Patient/123. Andre kontekstparametere kan også inkluderes, for utfyllende detaljer se SMART launch context parameters.

Parameterne for kontekstverdiene inkluderes som en del av innholdet i HTTP-responsen fra token-endepunktet som beskrevet i seksjon 5.1 av RFC6749.

Autorisasjonsserveren response skal inkludere en HTTP Cache-Control respons header med verdien no-store, i tillegg til en Pragma respons header med verdien no-cache.

Eksempel på respons fra token-endepunktet:

```
{
  "access_token": "i8hweunweunweofiwweoijewiwe",
```

```
  "token_type": "bearer",
  "expires_in": 3600,
  "scope": "patient/Observation.read patient/Patient.read",
  "intent": "client-ui-name",
  "patient":  "123",
  "encounter": "456"
}
```

3.1.7 G. Applikasjonen har nå tilgang til helsedata

Applikasjonen har nå tilgang til helsedata via det mottatte tilgangstoken.

Med et gyldig tilgangstoken har applikasjonen tilgang til beskyttede helsedata i EPJ ved å gjøre forespørsler mot EPJen sitt tilhørende FHIR API. Alle forespørsler til FHIR APIet må inkludere en Authorization header som representerer access_token som et "Bearer" token:

```
Authorization: Bearer {{access_token}}
```

Med hjelp fra kontekstparametrene fra responsen F. vet applikasjonen hvilken pasient som er en del av konteksten og har et OAuth Bearer tilgangstoken som kan brukes til å hente ut helsedata:

Eksepempel på forespørsel:

```
GET https://ehr/fhir/Patient/123
Authorization: Bearer i8hweunweunweofiwweoijewiwe
```

Eksempel på respons:

```
{
  "resourceType": "Patient",
  "birthTime": ...
}
```

Eksempel med komplett payload

4 Scopes og Launch Context

SMART App Launch Framework benytter OAuth scopes for å kommunisere og forhandle krav til tilgang. I tillegg til hvilke scope som er satt i tilgangstokenet er tilgang i tillegg begrenset til de privilegiene eller autorisasjonen brukeren har tilgang til. Generelt benyttes scopes for tre typer data:

1. Kliniske data
2. Kontekstuelle data
3. Identitetsdata

Enkel oversikt over de mest brukte scopene:

Scope	Tilgang	Scope Type
patient/*.read	Tilgang til å lese en hvilken som helst ressurs for den gjeldende pasienten	Kliniske data
user/*.*	Tilgang til å lese og skrive til alle ressurser den gjeldende brukeren har tilgang til	Kliniske data
openidfhirUser (ev. openidprofile)	Tilgang til å motta informasjon om den gjeldende innloggede brukeren	Identitetsdata
launch	Tilgang til å motta en launch context når applikasjonen er startet fra en EPJ	Kontekstuelle data
launch/patient	Når applikasjonen starter utenfor konteksten av en EPJ, be om å velge en pasient ved oppstart	Kontekstuelle data
offline_access	Be om en refresh_token som kan benyttes til å motta et nytt tilgangstoken, også etter at endebruker ikke lenger er innlogget	-
online_access	Be om en refresh_token som kan benyttes til å motta et nytt tilgangstoken, og som vil være brukbar så lenge endebruker er pålogget	-

4.1 Scopes for å kunne forespørre kliniske data

SMART App Launch Framework-spesifikasjonen definerer OAuth scopes som korresponderer direkte med FHIR ressurstyper, samt lese (read) og skrive (write) tilgang for pasientspesifikke eller brukernivå aksess. Applikasjoner som trenger leseaksess til data i EPJ (lese- og søkeinteraksjoner) skal be om lese (read) scope. Applikasjoner som trenger å kunne skrive data til en EPJ, FHIR-interaksjonen create, update, delete, skal be om skrive (write) scope.

4.1.1 Klinisk scope-syntaks

```
clinical-scope ::= ( 'patient' | 'user' ) '/' ( fhir-resource | '*' ) '.'
( 'read' | 'write' | '*' )`
```

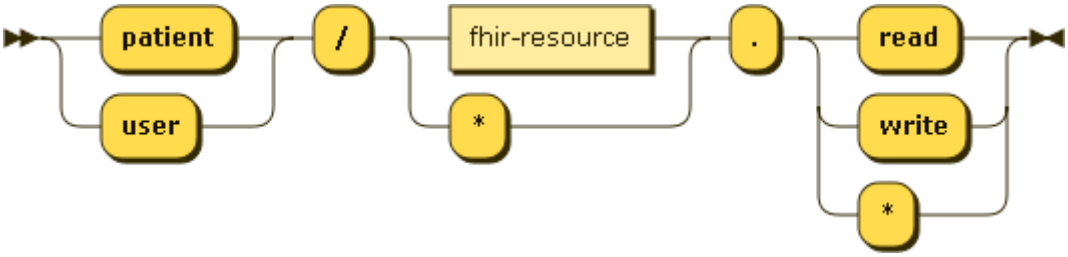


Figure 1 Clinical scope syntax diagram

4.1.2 Pasientspesifikke scope

Pasientspesifikke scope gir tilgang til data om en *enkelt* pasient. *Hvilken* pasient dette måtte være er ikke spesifisert her, kliniske scope omhandler *hva* ikke *hvem*. Pasientspesifikke scope innehar formen: patient/:resourceType.(read|write|*).

Eksempler på pasientspesifikke scope:

Mål	Scope	Kommentar
Lese alle observasjoner tilhørende en pasient	patient/Observation.read	
Lese alle demografiske data om en pasient	patient/Patient.read	Noter forskjellen i kapitalisering mellom "patient", tilgangstype, og "Patient", ressurs
Lagre en ny blodtrykksmåling for en pasient	patient/Observation.write	Noter at med dette scopet kan en applikasjon ikke bare lagre blodtrykk, men også andre typer observasjoner. NB: skrivetilgang (write) impliserer ikke lesetilgang (read)
Les alle tilgjengelige data for en pasient	patient/*.read	

4.1.3 Brukernivå scope

Brukernivå scope gir tilgang til data som brukeren kan aksessere. Dette omhandler ikke kun data *om* brukeren, men data som er *tilgjengelig* for brukeren. Brukernivå scope innehar formen: user/:resourceType(read|write|*).

Eksempler på brukernivå scope:

Mål	Scope	Kommentar
Lese en oversikt over lab-observasjoner på tvers av pasienter	user/Observation.read	
Håndtere alle avtaler/timeavtaler som den autoriserte brukeren har tilgang til	user/Appointment.read user/Appointment.write	Legg merke til at både read og write må begge være angitt (skrivetilgang impliserer *ikke* lesetilgang)
Håndtere alle ressurser som den autoriserte brukeren har tilgang til	user/*.read user/*write	
Kunne velge en pasient	user/Patient.read	Tillater applikasjonen å velge en pasient

4.2 Scopes for å kunne forespørre om kontekstuelle data

Mange applikasjoner baserer seg på kontekstuelle data fra EPJen som:

- Hvilken pasientjournal er "åpen" i EPJ i denne kontekst?
- Hvilken konsultasjon/møte er "åpen" i EPJ i denne kontekst?
- På hvilket behandlingssted befinner gjeldende helsepersonell seg i denne kontekst?

For å kunne svare ut slike detaljer kan en applikasjon be om launch-kontekst scope i tillegg til kliniske scope. I det bruksscenarioet som denne implementasjonsguiden beskriver er det kun scopet launch som kan benyttes. For mer informasjon 3.1 Apps that launch from the EHR.

4.3 Scopes for å kunne forespørre om identitetsdata

Applikasjoner som trenger å autentisere helsepersonellet kan benytte OpenID Connect scopene: openid og fhirUser. Når forespørslen om disse scopene innvilges mottar applikasjonen en id_token som en del av token-responsen. For mer informasjon 4. Scopes for requesting identity data.

5 Sikkerhet

5.1 Generelle sikkerhetsvurderinger

De som implementerer SMART App Launch Framework må være oppmerksom på krav til sikkerhet og gjennomføre en vurdering av risiko og sikkerhet.

Ved tilkobling til helsenettet binder man seg til Normens krav til informasjonssikkerhet. Selv om det er ikke er et krav om at SMART App Launch Framework tilbys via helsenettet, BURDE en tilbyder av SMART App Launch Framework vurdere sikkerhetskravene som er definert i Normen.

I tillegg til dette har Direktoratet for e-helse utgitt referansearkitekturer for datadeling og dokumentdeling. Under er det kort beskrevet noen områder og krav, men hver implementasjon må vurdere sikkerheten ved sin egen implementasjon.

5.2 Juridisk / Avtale

Ved etablering av SMART App Launch Framework SKAL det vurderes hvilke juridiske og avtalemessige krav som gjelder basert på informasjonen som tilbys. Dette kan inkludere databehandleravtaler eller andre juridiske krav for etablering av SMART App Launch Framework.

5.3 Krav til risikovurderinger

Ved etablering av SMART App Launch Framework SKAL det gjennomføres en risikovurdering, og denne skal dokumenteres for senere revisjon ved behov

5.4 Sikring av kommunikasjonen

Det er viktig at generell HTTP sikkerhet implementeres og det SKAL benyttes TLS eller annet transportsikring ved overføring av informasjon. Direktoratet for e-helse har utgitt en referansearkitektur for datadeling som burde vurderes.

5.5 Autentisering

Klienten og bruker som gis tilgang SKAL autentiseres.

5.6 Logging / Audit

Det SKAL logges nok informasjon slik at det i etterkant er mulig å verifisere tilgangsbeslutningen.

5.7 Refresh token

Dersom applikasjonen mottar et refresh token sammen med et tilgangstoken kan den bruke dette refresh token for å hente ut et nytt tilgangstoken med samme tilgangsnivå når tilgangstokenet utløper. Et refresh token SKAL være bundet mot den samme client_id og SKAL ha det samme eller et subsett av tilgangsnivåene som tilgangstokenet den er assosiert med.

5.8 Lagring av tilgangstoken klientside

Applikasjoner BURDE lagre tokens den mottar i applikasjonspesifikke lagringslokasjoner, ikke systemtilgjengelige lokasjoner.