Direktoratet for e-helse

HISD 1177:2017

Implementasjonsguide: API for bruk av kommunikasjonsparametere

Versjon 1.0

Publikasjonens tittel:
HISD 1177:2017 Implementasjonsguide: API for bruk av kommunikasjonsparametere
Versjon 1.0

Utgitt:
02/2017

Teknisk rapport nr.:
HISD 1177:2017

Utgitt av:
Direktoratet for e-helse

Kontakt:
postmottak@ehelse.no

Postadresse:
Postboks 6737 St. Olavs plass, 0130 OSLO

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

Publikasjonen kan lastes ned fra:
www.ehelse.no

Innholdsfortegnelse

1. INNLEDNING

1.1 Formål

Dette dokumentet er utarbeidet av Direktoratet for e-helse og Norsk Helsenett. Dokumentet beskriver hvordan kommunikasjonsparametere og samhandlingsavtaler skal benyttes i praksis for at elektronisk meldingsutveksling i helsesektoren skal bli mer robust. Dokumentet er først og fremst ment for utviklere hos leverandører av IT-systemer for elektronisk meldingsutveksling i helsesektoren.

2. REFERANSER

Følgende dokumenter er nyttige for anvendelsen av denne veiledningen:

[1] Norsk Helsenett, «Adresseregisteret - API-dokumentasjon»
[2] Direktoratet for e-helse, «[HISD 1168:2016] Veiledning til riktig bruk av applikasjonskvittering», 2016
[3] Helsedirektoratet, «[HIS 80415:2012] Applikasjonskvittering: Informasjonsmodell, XML meldingsbeskrivelse og retningslinjer for bruk», 2012
[4] Arbeidsgruppe oppnevnt av Helsedirektoratet, «Arkitekturdokument for bruk av CPP og CPA i norsk helse- og omsorgssektor», 2012
[5] Helsedirektoratet og NAV, «Guide til CPP og CPA», 2013
[6] Helsedirektoratet, «Profil for CPP/CPA – partnerprofiler og avtaler», 2014
[7] Direktoratet for e-helse, «Tjenestebasert adressering del 1: Generelle krav (HIS 1153-1:2016)», 2016

3.TERMER OG DEFINISJONER

Gjennom dokumentet er det benyttet en del nøkkelbegreper som skal forstås som følger:

Kommunikasjonspart	Logisk avgrenset del av en virksomhet i helse- og omsorgstjenesten, som sender og/eller mottar elektroniske meldinger. Merk 1: En kommunikasjonspart har alltid en HER-id. Merk 2: En kommunikasjonspart vil som hovedregel være knyttet til en tjenestetype.
Kommunikasjonsparametere	En samling attributter knyttet til en kommunikasjonspart som forteller potensielle samhandlere hvilke muligheter kommunikasjonsparten har for elektronisk utveksling av en bestemt type og versjon av en melding beskrevet i en bestemt standard
Meldingstype	Identifikasjon av en strukturert beskrivelse av en melding som benyttes for å utveksle en bestemt type informasjon for et bestemt formål, vanligvis beskrevet i form av en meldingsstandard Merk: Sammen med meldingsversjonen angir meldingstypen hvilken informasjon som skal inngå i meldingen og i hvilken struktur denne informasjonen skal foreligge (felter, dataformat osv).
Meldingsversjon	Attributt som skiller ulike versjoner av samme meldingstype fra hverandre. Merk: Alle meldinger som utveksles er av en *type* og en *versjon*, som bestemmer hvordan meldingen skal være. Sammen med meldingstypen angir meldingsversjonen hvilken informasjon som skal inngå i meldingen og i hvilken struktur denne informasjonen skal foreligge (felter, dataformat osv).
Meldingstjener (MSH)	Programvarekomponent som håndterer meldingsformidling i tråd med ebMS spesifikasjonen. Merk: En meldingstjener kan være sammensatt av flere programvarekomponenter i et større system for meldingsformidling og kommunikasjon med eksterne og interne kommunikasjonsparter.
Prosess	En flyt av meldinger som hører sammen og som angir av hvilken meldingstype/-versjon disse skal ha. Eksempel: En melding av type «Henvisning» versjon 1.1 skal etterfølges av «Applikasjonskvittering» versjon 1.1. Merk: Alle prosesser har en ID og et versjonsnummer.
Rolle	Funksjonen kommunikasjonsparten har i prosessen. Merk: Til hver prosess er det knyttet roller. For hver rolle er det angitt hvilke meldingstyper/-versjoner som skal kunne sendes og mottas. Rollene innehas av kommunikasjonsparter.
Samhandlingsprofil (CPP – Collaboration Protocol Profile)	Standardiserte kommunikasjonsparametere som beskriver hvilke muligheter en kommunikasjonspart har implementert i sin systemløsning for å kunne utføre elektronisk samhandling med andre kommunikasjonsparter.
Samhandlingsavtale (CPA – Collaboration Protocol Agreement)	Beskrivelse av hvilke roller to kommunikasjonsparter besetter og dermed hvilke meldingstyper- og versjoner de to kan utveksle seg imellom. Merk: En CPA representerer snittet av de to kommunikasjonspartenes

	CPP-er, dvs. det som er felles, og kan etableres på grunnlag av kommunikasjonspartenes publiserte CPP-er.
Meldingsstandard	Standard som beskriver informasjonen som utveksles mellom aktører. Merk: En melding kan beskrives ved hjelp av en syntaksuavhengig meldingsbeskrivelse (informasjonsmodell) og/eller en syntaksspesifikk meldingsbeskrivelse. Eksempel: Epikrise, Sykmelding, Standard for elektronisk kommunikasjon med pleie- og omsorgstjenesten m.fl.
Adresseregisteret	Felles nasjonalt register med opplysninger som skal benyttes ved kommunikasjon mellom kommunikasjonsparter. Merk: Adresseregisteret driftes av Norsk helsenett og forvaltes av Direktoratet for e-helse.
Kapabilitet	Hvilke meldingstyper og versjoner en kommunikasjonspart støtter
Sertifikat	En kopling mellom signaturverifikasjonsdata og undertegner som bekrefter undertegners identitet og er signert av sertifikatutsteder
Signaturverifikasjonssystem	Programvare eller maskinvare som benyttes for å verifisere elektronisk signatur ved hjelp av signaturverifikasjonsdata
Sertifikatutsteder	En fysisk eller juridisk person som utsteder sertifikater eller tilbyr andre tjenester relatert til elektronisk signatur
Feilmelding (ebXML Error Signal)	ebXML-konvolutt som benyttes som en ebXML Error Signal. Merk 1: Sendes som en signalmelding på ebXML-nivå (ikke transportprotokollnivået). Merk 2: En feilmelding varsler i hovedsak avsender av ebXML-meldingen at denne er avvist av mottakers meldingstjener.
Transportkvittering (ebXML Acknowledgment)	ebXML-konvolutt som benyttes som en ebXML Error Signal. Merk: Sendes som en signalmelding på ebXML-nivå (ikke transportprotokollnivået). Merk 2: En transportkvittering forteller avsender av ebXML-meldingen at mottakers meldingstjener har tatt i mot forsendelsen, og at fagmeldingen kan leveres til mottakers fagsystem.

4. OVERORDNET OM ELEKTRONISK SAMHANDLING

4.1 Situasjonsbeskrivelse

Helsesektoren kjennetegnes ved at det er et stort antall selvstendige aktører (legekontor, sykehus, kommuner, offentlige instanser m.fl.) med tilhørende kommunikasjonsparter som sender elektroniske meldinger seg imellom. Det er mange ulike systemer i bruk og de ulike systemene og installasjonene støtter ikke alltid de samme meldingstypene og -versjonene. Før to kommunikasjonsparter kan starte elektronisk meldingsutveksling må de avtale hvilke meldingstyper og -versjoner, sertifikater, kommunikasjonskanaler og hvilken adressering som skal benyttes når de skal utveksle meldinger seg i mellom. Dette gjøres typisk gjennom tidkrevende manuelle prosesser og med høy risiko for feil. Når noe endres (f.eks. at den ene aktøren oppdaterer sitt sertifikat eller installerer et nytt system som støtter en ny meldingsversjon) må prosessen gjentas.

4.2 Målbilde

Ved å benytte et sentralt register for kommunikasjonsparametere, hvor hver enkelt kommunikasjonspart er ansvarlig for å holde sine egne kommunikasjonsparametere oppdatert, vil utveksling av kommunikasjonsparametere kunne gjøres langt mer effektivt enn i dag. I stedet for å utveksle kommunikasjonsparametere manuelt i forkant av selve informasjonsutvekslingen med nye/endrede kommunikasjonsparter, vil avsender henvende seg til det sentrale registeret for å hente informasjon om mottakers kapabiliteter. Dette gjør at endringer i samhandlingsmønstre (hvem kommuniserer med hvem), kapabiliteter (hvilke meldingstyper og -versjoner som støttes) og attributter (f.eks. bytte av sertifikat) vil kunne gjennomføres raskere, med mindre manuelt arbeid og med færre feilkilder. Myndighetene vil i tillegg få bedre oversikt over hvilke meldingstyper og -versjoner (synkrone så vel som asynkrone) som er i bruk og vil dermed kunne planlegge inn- og utfasing av disse på en bedre måte.

4.3 Utveksling av kommunikasjonsparametere via Adresseregisteret

Adresseregisteret er etablert som felles nasjonalt register for presis adressering av helseopplysninger som sendes elektronisk eller per post innen helse- og omsorgstjenesten. Adresseregisteret har støtte for utveksling av kommunikasjonsparametere via et Web Services-basert API. Støtten er basert på den norske profilen av den internasjonale standarden CPP/CPA som er en del av ebXML-rammeverket til Oasis. API-et tilbyr både metoder for full tilgang til den underliggende CPP/CPA-strukturen og et forenklet sett for tilgang til essensen av kommunikasjonsparameterne. I de aller fleste tilfeller vil bruk av det forenklede settet være tilstrekkelig. Bruk av metodene i det forenklede settet er beskrevet i de etterfølgende kapitlene.

Det er en målsetning at oppsett og bruk av kommunikasjonsparametere langt på vei skal være automatiserte prosesser som ikke berører sluttbruker direkte. Følgende overordnede designprinsipper legges til grunn når det utarbeides støtte for kommunikasjonsparametere:

- Sluttbruker eksponeres ikke for annen kompleksitet og terminologi enn det som er en naturlig del av brukerens fag-/arbeidsområde. Teknisk terminologi (som «CPP», «CPA») forbeholdes teknisk personell som systemadministratorer, driftsoperatører o.l.
- Det er sluttbrukers faglige vurderinger som legges til grunn for hvordan systemet agerer i de forskjellige situasjoner. Hvis for eksempel sluttbrukeren ønsker å sende henvisning til en bestemt mottaker må systemet tilby å sende meldingen på annen forsvarlig måte (f.eks. på papir) dersom mottakeren ikke kan nås elektronisk.

4.4 Samhandlingsprofiler og samhandlingsavtaler

Begrepene *samhandlingsprofil* (CPP – Collaboration Protocol Profile) og *samhandlingsavtale* (CPA – Collaboration Protocol Agreement) er grunnleggende for forståelsen av API-ene. Selve begrepene er hentet fra den internasjonale CPP/CPA-standarden og ettersom Adresseregisteret bruker strukturen herfra internt er begrepene også brukt i navngivningen av metoder og klasser i det forenklede API-et.

Alle kommunikasjonsparter har en *samhandlingsprofil*. Samhandlingsprofilen er en informasjonsstruktur som inneholder følgende:

- Kommunikasjonspartens identitet
- Hvilke prosesser kommunikasjonsparten er i stand til å utføre
- Hvilke roller kommunikasjonsparten kan inneha i de ulike prosessene
- Kommunikasjonskanal, protokoller og tilhørende parametere (gitt av prosessen/rolle)
- Hvilke sertifikater som skal benyttes for kryptering og signering
- Pakking av filer (hvordan filene er satt sammen av header, payload, sikkerhet osv.) (gitt av prosess/rolle)

Før to kommunikasjonsparter kan sende elektroniske meldinger til hverandre skal det etableres en *samhandlingsavtale*. Samhandlingsavtalen genereres av Adresseregisteret basert på snittet av to kommunikasjonsparters samhandlingsprofiler. På tross av begrepet *avtale* er det altså ikke snakk om et dokument e.l. som begge parter må enes om og signere på, men snarere en datastruktur som Adresseregisteret genererer og som forteller hvilke kapabiliteter de to har, ut fra hva som er registrert i Adresseregisteret. Samhandlingsavtalen forteller hvilke roller de to kommunikasjonspartene kan inneha og dermed hvilke meldingstyper og -versjoner de kan utveksle.

En samhandlingsavtale inneholder følgende:

- De to kommunikasjonspartenes identiteter
- Hvilke prosesser samhandlingsavtalen gjelder for
- Hvilke roller kommunikasjonspartene kan inneha i de ulike prosessene
- Hvilket tidsrom avtalen gjelder for
- Hvilken status avtalen har
- Kommunikasjonskanaler, protokoller og tilhørende parametere
- Hvilke sertifikater som skal benyttes for kryptering og signering
- Pakking av filer (hvordan filene skal settes sammen av header, payload, sikkerhet osv.)

5. ADMINISTRERE EGEN SAMHANDLINGSPROFIL

For å administrere en kommunikasjonsparts samhandlingsprofil er det utviklet en web service som eksponerer de metodene man trenger for å opprette og fjerne referanser til prosesser fra samhandlingsprofilen. Disse er beskrevet i henholdsvis avsnitt 5.1 og 5.2. Ettersom også referanser til sertifikater inngår i samhandlingsprofilen er sertifikatbytte omtalt i et eget avsnitt (5.3). Alle metodeangivelser i dette kapitlet viser til metoder denne web service-en tilbyr.

Når en kommunikasjonsparts samhandlingsprofil endres vil Adresseregisteret begynne å utstede samhandlingsavtaler basert på den nye samhandlingsprofilen. Samhandlingsavtaler basert på forrige versjon av samhandlingsprofilen vil ikke lenger bli utstedt. De gamle samhandlingsavtalene vil bare eksistere så lenge det finnes meldinger i flyt som refererer til dem. Så langt har ikke Adresseregisteret noen mekanisme for å distribuere tilbakekallingsinformasjon aktivt. Av den grunn er det viktig at mønstrene for sending og mottak av melding beskrevet i kapittel 6 og 7 følges. Dette innebærer at både sender og mottaker må etterspørre samhandlingsavtaler fra Adresseregisteret regelmessig, innenfor de retningslinjer for caching som er angitt i kapittel 8.

5.1 Legge til støtte for prosess

Et eksempel kan være at det skal implementeres funksjonalitet for å kunne sende og motta henvisninger i et system – dvs. at systemet skal støtte prosessen «Henvisning», og at det er versjon 1.1 av prosessen det skal implementeres støtte for. I så fall må samtlige installasjoner av systemet etter installasjon støtte både sending og mottak av henvisninger med versjon 1.1. Det skal lages et automatisk konfigurasjonsprogram («post installation script») som skal oppdatere Adresseregisteret ved installasjon av den nye funksjonaliteten.

I Referansekatalogen for e-helse[1] og Sarepta[2] har utvikleren funnet nødvendig spesifikasjon for å utvikle støtte for meldingstypene og -versjonene som inngår i prosessen. For å lage konfigurasjonsprogrammet er det behov for kunnskap om prosessens ID og navn på rollene for å motta og sende henvisninger. Utvikleren henter denne informasjonen fra Adresseregisteret ved å kjøre metoden `GetAllCollaborationRoles()`[3] i et utviklingsverktøy. Ut fra objektet som returneres finner utvikleren at den aktuelle prosessen har id `4d9a222f-5076-4221-8668-438277d2cf8a`, versjon `1.1` og rollene `HENVISNINGsender` og `HENVISNINGreceiver`.

Eksempelkode:

[1] https://ehelse.no/standarder-kodeverk-og-referansekatalog/referansekatalogen
[2] https://sarepta.ehelse.no
[3] https://register-web.test.nhn.no/docs/api/html/215516b4-166d-fd70-0072-dea84c853174.htm

```
var collaborationRoles = client.GetAllCollaborationRoles();
```

Vi er kun interessert i å finne prosessen «Henvisning» og kan gjøre dette med følgende LINQ-spørring:

```
collaborationRoles
.Where(cr => cr.ProcessSpecification.Name=="Henvisning")
.Select(cr => cr.RoleName + " " + cr.ProcessSpecification.Uuid + " " +
cr.ProcessSpecification.Version)
```

Siden prosessen «Henvisning» har to roller og det finnes flere versjoner gir dette følgende resultat:

```
[0]   "HENVISNINGreceiver 4d9a222f-5076-4221-8668-438277d2cf8a 0.9"
[1]   "HENVISNINGsender   4d9a222f-5076-4221-8668-438277d2cf8a 0.9"
[2]   "HENVISNINGreceiver 4d9a222f-5076-4221-8668-438277d2cf8a 0.91"
[3]   "HENVISNINGsender   4d9a222f-5076-4221-8668-438277d2cf8a 0.91"
[4]   "HENVISNINGreceiver 4d9a222f-5076-4221-8668-438277d2cf8a 1.0"
[5]   "HENVISNINGsender   4d9a222f-5076-4221-8668-438277d2cf8a 1.0"
[6]   "HENVISNINGreceiver 4d9a222f-5076-4221-8668-438277d2cf8a 1.1"
[7]   "HENVISNINGsender   4d9a222f-5076-4221-8668-438277d2cf8a 1.1"
```

Utvikleren har da det som trengs for å programmere konfigurasjonsprogrammet til å legge til en ny prosessreferanse i kommunikasjonspartens samhandlingsprofil ved hjelp av metoden `AddProcessToCpp(processParameter)`[4], der `processParameter` består av flere verdier:

- HER-id: HER-id for den kommunikasjonsparten samhandlingsprofilen skal oppdateres for
- Guid: Uuid på prosessen som skal støttes
- Version: Versjonsnummeret for prosessen som skal støttes
- Roles: Liste over tilhørende roller som skal støttes

Eksempelkode:

```
client.AddProcessToCpp(new ProcessParameter()
{
      HerId = Config.HerId,
      Guid = new Guid("4d9a222f-5076-4221-8668-438277d2cf8a"),
      VersionNumber = "1.1",
      Roles = new List<Role> {
           new Role() { Name = "HENVISNINGsender" },
           new Role() { Name = "HENVISNINGreceiver" }
      }.ToArray()
});
```

5.2 Fjerne støtte for prosess

Et annet eksempel kan være at systemleverandøren utvikler en administrasjonsmodul som tillater systemadministrator å fjerne rolleangivelser fra kommunikasjonspartens samhandlingsprofil.

Utvikleren har da behov for følgende metoder i Adresseregisterets API:

- GetCPPForCommunicationParty(herid)[5] – returnerer kommunikasjonspartens samhandlingsprofil. Administrasjonsmodulen bruker denne for å kunne presentere administratoren for nåværende tilstand (hvilke prosesser og roller kommunikasjonsparten har oppgitt å støtte).

[4] https://register-web.test.nhn.no/docs/api/html/af3cf8db-3500-3732-28a9-4b0718cd9dbe.htm
[5] https://register-web.test.nhn.no/docs/api/html/a93b47b9-e5e3-75d7-185f-922cf595f24e.htm

- RemoveProcessFromCpp(ProcessParameter)[6] – fjerner en gitt rollereferanse fra HER-id. ProcessParameter består av følgende:
  - HER-id for den kommunikasjonsparten samhandlingsprofilen skal oppdateres for
  - Guid: Uuid på prosessen som kommunikasjonsparten skal ha endret støtte for
  - Version: Versjonsnummeret for prosessen som kommunikasjonsparten skal ha endret støtte for
  - Roles: Liste over tilhørende roller som ikke lenger skal støttes

Eksempelkode:

Hente samhandlingsprofil for å vise frem i GUI:

```
var cpp = client.GetCppForCommunicationParty(Config.Herid);
```

Fjerne rolle for mottak av henvisning:

```
client.RemoveProcessFromCpp(new ProcessParameter()
{
	HerId = Config.Herid,
	Guid = new Guid("4d9a222f-5076-4221-8668-438277d2cf8a"),
	VersionNumber = "2.0",
	Roles = new List<Role>
	{
		new Role() { Name = "HENVISNINGreceiver" }
	}.ToArray()
});
```

5.3 Sertifikatbytte

I Adresseregisteret ligger det for hver kommunikasjonspart informasjon om hvilke krypterings- og signeringssertifikater som skal benyttes. Informasjonen foreligger i form av en LDAP-peker inn i sertifikatutsteders (f.eks. Buypass') katalog. Ettersom kommunikasjonspartenes krypterings- og signeringssertifikater inngår i de enkelte samhandlingsprofilene og -avtalene vil sertifikatbytte påvirke både samhandlingsprofiler og -avtaler.

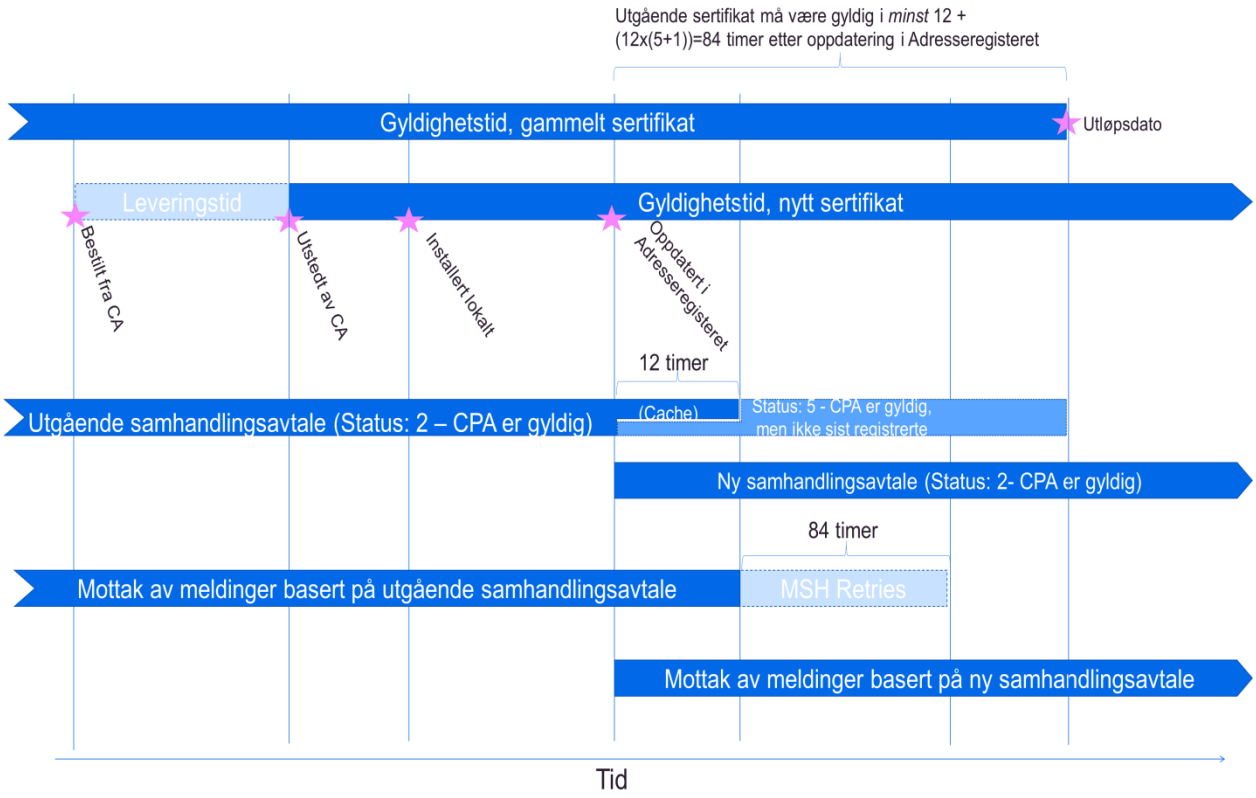
Merk! Adresseregisteret tillater p.t. bruk av generiske LDAP-pekere som ikke peker på *ett bestemt* sertifikatpar (for signering og kryptering), men snarere på *alle* virksomhetens sertifikater. Ulempen med å bruke generiske pekere er at virksomheten ikke har kontroll på når sertifikatbytte effektueres i Adresseregisteret. Man kan risikere at sertifikatbyttet effektueres *før* virksomheten har fått installert privatdelen av sertifikatene i sine systemer. Virksomheten vil dermed kunne motta krypterte meldinger den (ennå) ikke er i stand til å dekryptere. For å unngå dette problemet er det derfor anbefalt at LDAP-pekeren inneholder serienummer, slik at den peker på ett bestemt sertifikatpar.

Fra det øyeblikket en kommunikasjonspart oppdaterer Adresseregisteret med en ny LDAP-peker (som peker på det nye sertifikatparet) vil Adresseregisteret begynne å utstede samhandlingsavtaler som inneholder de nye sertifikatene. Kommunikasjonsparten må derfor anta at det går kort tid fra Adresseregisteret er oppdatert til den begynner å motta meldinger som er kryptert med det nye sertifikatet. Ettersom samhandlingsavtaler kan caches i inntil 12 timer (se kapittel 8) og meldinger har en standard levetid på 72 timer[7], vil kommunikasjonsparten i inntil 12+72=84 timer etter at Adresseregisteret

[6] https://register-web.test.nhn.no/docs/api/html/f933468d-d98c-f45b-af8b-22f2f01871e2.htm

[7] Levetiden til ebXML meldinger er beskrevet i HIS 1037:2011: Rammeverk for elektronisk meldingsutveksling i helsevesenet basert på ebXML, i kapittelet 5.2 Pålitelig meldingsutveksling. Enkelte applikasjoner (for eksempel reseptformidleren) forutsetter en levetid som er enda lenger (i reseptformidlerens tilfelle 144 timer). Kommunikasjonsparter som benytter disse tjenestene må oppdatere sine sertifikater desto tidligere i forhold til utløpsdato.

er oppdatert kunne motta meldinger som er kryptert med det utgående sertifikatet. I denne perioden må kommunikasjonsparten derfor kunne dekryptere med både det gamle og det nye sertifikatet. Dette betyr også at kommunikasjonsparten må oppdatere Adresseregisteret senest 108 timer før det gamle sertifikatet utgår. Se Figur 1 som fremstiller dette skjematisk.



Figur 1 Tidslinje for sertifikatoppdatering

6.SENDE MELDING

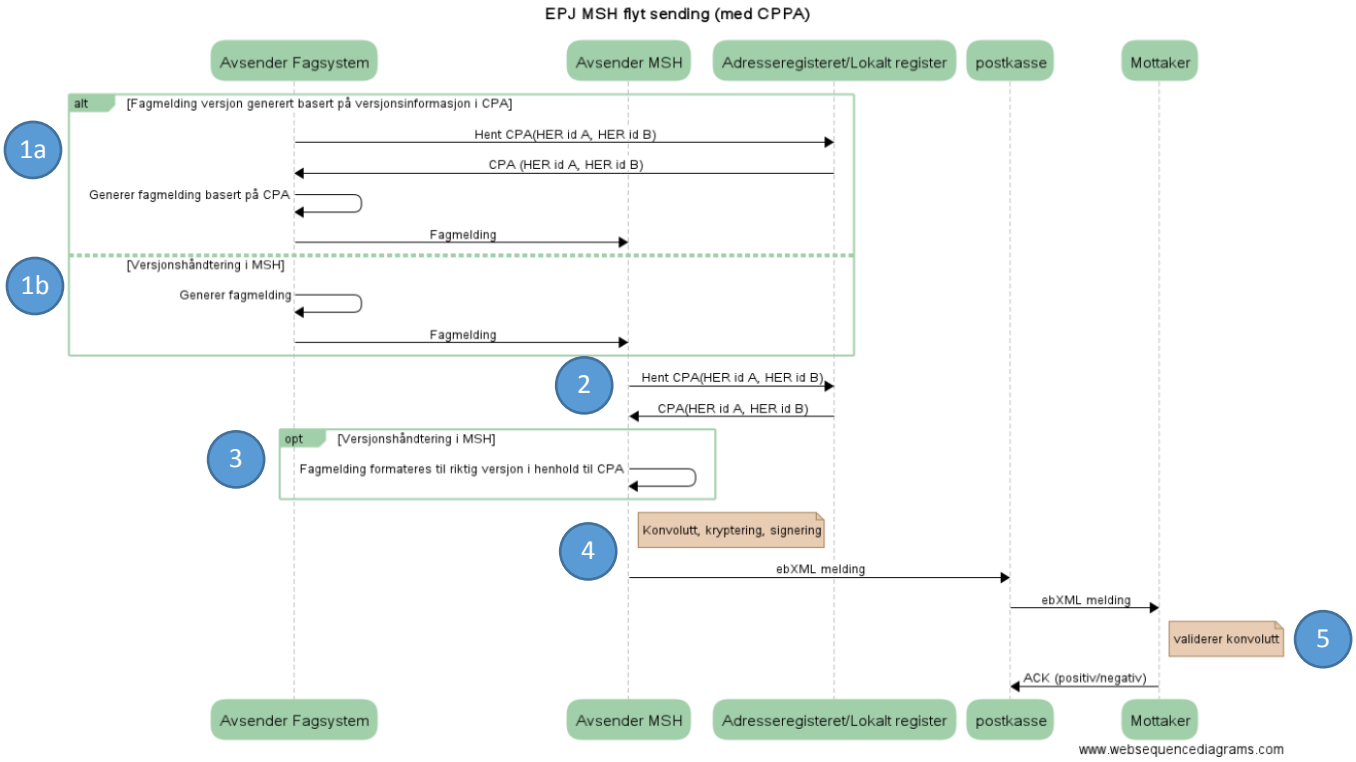
Sending og mottak av melding er hovedbruksområdet for CPA. Når et system skal sende en melding må systemet først velge meldingstype/-versjon (i den grad systemet støtter flere ulike typer og versjoner), signere meldingen med avsenders riktige sertifikat og kryptere meldingen med mottakers riktige sertifikat. Systemet finner informasjon om dette i CPA-en som gjelder for de to kommunikasjonspartene (avsender og mottaker). CPA-ens uuid skal legges i ebXML-konvoluttens CPA_ID-felt slik at mottaker kan verifisere at en gyldig CPA ble lagt til grunn for forsendelsen.

Inntil samtlige leverandører har implementert støtte for oppdatering av samhandlingsprofiler (se kap. 5) vil det være tilfeller der Adresseregisteret ikke vil kunne generere CPA for avsender/mottaker. I disse tilfellene må avsendersystemet avgjøre om a) meldingen skal stoppes (med varsling til bruker), b) om meldingen skal sendes på papir eller c) om meldingen skal sendes på tross av at det ikke foreligger en CPA. Inntil man har oppnådd «kritisk masse» for bruk av kommunikasjonsparametere vil alternativ c) være å foretrekke i de fleste tilfeller. Dette innebærer at meldingsutvekslingen vil fortsette som i dag i de tilfellene der mottaker ikke har registrert samhandlingsprofil.

Bruk og oppsett vil variere litt ut ifra hvor og hva man skal implementere. For eksempel vil bruken være litt forskjellig om man skal utvikle i et fagsystem eller i mellomvare. Sekvensdiagrammene og use casene som følger vil derfor ikke nødvendigvis dekke alle eksempler, men de vil gi en forståelse av hvordan det fungerer.

CPA-er utstedes via Adresseregisteret og må hentes fra Adresseregisterets webservice. En kan velge å hente ned CPA-er enkeltvis, og/eller hente ned alle CPA-er for en gitt kommunikasjonspart (evt. med valgfritt fra-tidspunkt). Dette muliggjør at en også kan benytte lokalt register/cache for påfølgende bruk. Hvis en velger denne løsningen, så er det viktig at en holder CPA-ene oppdatert og ikke overstiger maksimal cachetid på 12 timer. Se avsnitt 8 for ytterligere informasjon.

6.1 Sekvensdiagram ved sending av melding



1. Ved sending av melding kan enten fagsystemet selv ha et aktivt forhold til Adresseregisteret eller den kan la mellomvaren(MSH) ta seg av det.
    a. I det første tilfellet henter *fagsystemet* CPA basert på sender og mottakers HER-id:

```
var cpa = arClient.GetCpaForCommunicationParties(senderHerId, mottakerHerId)[8];
```

    Objektet `cpa` vil inneholde hvilke prosesser sender og mottaker begge støtter.
    Ut i fra informasjonen i den aktuelle CPA'en må fagsystemet generere en fagmelding av en meldingstype og -versjon som mottaker støtter.
    For mer informasjon om oppbygningen av CPA-objektet, se [1].
    b. Når versjonshåndtering skjer i MSH vil fagsystemet generere en melding på fagsystemets format.
2. MSH henter CPA fra Adresseregisteret.
3. Om fagmeldingen ble levert fra fagsystemet i fagsystemets format må MSH bruke den aktuelle CPA'en for å finne sender og mottakers felles meldingstyper og versjon av disse. Ut i fra hva som er angitt i CPA formateres det en fagmelding mottaker kan motta. Dersom det ikke er mulig å finne en felles meldingstype og -versjon mellom avsender og mottaker må meldingen sendes på annen forsvarlig måte, for eksempel på papir. MSH må varsle fagsystemet om dette.
4. Mottaker hentes ut fra meldingen før den krypteres. Hvis det er et krav om at meldingen skal signeres, skal den signeres med aktørens personlige signeringssertifikat *før* fagmeldingen krypteres med mottakers krypteringssertifikat. Den krypterte meldingen pakkes i en ebXML-konvolutt. ebXML-konvolutt fylles ut med nødvendige felt (bl.a. avsender, mottaker) og CPA_ID settes til CPA'ens UUID. ebXML-meldingen signeres og sendes mottakers postkasse.
5. Mottaker validerer ebXML-konvolutt og sender transportkvittering tilbake dersom den validerer OK. I motsatt fall sendes en feilmelding.

[8] https://register-web.test.nhn.no/docs/api/html/055a24ce-5474-8dec-7e67-18f009dac562.htm

6.2 Eksempel på sending av melding

Pseudokoden under viser et eksempel på hvordan sending av melding foregår:

MSH/mellomvare:

I det første tilfelle mottar MSH en ikke-behandlet melding fra fagsystemet og må gjøre alle operasjoner:

```
void SendMessage(message) {
	var cpa = cppa.GetCpa(message.senderHerId, message.receiverHerId);
	if (!cpa || !cpa.Active || cpa.codeValue != 2)
		throw new Exception("Fant ikke CPA eller CPA er ugyldig");
	//Sjekk om mottaker støtter melding og versjon
	bool validMessage = ValidateMessage(message, cpa);
	if (!validMessage)
		throw new Exception("Motaker støtter ikke melding");
	//Signering, kryptering og pakking
	var ebXML = SignAndCrypt(message, cpa);
	ebXML.cpa_id = cpa.id;
	transport.SendMessage(ebXML);
}
```

I det andre tilfelle har allerede fagsystemet sjekket mottakers CPA slik at sendingsprosessen blir mindre omfattende:

```
void SendMessage(formattedMessage, cpa) {
	//Signering, kryptering og pakking
	var ebXML = SignAndCrypt(message, cpa);
	ebXML.cpa_id = cpa.id;
	bool sent = transport.SendMessage(ebXML);
	if (!sent)
		throw new Exception("Feilet ved sending av melding");
}
```

Fagsystem:

Følgende pseudokode passer til det andre tilfellet over:

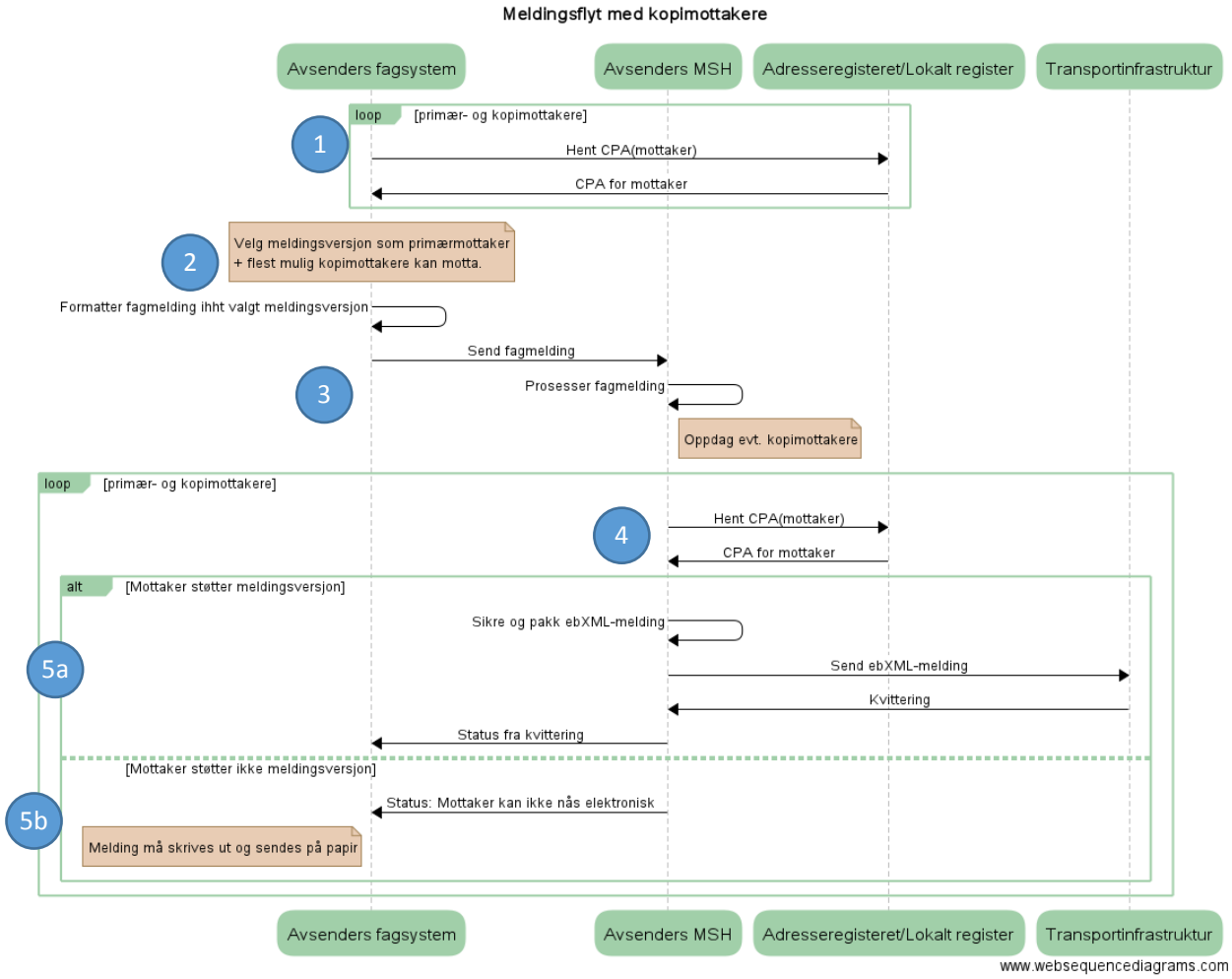
```
void SendMessage(fromHerid, toHerId, message) {
	var cpa = cppa.GetCpa(fromHerid, toHerId);
	if (!cpa || !cpa.Active || cpa.codeValue != 2)
		throw new Exception("Fant ikke CPA eller CPA er ugyldig");
	bool validMessage = ValidateMessage(message, cpa);
	//Sjekk om mottaker støtter melding og versjon
	if (!validMessage)
		throw new Exception("Motaker støtter ikke melding");
	msh.SendMessage(message, cpa);
	//Fortell bruker at melding er sendt
}
```

6.3 Sekvensdiagram ved sending av melding med kopimottakere

Obs! Dette og neste kapittel (6.4) legger gjeldende praksis for kopimottakere til grunn. Bruk av kopimottaker beskrives blant annet i HIS 80415:2012 Applikasjonskvittering. En konsekvens av etablert praksis er at kopiene må være syntaktisk like – i den forstand at de må være av samme meldingstype og -versjon. Det finnes også eksempler på applikasjoner som tilbyr «kopimottaker-aktig» funksjonalitet ved å iterere over et sett mottakere og generere en ny melding for hver av dem. Disse meldingene behøver ikke å være syntaktisk like (de kan i prinsippet være av ulik meldingstype, eller –versjon), men de vil typisk

være semantisk like. «Kopimottaker-aktig» funksjonalitet er ikke omtalt nærmere her, men vil være en variant av sekvensdiagrammet beskrevet i kapittel 6.1.

Ved sending til kopimottakere (i henhold til rammeverkets definisjon) må løsningen finne ut hvilken meldingsversjon som skal benyttes, tatt i betraktning at de ulike mottakerne kan støtte ulike versjoner av meldingstypen. Ettersom en CPA omhandler kun to parter (avsender og mottaker) må løsningen hente flere CPA-er for å avgjøre hvilken versjon som skal benyttes:



I dette sekvensdiagrammet er det lagt til grunn at fagsystemet har egen kobling mot Adresseregisteret, jf. Punkt 1a i «6.1 Sekvensdiagram ved sending av melding».

1. Det hentes en CPA for hver mottaker det skal utveksles melding med.
2. Når alle CPA'er er hentet vet fagsystemet hvilket versjonsnummer av meldingen de ulike mottakere støtter. Ut fra denne informasjonen må systemet velge hvilken versjon som skal benyttes, dersom systemet støtter flere versjoner. Kommunikasjonsparter som ikke støtter valgt versjon må få tilsendt meldingen på annen forsvarlig måte, for eksempel på papir. Se også avsnitt 6.4.
3. Fagmeldingen sendes MSH
4. For hver mottaker henter MSH en CPA.
5. Ut ifra om mottaker støtter den aktuelle meldingsversjonen avsenders fagsystem eller MSH støtter, vil systemet:

a. Om mottaker støtter meldingsversjonen skal systemet pakke og signere meldingen som i 6.1.4 og deretter motta transportkvittering fra mottaker.
b. Om mottaker ikke støtter meldingsversjonen må fagsystemet få beskjed om dette slik at meldingen kan sendes på annen forsvarlig måte, for eksempel på papir.

6.4 Valg av versjon ved sending av melding

Ved sending av melding kan man komme i situasjoner hvor mottaker ikke støtter en bestemt meldingtype/-versjon. Om meldingstypen/-versjonen ikke støttes må meldingen sendes på annen forsvarlig måte, for eksempel på papir.

Dersom et system skal sende til flere mottakere må systemet sjekke hva hver enkelt er i stand til å motta. En CPA inneholder informasjon om dette. Mottakere som støtter meldingsversjonen avsender velger vil få meldingen elektronisk. Er det én eller flere mottakere som ikke støtter den meldingsversjonen avsender velger, må meldingen sendes på annen forsvarlig måte, for eksempel på papir til disse.

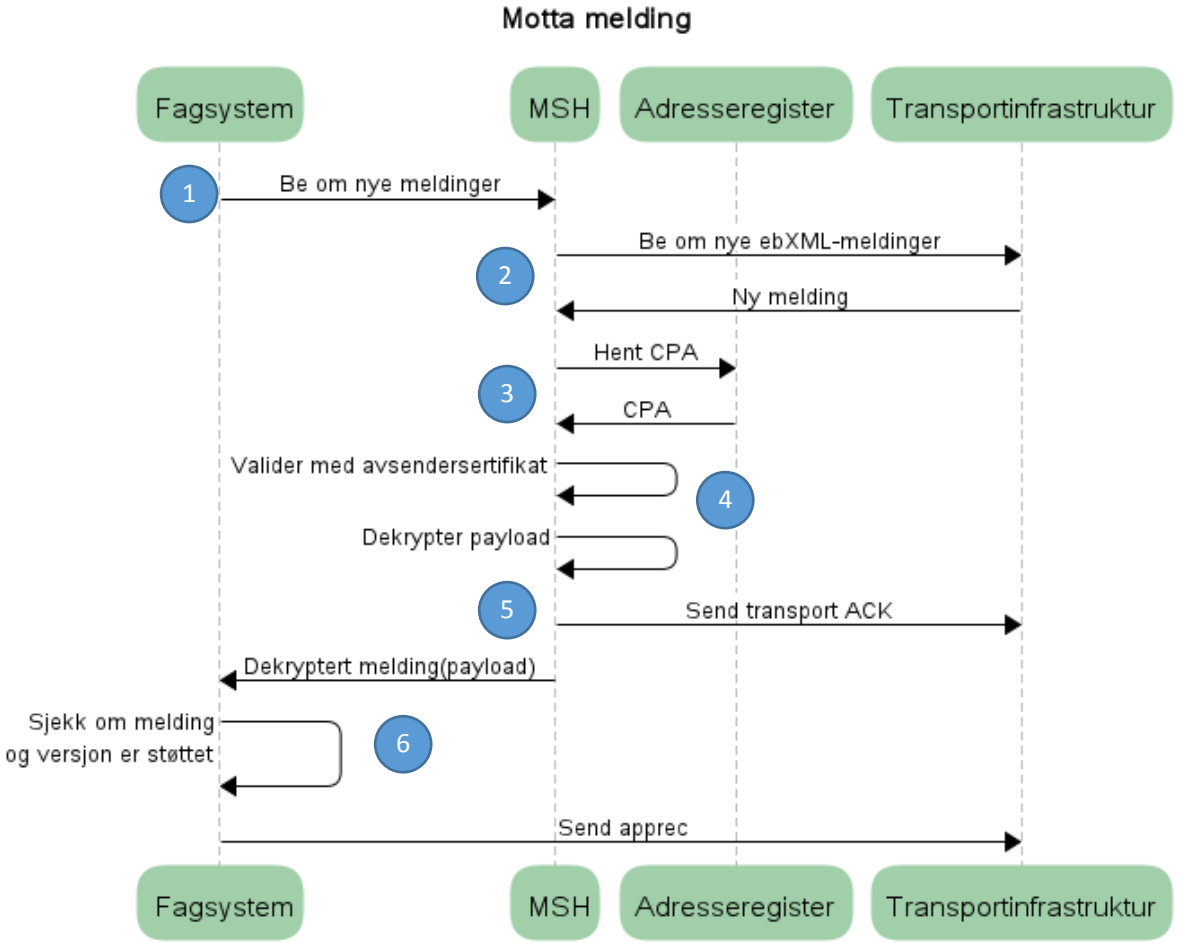
Et avsendersystem som støtter flere meldingsversjoner vil kunne ha flere strategier for å velge hensiktsmessig versjon ut fra informasjon om hvilke versjoner mottakerne støtter. I noen tilfeller vil hovedmottakers støtte være førende for hvilken versjon som velges, i andre tilfeller velges den versjonen som flest mottakere totalt sett støtter. Mottakere som ikke støtter valgt versjon må uansett få tilsendt meldingen på annen forsvarlig måte, for eksempel på papir.

7. MOTTA MELDING

Ved mottak av meldinger vil man også få bruk for CPA. Mottaker benytter informasjon fra CPA om hvilke sertifikater som skal benyttes for dekryptering og verifikasjon av signatur. Videre må mottaker verifisere at den mottatte meldingen er i henhold til oppgitt CPA og at CPA-en fremdeles er gyldig. Dette gjøres ved å hente CPA-ens uuid fra ebXML-konvoluttens CPA_ID-felt, for så å hente ut den aktuelle CPA-en fra Adresseregisteret.

7.1 Sekvensdiagram

Sekvensdiagrammet viser henting av meldinger fra server.



1. Fagsystemet ber om å få nye meldinger fra transportlaget. Om det finnes meldinger som ikke er hentet fra før blir disse returnert.
2. ebXML-meldingene har et felt som heter CPA_ID. Feltet består av en UUID som er id'en til en CPA.
3. CPA hentes inn:

```
var cpa = client.GetCpa(uuid)[9];
```

CPA-objektet har én av følgende stautsverdier:

1. CPA er foreslått. Venter på godkjenning (signering). Brukes kun unntaksvis, hvis én av partene krever signerte (godkjente) samhandlingsavtaler.
2. CPA er gyldig
3. CPA er revokert
4. Gyldighetsperioden til CPA-en er utgått
5. CPA er gyldig, men ikke sist registrerte

Hvis status er 2 eller 5 går man videre i flyten, ellers sendes feilmelding (ebXML Error).

4. Ved hjelp av sertifikatene valideres og dekrypteres payloaden i meldingen.
5. MSH sender transportkvittering (ebXML Acknowledgment) til mottakeren på at meldingen er mottatt og validert ok. Den dekrypterte meldingen returneres/sendes til fagsystem.
6. Fagsystemet sjekker om det forstår melding ut fra meldingstype og versjon. Etter at meldingen er persistert sender Fagystemet en applikasjonskvittering til avsender for å bekrefte at at fagmeldingen er kommet frem, at innholdet er i rett format, og at meldingen er klar for behandling i mottakers fagsystem. Sending og mottak av applikasjonskvittering gjøres også ved hjelp av CPPA. For nærmere beskrivelse av betydningen og bruk av applikasjonskvittering, se [2].

7.2 Eksempel på mottak av melding

Eksempel: En kommunikasjonspart har et fagsystem som skal sjekke om det finnes uleste meldinger i systemet.
Pseudokode for systemet blir som følger:

MSH/mellomvare:

```
Messages[] PopMessageFromServer() {
   var msg = transport.PopMessage();
   if (msg) {
      var cpa = cppa.GetCpa(msg.cpa_id);
      if (cpa finnes ikke eller cpa.Status er noe annet enn 2 eller 5) {
         transport.SendError(msg.sender);
         throw new Exception("Fant ikke CPA eller CPA er ugyldig");
      }
      var valid = ValdiateMessage(msg, cpa);
      var decryptedMessage = DecryptMessage(msg, cpa);

      if (valid)
         transport.SendTransportAck(msg.sender);
      else {
         transport.SendError(msg.sender);
         throw new Exception("Meldingen validerte ikke");
      }

      return decryptedMessage;

   }

}
```

[9] https://register-web.test.nhn.no/docs/api/html/781cea98-6490-4453-92c6-fbd4250d1bf7.htm

Metodene ValidateMessage() og DecryptMessage() skal bruke sertifikatene fra CPA for å verifisere meldingen.

Fagsystem:

```
void PopMessageFromMiddleware() {
   try {
      var newMessage = middleware.PopMessageFromServer();
      if (newMessage && IsMessageSupported(newMessage.type, newMessage.version) {
         SendApprec(newMessage);
         ShowMessageToUser(newMessage);

      }
   }
   catch(Exception e) {
      ShowErrorMessage(e.message);
   }
}
```

Pseudokoden over er kun et utgangspunkt. Det finnes mange varianter for hvordan fagsystem og EPJ samhandler og dette må tas hensyn til under implementeringen.

8. CACHING OG HÅNDTERING AV UTILGJENGELIGHET

Hver henting av CPA med tilhørende sertifikater fra Adresseregisteret tar tid og påfører belastning på både Adresseregisteret og endepunktene, så i de fleste tilfeller er det anbefalt å lagre disse lokalt i et mellomlager. For å unngå lang forsinkelse mellom endring av CPP (f.eks. fjerning av en prosess eller endring av sertifikat) og effektuering må imidlertid cachetiden ikke settes høyere enn 12 timer. Se også avsnitt 5.3, som forklarer hvordan cachetid har innvirkning på tidslinjen ved sertifikatbytte.

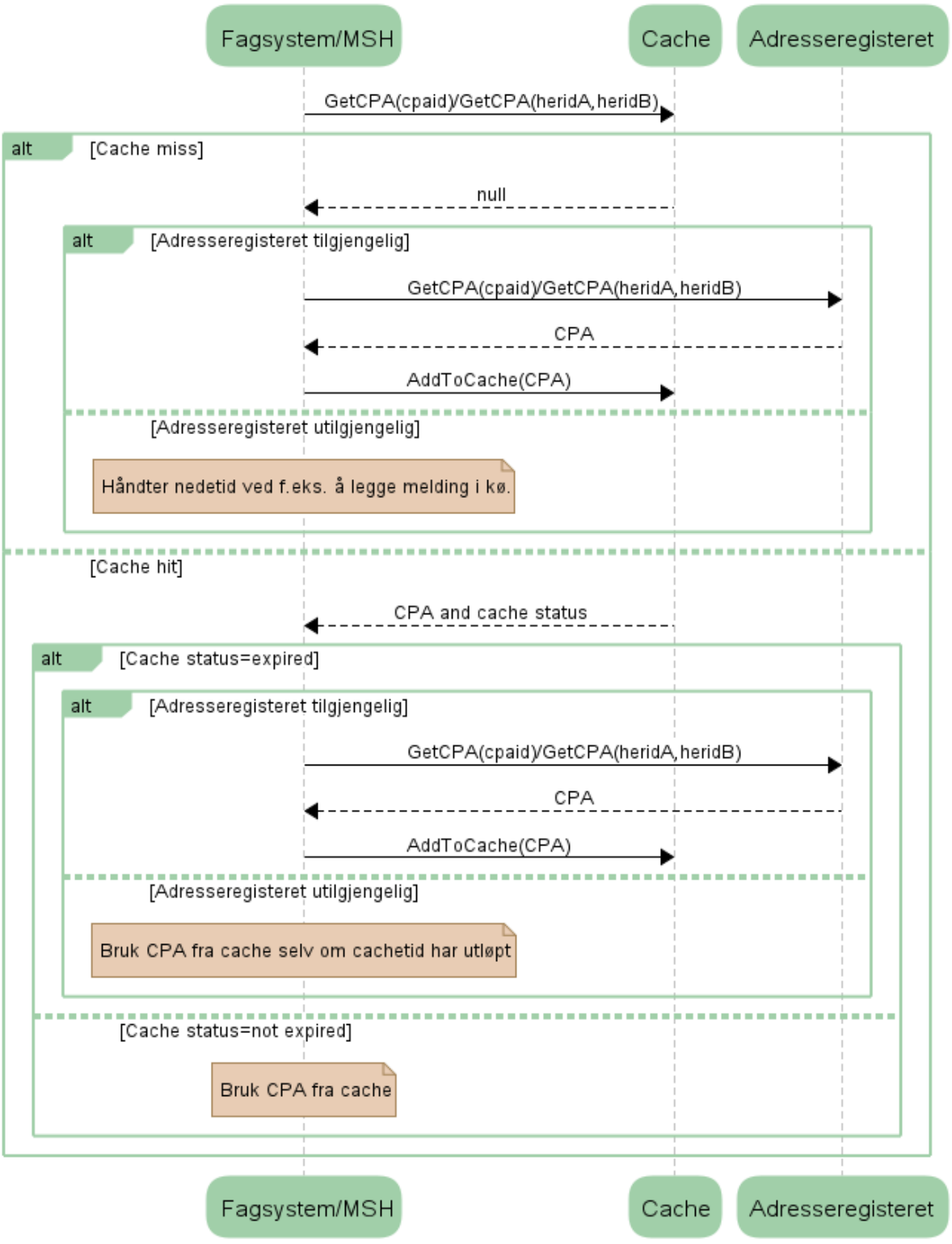
Bruk av cache i forbindelse med utilgjengelighet av Adresseregisteret
Mønsteret der alle meldingsutvekslinger baseres på en fersk CPA fra Adresseregisteret fører til at Adresseregisteret får en svært sentral rolle. For å unngå at Adresseregisteret representerer et logisk «single point of failure» for all meldingsutveksling i sektoren er det viktig å etablere reservemekanismer som kan benyttes dersom Adresseregisteret blir utilgjengelig i en periode. Den enkelte installasjons cache kan fungere som en slik reservemekanisme – gitt at den designes og implementeres med tanke på nettopp dette. Konkret betyr det at cachen må etableres ut fra følgende prinsipper:

- Cachen må persisteres på disk e.l. slik at den er tilgjengelig også etter omstart
- Alle CPA-er hentet fra Adresseregisteret må legges i cache – både ved sending og mottak av meldinger
- Systemet må kunne søke i cache med både cpaid og herid-par som nøkler

Normal prosess for å hente *nye* CPA vil være å hente dette direkte fra Adresseregisteret.
For å vedlikeholde lokal cache/lokalt register med CPA-er i etterkant så kan en benytte metoden GetCpasForCommunicationParty(herId, createdAfter)[10]. Denne metoden vil returnere en liste med CPAer som er opprettet/endret etter ønsket tidspunkt på angitt kommunikasjonspart (herid).
Om en benytter denne to ganger i døgnet så vil en kunne dekke behovet for 12 timers cache.

I tilfeller der Adresseregisteret er utilgjengelig skal systemet kunne bruke en CPA fra cache – selv om maksimal cachetid i utgangspunktet er oversteget. Dette innebærer at alle etablerte samhandlingsmønstre kan fortsette selv om Adresseregisteret er utilgjengelig. Nye og endrede samhandlingsmønstre (kommunikasjonsparter som ikke tidligere har samhandlet eller samhandlingsmønstre der en av kommunikasjonspartene akkurat har endret sin samhandlingsprofil) vil først kunne etableres når Adresseregisteret er tilgjengelig igjen. Sekvensdiagrammet under illustrerer dette:

[10] Lanseres i 2017


Fagsystem/MSH
Cache
Adresseregisteret
GetCPA(cpaid)/GetCPA(heridA,heridB)
alt
[Cache miss]
null
alt
[Adresseregisteret tilgjengelig]
GetCPA(cpaid)/GetCPA(heridA,heridB)
CPA
AddToCache(CPA)
[Adresseregisteret utilgjengelig]
Håndter nedetid ved f.eks. å legge melding i kø.
[Cache hit]
CPA and cache status
alt
[Cache status=expired]
alt
[Adresseregisteret tilgjengelig]
GetCPA(cpaid)/GetCPA(heridA,heridB)
CPA
AddToCache(CPA)
[Adresseregisteret utilgjengelig]
Bruk CPA fra cache selv om cachetid har utløpt
[Cache status=not expired]
Bruk CPA fra cache
Fagsystem/MSH
Cache
Adresseregisteret