SOSI-Gateway

Design og arkitektur

SUNDHEDSDATA-
STYRELSEN

Indhold

Version	Dato	Beskrivelse	Forfatter
1	21-06-2016	Første version, udarbejdet med udgangspunkt i Arkitektur og design for SOSIGW 1.0	OBJ

Indledning

Nærværende dokument indledes med en kort overordnet beskrivelse af, i hvilken kontekst SOSI Gateway (SOSI-GW) indgår på NSP-platformen.

I resten af dokumentet beskrives design og arkitektur for SOSI-GW komponenten.

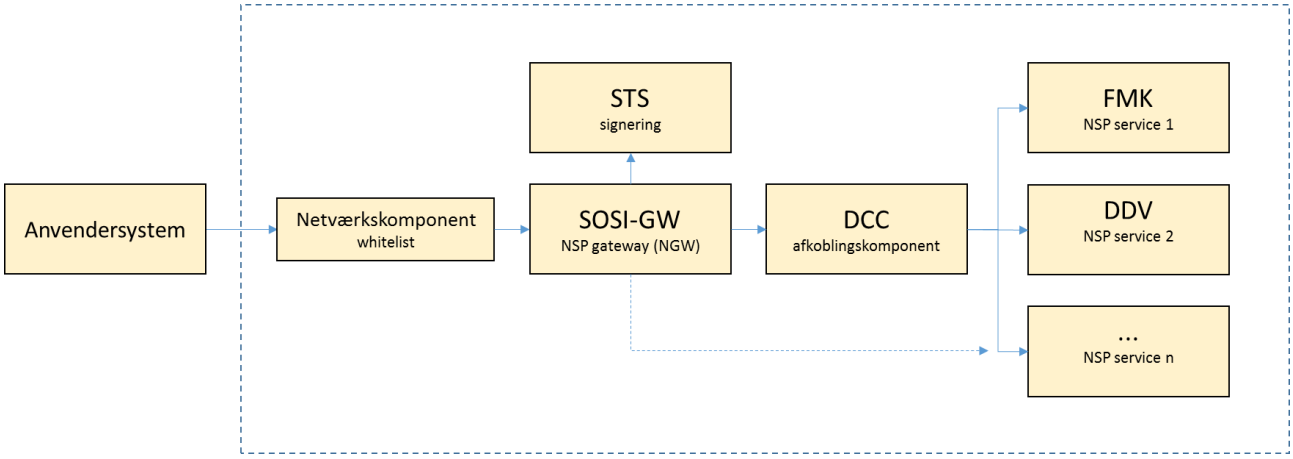
NSP-platformen – Kontekst for SOSI-GW

Sikkerhedsmodellen omkring kald til NSP services er baseret på Den Gode Webservice (DGWS), hvilket indebærer at der skal findes en SOAP-header med et signeret SOSI id-kort, sikkerhedsniveau 4.

Hvis anvendersystemer skulle kalde en NSP service direkte, ville alle anvendersystemer dermed være nødt til at håndtere signering af SOSI id-kort, hvilket kan være komplekst.

Det er her SOSI-GW kommer ind i billedet. I stedet for at alle anvendersystemer kalder NSP service endpoints direkte, kaldes i stedet igennem en central SOSI-GW kaldet "NSP Gateway". Signering af SOSI id-kort og viderestilling til konkrete endpoints håndteres dermed kun ét sted.

NSP arkitekturen består, udover SOSI-GW, af et antal øvrige komponenter. Figur 1 illustrerer i hvilken kontekst SOSI-GW er placeret på den centrale NSP platform (SSL variant af cNSP):



Figur 1: SOSI-GW kontekst på NSP platform

Komponenterne i arkitekturen er følgende:

Netværkskomponent Anvendersystemets kald til NSP går i praksis til en netværkskomponent, som kaldes via HTTPS. Denne komponent validerer kaldet og det medsendte FOCES-klientcertifikat i forhold til en whitelist. Kun godkendte forespørgsler sendes videre herfra til SOSI-GW.

Security Token Service Når et id-kort skal signeres med føderationens certifikat, kalder SOSI-GW STS'en. Herefter indsætter SOSI-GW id-kortet i en cache. Ved efterfølgende kald fra

anvendersystemet erstattes det medsendte id-kort med det signerede id-kort fra cachen, inden viderestilling til den relevante NSP service[1].

Afkoblingskomponent Som udgangspunkt kaldes der videre til den relevante NSP service igennem en afkoblingskomponent kaldet DCC (Decoupling Component). Afkoblingskomponenten vil ud fra beskedens SOAP action afgøre hvilket konkret endpoint, der skal kaldes.

NSP services Den konkrete NSP service vil oftest blive kaldt igennem DCC. Det er dog muligt for anvendersystemet at specificere den ønskede endpoint-URL selv i et tag'et "To" i SOAP-headeren WSAddressing. Hvis dette er tilfældet vil SOSI-GW kalde det specificerede endpoint udenom DCC.

Såfremt SOSI-GW placeres efter DCC i stedet for før, vil DCC tilføje en WsAddressing SOAP-header med "To" udfyldt. SOSI-GW vil i dette tilfælde derfor altid kalde NSP service endpoint direkte. Et sådant setup anvendes decentralt i regionerne.

For yderligere information om, hvorledes SOSI-GW indgår i NSP platformen, henvises til dokumentet "SOSI-GW Guide til anvendere".

Arkitektur for SOSI-GW

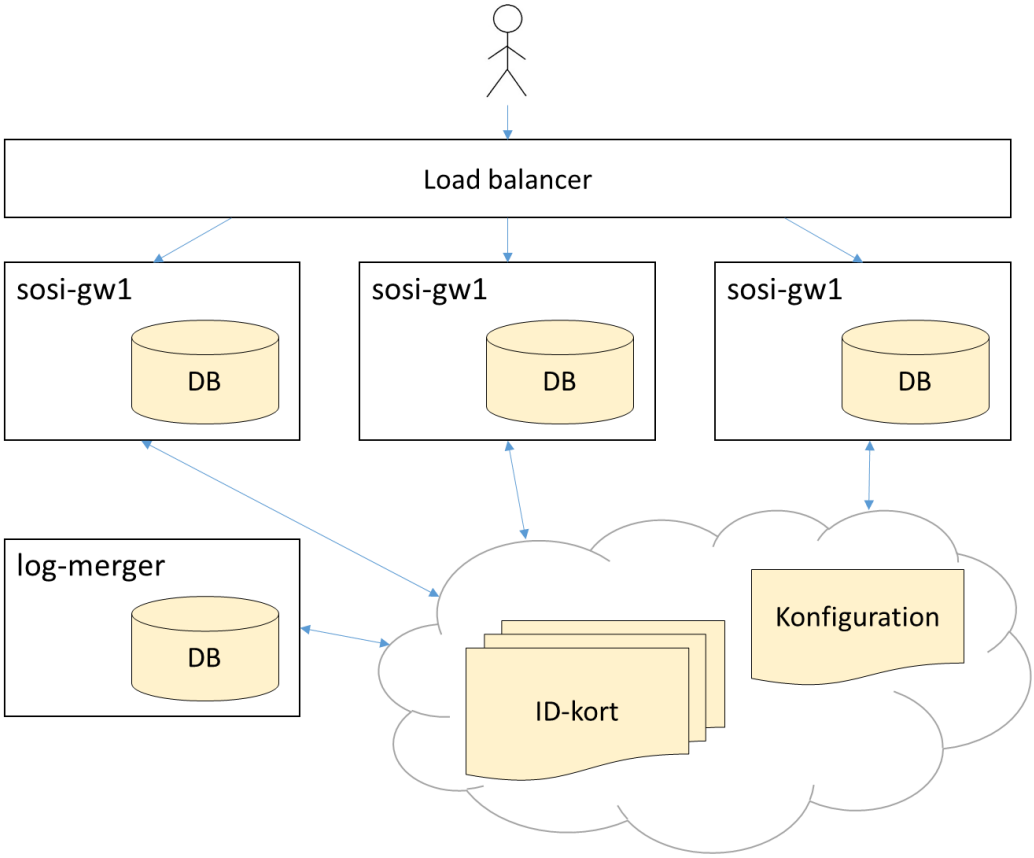
Den grundlæggende arkitektur for SOSI-GW er baseret på et cluster af servlet containere. Der er tale om en symmetrisk løsning, hvor alle servere i clusteret er identiske i opsætning og konfiguration. Clustering funktionaliteten opnås på applikationsniveau, der er derfor ikke krav om clustering support i den benyttede servlet container. Den nedre grænse for cluster-størrelse vil være en enkelt server, den øvre grænse i princippet ubegrænset. Der understøttes rullende opdatering, load balancering og fail-over ved en clusterstørrelse på 2 eller flere.

Hver server vil være bestykket med en servlet container (NSP benytter JBoss 6 og Wildfly 8.2) samt en lokal MySQL database server. Den lokale database anvendes udelukkende til at opsamle log-entries samt til lokal konfiguration. En knude i nettet kræver at begge komponenter (servlet container og database server) er kørende for at være funktionelle.

Til opsamling og søgning på audit logs opstilles en log-merger server, som ligeledes er bestykket med en lokal MySQL database.

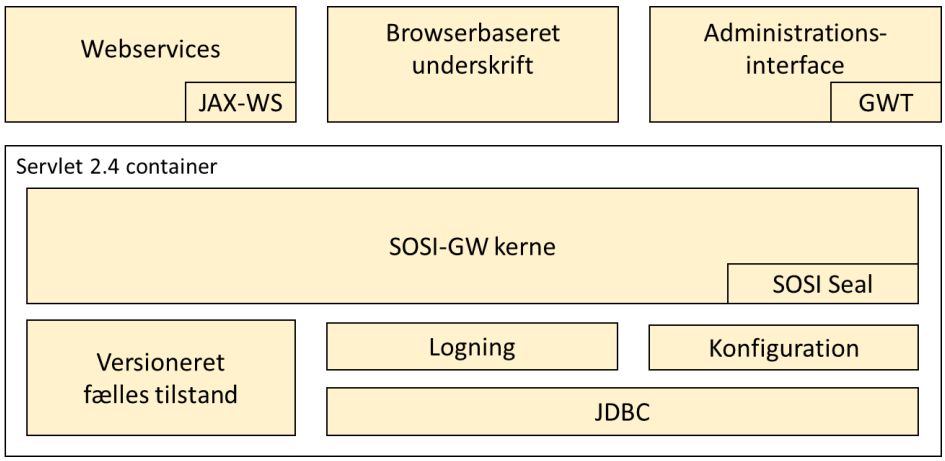
Figur 2 nedenfor skitserer SOSI-GW løsningen bestående af et antal gateway servere (sosi-gw1, … sosi-gw3) og en central database til opsamling og fletning af audit logs (log-merger).

[1] Hvis anvendersystemet har angivet Passthrough header, eller der sendes et id-kort med sikkerhedsniveau > 1, foretages ingen erstatning.



Figur 2: Cluster med tre SOSI-GW servere.

Intern arkitektur



Figur 3: SOSI-GW intern arkitektur

Figur 3 viser den interne arkitektur i SOSI-GW. Løsningen består af en kerne, der anvender SOSISeal til at håndtere id-kort. Id-kort distribueres over en fælles cache, der drives af multicasting på lokalnettet.

Konfiguration og logning gemmes i en lokal JDBC database. Ovenpå kernen ligger tre snitflader, nemlig webservices (JAX-WS), http til browserbaseret signering samt GWT (Google Web Toolkit) til administration.

Tilstandsdeling imellem clustermedlemmer

De servere, der indgår i et cluster, kommunikerer indbyrdes via et distribueret, delt hukommelseområde, realiseret via UDP multicast på lokalnettet. Herigennem overføres løbende opdateringer i form af oprettelse, opdatering og nedlæggelse af id-kort.

Udover id-kort, bruges clusteret også til at synkronisere konfigurationsændringer, der er foretaget igennem administrationskonsollen.

Multicast benyttes endvidere som discovery mekanisme, dvs. som måden hvorpå clustermedlemmer fastlægger hvilke øvrige medlemmer clusteret indeholder på et givet tidspunkt. Dette betyder, at der ikke er behov for manuel konfiguration i forbindelse med at en server tilføjes eller fjernes.

Versionsstyring af delte data

Som led i understøttelsen af rullende opgraderinger, påhæftes alle delte datatyper, dvs id-kort og dynamisk konfiguration, et versionsnummer. Software-opgraderinger, der involverer modifikationer af de delte datatyper, må understøtte både det nye og det gamle format, og fortsætte med at udgive data i det forrige format indtil det detekteres at der ikke længere er cluster-medlemmer, der kræver dette.

Der understøttes kun det aktuelle og det forrige format. Ved en forskel på mere end ét versionsnummer af beskedformatet, kan der ikke kommunikeres. Ved opstart undersøger en server om beskeder afsendt af andre servere er i et af de to understøttede formater. Hvis dette ikke er tilfældet afbrydes opstarten.

Konfiguration

Den statiske, filbaserede konfiguration af SOSI-GW applikationen holdes på et minimum for at undgå omstændig kopiering af konfiguration, og for lettere at kunne honorere kravet om rullende opdateringer mm. Al anden konfiguration lægges i den lokale database på de enkelte servere og udveksles via multicast.

Den dynamiske konfiguration persisteres i den lokale database på alle servere i SOSI-GW clusteret. Konfigurationsdata gemmes altid som et komplet hele, der er forsynet med et tidsstempel, der identificerer og sammenbinder sammenhørende konfigurationsdata. Det er ikke påtænkt, at administrationskonsollen skal betjene mere end én samtidig bruger, men der foretages ikke eksplicit sikring af, at dette overholdes. Derimod håndteres eventuelle konflikter ved at lade den yngste konfiguration vinde. Den statiske konfiguration replikeres ikke automatisk. I stedet bør konfigurationsfilen synkroniseres via andre mekanismer, f.eks. rsync.

En vilkårlig server i clusteret kan servicere administrationskonsollen. Når administratoren markerer at konfigurationsændringer skal gemmes, distribueres de nye konfigurationsdata, og alle servere persisterer konfigurationen i den lokale database. Når en server startes, sammenlignes tidsstemplet på de persisterede konfigurationsdata med den version, der kan aflæses via netværket, og den nyeste af de to benyttes.

En konsekvens af den anvendte strategi vil være, at det er ukompliceret at pakke løsningen som f.eks. et VMware image, hvilket letter opsætningen af yderligere SOSI-GW servere.

Administrationskonsol

Administration af et SOSI-GW cluster foretages via en webbaseret administrationskonsol. Grænsefladen er baseret på GWT (Google Web Toolkit). Administrationskonsollen indeholder flg. funktioner:

- Bootstrap mode, hvor gateway funktionalitet er slået fra, men hvor adgangsbegrænsning er baseret på filbaseret brugernavn/password. Denne tilstand benyttes kun under installationsprocessen.
- Opsætning af whitelists, hvor tilladte klienter sættes op og identificeres.
- Opsætning af service positivliste, hvor services, som må kaldes på vegne af klienter, sættes op og identificeres.
- Id-kort revokering, hvilket giver mulighed for at fremsøge et id-kort på baggrund af et bruger id, og efterfølgende revokere dette, dvs. fjerne id-kortet fra den delte cache.
- Søgning i audit logs. Administrationskonsollen giver mulighed for at foretage simple søgninger i auditloggen, f.eks. baseret på bruger-id og/eller tidsrum.

Logning

Auditlogning foretages lokalt på de enkelte servere i clusteret. Persistering af log-elementer foretages i en asynkron afkoblingstråd, således at gateway funktionaliteten kan afvikles optimalt. Auditlogs persisteres lokalt i de decentrale databaser, der er installeret på alle SOSI-GW servere.

Med et konfigurerbart interval sender hver enkelt server opsamlede logs til den centrale log-merger database server. Dette foregår via JDBC adgang til log-mergerens database. Når data er committet til log-merger databasen, slettes disse lokalt. Dette betyder, at selvom log-mergeren ikke er dubleret, mistes der ikke auditlogs hvis log-mergeren er ude af drift. I det øjeblik log-mergeren er reetableret, vil akkumulerede auditlogs automatisk blive overført. De data, som er overført til log-mergeren, bør sikres med passende diskspejlings- og backup-strategier.

Søgning i auditlogs foregår på log-mergerens database via administrationskonsollen, eller alternativt via et generisk SQL værktøj (PgAdmin, DbVisualizer el. lign.) direkte i databasen.

I en minimal ikke-redundant konfiguration kan log-merger og SOSI-GW server køre på den samme host, (evt. under samme DBMS).

Brugen af en central log mens der logges direkte til lokale logs gør, at den centrale log kan undværes i kortere eller længere perioder. Dermed er der ikke nogen faste afhængigheder mellem serverne, og SOSI-GW går ikke ned hvis den centrale log forsvinder. Konsekvensen er, at den centrale log ikke viser realtidsoplysninger, da der kan gå et stykke tid fra en hændelse sker til den optræder i den centrale log. Det betyder også, at skulle en af serverne stoppe helt med at fungere, så kan logbeskeder gå tabt. Det anbefales dog, at der også på de enkelte servere køres med spejlede diske. Derfor skal det gå meget galt før noget går helt tabt.

Ud over den beskrevne logning, som fungerer som audit-logning, bliver der også internt foretaget logning via log4j. Denne logning sker lokalt på de enkelte maskiner, og er beregnet til at overvåge og debugge systemet.

Til auditloggen logges følgende:

Proxyrequests Tidspunkt, NameID, Systemid, IP for afsender, endpoint, IDCardID (kortets) unikke id

Service requests Tidspunkt, NameID, Systemid, IP for afsender, operation, status (OK/ERR)

URL requests Tidspunkt, NameID, Systemid, IP for afsender, status (OK/ERR)

Desuden logges alle ugyldige requests, både Webservicerequests og requests, der stammer fra den browserbaserede signering.

Samtidighed

Metoderne til at oprette, signere og fjerne id-kort implementerer ikke nogen form for låsning. Det betyder, at hvis to klienter på samme tid bestiller et nyt id-kort for den samme bruger, så vil kun en af dem kunne signere det endelige id-kort, nemlig den klient, der bestilte sidst. Det antages at der ikke foretages samtidige kald fra den samme bruger, og at problemet derfor ikke er relevant.

Load balancing

For at distribuere load til de servere, der indgår i et SOSI-GW cluster, er der behov for at opsætte en load balancing mekanisme. Dette er ikke en del af selve SOSI-GW, men kunne f.eks. bestå i en af følgende:

- DNS round-robin
- TCP load balancer (f.eks. Linux Virtual Server)
- HTTP load balancer

På NSP platformen anvendes Riverbed Stingray til load balancing.

Der er ikke behov for understøttelse af session affinity el. lign., da alle servere har adgang til alle cachede id-kort via det distribuerede netværk. Dette gælder for alle funktioner, også administrationskonsol og signeringsapplet.

Der vil være behov for at enten den anvendte load balancer eller (i tilfældet DNS round-robin) klienterne forsøger igen i tilfælde af at en server ikke kan nås.

I tilfælde, hvor det ønskes at kommunikation imellem klient og SOSI-GW foregår over SSL (f.eks. den genererede URL til signeringsapplet), skal der enten opsættes en SSL terminering foran SOSI-GW serverne eller også skal det være muligt for de individuelle SOSI-GW servere at kunne uddele en URL, der rammer en specifik server.

Afvikling på en enkelt maskine

Det er muligt at afvikle SOSI-GW på en enkelt maskine uden ændringer, da systemet selv vil detektere at der kun er ét medlem af clusteret. Det vil også være muligt at fravælge distribuerings-funktionaliteten via den statiske konfiguration, da det i dette tilfælde vil påføre et unødvendigt (omend minimalt) overhead. Det vil fortsat være muligt at køre log-mergerens database på samme eller en anden maskine.

Anvendte teknologier og komponenter

I SOSI-GW er følgende teknologier og komponenter anvendt:

- SOSI Seal til udstedelse af ID-kort
- MySQL database server til opbevaring af konfiguration og auditlogs
- JAX-WS-RI til implementering af web services
- Google Web Toolkit til implementering af administrationskonsol
- OpenSign applet til signering af ID-kort via browser
- Apache JMeter til performancetest

Webservice API

SOSI-GW's primære funktionalitet stilles til rådighed gennem Webservices. Funktionaliteten er delt op i to dele: Services til at håndtere id-kort (oprette, signere og nedlægge) og en proxyservice.
Proxyservicen er et generisk endpoint, der står for at videresende requests fra en klient til et andet endpoint mens der også tilføjes et signeret id-kort. Når proxyservicen anvendes, foretages der ikke nogen validering af SOAPbody, og ligeledes checkes headeren kun for felter, der er nødvendige for at SOSI-GW kan fungere.

Alle services bygger på den allerede definerede form for id-kort fra DGWS, hvilket betyder, at data sendes som SAMLassertions. Ved samtlige kald til SOSI-GW, checkes om kalderen, identificeret ved en IP-adresse og en shared secret, har adgang til at benytte SOSI-GW. Der benyttes altså kun standarder, der allerede er i anvendelse (WSAddressing, SAML m.v.).

Oversigt over metoder tilgængelige som webservices

Følgende metoder er tilgængelige via webservices:

- proxy
- requestIdCardDigestForSigning
- signIdCard
- getValidIdCard
- logout

Proxyservice

Proxy tager et generisk SOAPrequest og tilføjer id-kort til headerne. For at gøre dette, skal klienten medsende sessionsidentifikation i form af et brugerid. Brugerid'et transporteres i en SAMLassertion i stil med DGWS' idkort, hvor det eneste benyttede felt er <NameID>. Når requestet er modtaget, og brugerens id-kort er vedhæftet, sendes requestet direkte videre til det endelige endpoint. SOSI-GW understøtter opgradering af niveau 1 til 4. Niveau 1 er defineret i DGWS, og betyder at der kommer et usigneret id-kort med brugernavn. Til requests, hvor der allerede eksisterer et signeret id-kort, ignoreres alt andet indhold end <NameID> feltet, og kortet erstattes med det, der opbevares af serveren. I de tilfælde, hvor der ikke eksisterer et kort i forvejen, fejler operationen.

Servicen tager ikke argumenter direkte, da SOAPbody indeholder det, der skal sendes videre til den endelige aftager. Til gengæld kræves at requests er udstyret med en header, der indeholder følgende:

- Unikt brugerid i form af en SAMLassertion med <saml:NameID>
- WSAddressing headers i henhold til http://www.w3.org/Submission/wsaddressing/
- Et optionelt id-kort. Dette fjernes inden dispatch og erstattes evt. af det cachede kort.

Returværdier:

- "200" (OK) på HTTP niveau, og som indhold det uændrede svar fra den ønskede service.
- "500" (Error) på HTTP niveau, hvilket kan skyldes:
- Den ønskede service returnerede denne fejl.
  - Den ønskede service er ikke konfigureret på listen over kendte endpoints.
  - Manglende eller ugyldige argumenter i SOAPheaderne, f.eks. manglende WSaddressing.
  - Der findes ikke et gyldigt signeret Id-kort i SOSI-GW.

I fejltilfældene sendes en beskrivelse af fejlen i <fault> elementet, dels som kodede værdier, og dels som læselig tekst i henhold til DGWS.

Hvordan NameID tilføjes til SOAPbeskeden er op til klienten selv, men det vil formentlig være smartest at skubbe ansvaret for det ned i den webservicestak, der anvendes. Det vil kunne holde selve klient-API'et fri for at skulle håndtere SOAPheaders, og i stedet lade stakken håndtere det internt. Det samme vil gøre sig gældende i forhold til WSAddressing headers. Disse er ikke en del af standardprofilerne, og skal ligeledes tilføjes eksplicit.

WSAddressing er nødvendigt for at identificere det endelige endpoint for den forespurgte service. Nogle services understøtter dog ikke WSAddressing endnu, og da nogle webservicestakke sætter mustUnderstand="1" på WSAddressing headers, vil disse kald umiddelbart fejle når de når til endpointet. Derfor understøtter SOSI-GW en filtreringsmekanisme, så WSAddressing headers automatisk kan fjernes inden et request sendes videre. Hvorvidt headers skal fjernes konfigureres pr. endpoint. WSAddressing understøttes i øvrigt kun til at finde endpoint – det forudsættes at svaret sendes tilbage på samme kanal som requestet kom. SOSI-GW understøtter derfor ikke at der svares på en ny service, som klienten selv udstiller. Endelig understøtter SOSI-GW heller ikke, at svar kommer tilbage fra endpoint på andre kanaler.

Håndtering af implicit Id-kort oprettelse ved proxy-requests

Hvis der er et tilstrækkeligt udfyldt id-kort i et proxy-request, mens der ikke findes et signeret id-kort i SOSI-GW, bliver der oprettet et nyt usigneret id-kort. Fra det fremsendte id-kort benyttes data fra afsnittene "Obligatoriske system og organisationsoplysninger" og "Eventuelle brugeroplysninger", jævnfør "Den Gode Webservice". Digest af id-kortet samt signingurl sendes retur til klienten i en ny SOAPheader. Det er op til klienten at inspicere fejlbeskeder og headers for at afgøre om den skal præsentere brugeren for en signeringsdialog, enten indbygget i klienten, eller gennem start af en browser mod den returnerede url.

Det skal bemærkes, at denne implicitte oprettelse af id-kort har den konsekvens, at den etablerede kontrakt mellem klienten og den oprindelige serviceudbyder bliver brudt, da der pludselig kan opstå

tilfælde, som ikke var planlagt oprindeligt. Konkret betyder dette, at en klient genereret ud fra en WSDL-fil ikke vil være forberedt på alle fejlmulighederne, og at de nye situationer derfor bliver svære at håndtere.

Iflg. DGWS er de eneste tilladte returkoder fra en service 200 (OK) eller 500 (fejl). I det aktuelle tilfælde med implicit login, skal der hæftes noget mere på svarene for at skelne mellem en "normal" servicefejl og det tilfælde, hvor det ikke er muligt at kalde servicen pga. manglende id-kort. Især ved autogenererede klientstubbe bliver dette problematisk, da de genererede klasser ikke er beregnet på at håndtere disse tilfælde.

Endelig har implicit oprettelse den svaghed, at hvis der ikke er oprettet et id-kort inden et request, så skal det fulde request alligevel sendes til SOSI-GW, som straks derefter vil afvise det. Især ved store requests, f.eks. hvis der sendes billeder, vil dette være meget ineffektivt.

requestIdCardDigestForSigning

Denne benyttes til eksplicit oprettelse af et id-kort, klar til signering. Servicen kaldes af en klient for at etablere en session for en bruger, så der kan sendes signerede requests til andre services.

Servicen kræver et partielt id-kort, som SOSI-GW skal opbevare i signeret tilstand. I SOAPbody sendes derfor en SAMLassertion med samme indhold som for implicit oprettelse. Servicen returnerer den digest-værdi som klienten skal signere sammen med en URL, der kan anvendes af en browser. Der skelnes ikke mellem URLrequests og WSrequests.

Usignerede id-kort fjernes automatisk fra serverne hvis der ikke er modtaget en signeret digest inden et vist interval. Dette interval er konfigurerbart.

signIdCard

Denne metode benyttes til at tilføje et signeret digest til det cachede idkort i SOSI-GW. Argumenter til dette kald er NameID, SignatureValue og brugerens offentlige nøgle. SOSI-GW lader herefter STS'en signere id-kortet, checker at signaturen er gyldig, og gemmer det i sin cache.

getValidIdCard

Denne metode henter id-kortet for en specifik bruger. Argumentet er NameID for brugeren, og returværdien er brugerens id-kort eller en fault.

Metoden er et supplement til de metoder, der er angivet i kravspecifikationen, men er nødvendig for at håndtere to tilfælde pænt:

- Når et id-kort er ved at blive signeret af en bruger gennem browserbaseret signering, er der ikke noget feedback, der fortæller klienten hvornår brugeren er færdig med signeringen. Klienten har derfor to muligheder: Poll serveren periodisk via denne metode eller lad brugeren manuelt indikere at processen er færdig.

  Metoden erstatter [Krav 10a], som havde samme effekt, bare med et blokerende kald til serveren.

Det blokerende kald har den ulempe, at det optager serverressourcer i længere tid, og kan derfor være en hindring for høj skalerbarhed. Til gengæld bliver en del af ansvaret flyttet til klienten, der aktivt skal checke for id-kort, men dette er vurderet som den bedre løsning.

- Det andet tilfælde er det, hvor en webservice kræver et andet timeoutniveau (i DGWS termer). Nogle services kræver f.eks. at id-kort er underskrevet for hvert request, og denne situation skal håndteres i klienten. Klienten er derfor nødt til at kunne afgøre, om id-kortet er for gammelt til at kunne bruges. Dette kan afgøres ved at benytte getValidIdCard, der returnerer brugerens id-kort.

Et tredje tilfælde er det, hvor en klient selv står for at kontakte en service, men ikke vil håndtere opbevaringen af id-kort og kommunikationen med STS. Her kan klienten få det underskrevne id-kort fra SOSI-GW og selv vedhæfte det i en besked, der skal sendes.

Hvis der ikke eksisterer noget id-kort for brugeren, returneres der en fault. Denne fault angiver status for brugerens id-kort: Enten kan en bruger være i gang med signering, og der kan derfor med fordel prøves igen lidt senere, eller også er der slet ikke oprettet noget id-kort, og det giver derfor ikke mening at prøve igen, før et nyt id-kort er bestilt.

Metoden kan i øvrigt konfigureres fra serveren, så selve id-kortets signerede digest ikke sendes med ud, da dette i visse tilfælde kan betragtes som et brud på sikkerheden.

logout

Denne metode fjerner det signerede id-kort for en bestemt bruger. Som argument kræves NameID.

Tilgang til services til at håndtere ID-Kort

Der er udstillet to snitflader til at tilgå id-kort, begge som webservices. Den ene service udstiller den fulde funktionalitet, og kan passende beskyttes, så den kun kan tilgås fra bestemte fagsystemer.

Den anden webservice udstiller kun håndtagene requestIdCardForSigning og SignIDCard. Denne webservice er lavet, så disse håndtag kan udstilles til at bredere publikum, eksempelvis gennem SignOn Biblioteket.

HTTP API

En del af SOSI-GW's funktionalitet sker gennem et normalt HTTP API for at understøtte, at almindelige browsere kan komme i kontakt med systemet for at signere digests. Dette sker ved at klienten får svar på et request om at oprette et nyt id-kort, enten eksplicit eller implicit, og bruger den url, der kommer ud af det, til at sende til en browser. For at understøtte proxy/nat mellem browserne og SOSI-GW, er det muligt at konfigurere host-adressen for SOSI-GW og dermed gøre det muligt at sende requests gennem en proxy på det lokale netværk.

Browseren åbner adressen og brugeren bliver præsenteret for en webside med en applet, der kan anvendes til at signere digestværdien fra det nye id-kort. Adressen er en engangsadresse, der er identificeret ved en lang tilfældigt genereret tekststreng, og som kun er gyldig i max. 30 sekunder. Serveren genererer herefter en ny URL, som bruges af appletten til at sende svaret. Denne URL er gyldig i 60 sekunder, hvilket betyder at brugeren skal have signeret inden dette tidsrum. Sker det ikke, vil der fremkomme en fejlmeddelelse når brugeren prøver at signere. På websiden med appletten vil tiden være angivet. Gyldighedsperioderne kan konfigureres i administrationsinterfacet.

Den URLbaserede signering baseres på OpenSign, der er en Java-applet, som implementerer browser-baseret signering. Resultatet af denne applet er en XML-repræsentation af signaturen, struktureret i forhold til W3C's XML Signature skema.