

**DIGITALISERINGS
KATALOGET**

KOM GODT I GANG

WEBSERVICE

En trin for trin guide til dig, der skal
anvende en webservice for første gang

November 2020

KOMB:T

Kommunernes it-fællesskab

1 Introduktion

Denne guide henvender sig til dig, der for første gang skal integrere med en webservice i den fælleskommunale infrastruktur, der anvender token-baseret sikkerhed. Guiden arbejder med integrationen SF1520 CPR replika opslag, da denne integration er simpel og er anvendt af mange. Samme fremgangsmåde og principper gør sig gældende for de øvrige webservice integrationer.

Indledningsvis beskrives et praktisk eksempel på ibrugtagning af en webservice med udgangspunkt i .NET client for Serviceplatformens DemoService. Dernæst følger et eksempel hvor samme trin udføres med Java-versionen. I appendiks beskrives de grundlæggende teknologier og principper for integration mod en webservice.

2 Baggrundsdocumentation

Information om den anvendte sikkerhedsmodel finder du i *Programmers Guide til Sikkerhed i den Fælleskommunale Infrastruktur*, der henvises til på [\[INTRO\]](#). Den essentielle information for den specifikke integration du skal benytte findes i digitaliseringskatalogets oversigt af udstillede [integrationer](#). Her eksempel med *CPR replika opslag* som vi skal integrere med.



The screenshot shows a service card with the following details:

- Service ID: SF1520
- Category: Person
- Service Name: CPR replika opslag (SKI 02.19)
- Version: 3.6
- Status: I DRIFT (in a green box)
- Description: Integrationen giver kommunernes fagsystemer mulighed for at fremsøge de personoplysninger, der er nødvendige for den kommunale sagsbehandling og forvaltningsvirksomhed.
- Additional Info: 1 version i alt
- Action: [Læs mere](#)

Herfra vælger du "Læs mere" og dernæst vises informationssiden for den pågældende integration. Her henter du dokumentationspakken for integrationen, der indeholder den essentielle information og tekniske bilag, som illustreret senere i denne guide.

Referencer:

[INTRO]	Webservice Introduktionsside
[KODE]	Serviceplatformens GitHub repository
[KGIG-VEJL]	Kom-godt-i-gang vejledninger
[RETL]	Retningslinjer for anvendelse af det eksterne testmiljø
[ADMINTRO]	Fælleskommunalt Administrationsmodul - Introduktionsside
[ADMGUI]	Fælleskommunalt Administrationsmodul - Brugerside

3 Forudsætninger

Hvis du ikke allerede er oprettet som leverandør og har et anvendelsesystem registreret, følg da vejledninger der henvises til i [ADMINTRO] - *Brugervejledning til Administrationsmodulerne for leverandører*. Her beskrives hvordan du oprettes som organisation (leverandør) og hvordan du opretter dit IT-system. For IT-systemer der tilgår webservices benyttes betegnelsen *Anvendelsesystem*. For at kunne registrere et Anvendelsesystem skal du have et funktionscertifikat, hvilket er beskrevet i [[KGIG-VEJL](#)] - *certifikater*.

De grundlæggende forudsætninger er:

For at kunne tilgå [ADMGUI] skal din medarbejdersignatur have tildelt følgende NemLogin-rolle:

- "KOMBIT STS Administrationsmodul (test) Leverandøradministrator"

Inden du kan starte skal du:

- Have Anvendelsesystem oprettet i [ADMGUI]
- Have den offentlige version af dit funktionscertifikat registreret på Anvendelsesystemet
- Have den private version af dit funktionscertifikat klar samt adgangskoden til denne
- Aftale med en kommune om at kunne teste dit fagsystem med dem som myndighed

4 Kald CPR webservice - .NET

4.1 Hent dokumentationspakken

Der følger en dokumentationspakke til alle integrationer og disse hentes fra [Digitaliseringskataloget](#). Til dette eksempel hentes dokumentationspakken for [SF1520](#). Du vælger blot "Download dokumentation".

SF1520_3.6 **CPR replika opslag** Status gyldig fra 10/01/2019 Version 3.6 I DRIFT

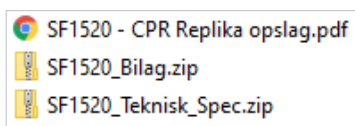
Beskrivelse

Integrationen indeholder en service, der kan bruges til at fremsøge personer ud fra udvalgte søgekriterier, samt en service, der kan bruges til opslag af detaljerede personoplysninger på en enkeltperson ud fra CPR-nummer. Den adskiller sig fra de øvrige integrationer om CPR-oplysninger ved, at der kan forespørges på andre parametre end personnummer. Den returnerer yderligere særskilt outputfelter vedr. CPR Vejregister.

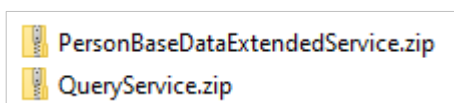
Serviceplatformen vedligeholder lokalt en komplet kopi af CPR-registret, CPR Vejregister og CPR's myndighedsregister. Kopien hedder CPR-replika. Dagligt og periodisk modtager Serviceplatformen ændringsudtræk fra CPR, som synkroniseres ind i CPR-replikaet. Derudover bliver CPR-replikaet også dagligt beriget med data om en borgers mulighed for at benytte NemSMS/Digital Post.

 Download dokumentation

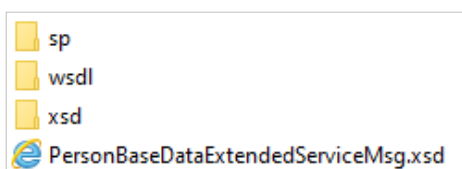
Dokumentationen består af en zip-fil, der oftest indeholder en integrationsbeskrivelse (pdf), en pakket fil med bilag samt en pakket fil med tekniske specifikationer.



Integrationsbeskrivelsen indeholder detaljeret information om integrationen og skal altid læses, inden den tages i anvendelse. Hvis integrationen er en webservice, indeholder den pakkede fil tekniske specifikationer, WSDL samt schema-filer. I dette tilfælde *SF1520_Teknisk_Spec.zip* der indeholder:



Udpak *PersonBaseDataExtendedService.zip*. Strukturen i denne er:



Filen "sp/service.properties" indeholder parameter *service.entityID*, som skal anvendes, når der anmodes om et Security Token. Alternativt kan man også finde denne parameter på serviceaftaler i [ADMGUI] ved at klikke på informations-ikonet for en service (det er beskrevet senere i denne guide).

Du skal anvende den WSDL, der findes i folderen "/wsdl/token/".

4.2 Opret serviceaftale

Du skal have en gyldig serviceaftale, så du kan få udstedt et Security Token. For at kunne anmode om en aftale, skal du være i dialog med en myndighed om at kunne teste på deres vegne, da de skal godkende aftalen. Klik på "+" udfør den service du skal anvende for at se detaljer:

Services			
Webservice / Serviceplatformen CPRQueryService	Version 2.0	I DRIFT	+
Webservice / Serviceplatformen PersonBaseDataExtendedService	Version 4.0	I DRIFT	+



Her finder du adressen til endpoint der skal kaldes samt UUID for den service du skal anmode om aftale på. Notér disse i et arbejdsdokument:

Eksternt testmiljø

Endpoint: <https://exttest.serviceplatformen.dk/service/CPR/PersonBaseDataExtended/4>

UUID: [e6be2436-bf35-4df2-83fe-925142825dc2](#)

Fejlkode: [Link til fejlkode](#)

Du kan benytte service UUID når du skal fremfinde og vælge service ved anmodning om serviceaftale.

I nedenstående eksempel er der oprettet en serviceaftale for myndigheden KOMBIT A/S for anvendelsesystemet *KDI CTT Test System #2*. Det giver vores test-system lov til at kalde på vegne af KOMBIT. Serviceaftalen dækker opslag på CPR-nummer samt søgninger i CPR-data, hvilket er to separate services. Dette er således et eksempel på en integration (SF1520) der udstiller to services. Hvis man skal anvende begge giver det mening at have dem i samme serviceaftale, da de er tæt relateret. For at gennemføre denne øvelse behøver du blot oprette en serviceaftale på *Person stamdata udvidet (lokal) - v4.0*.

System:	KDI CTT Test System #2	Oprettet:										
Begrundelse:	Til test											
Betingelser:	Vis vilkår og betingelser ved anmodning	Ændret:										
Datamodtager:	KOMBIT A/S											
Kommentar:	Til testformål											
Betingelser:	Vis vilkår og betingelser ved godkendelse											
Godkendt:	2020-02-26 14:57:29 af Mads Gunne, KOMBIT A/S											
Services:	<table><tr><td>Navn:</td><td>Query - v2.0</td></tr><tr><td>Type:</td><td>Fælleskommunal Service</td></tr><tr><td>Udbyder:</td><td>Serviceplatformen</td></tr><tr><td>Dataafgrænsning:</td><td>dummy</td></tr><tr><td>Yderligere information:</td><td></td></tr></table>		Navn:	Query - v2.0	Type:	Fælleskommunal Service	Udbyder:	Serviceplatformen	Dataafgrænsning:	dummy	Yderligere information:	
Navn:	Query - v2.0											
Type:	Fælleskommunal Service											
Udbyder:	Serviceplatformen											
Dataafgrænsning:	dummy											
Yderligere information:												
	<table><tr><td>Navn:</td><td>Person stamdata, udvidet (lokal) - v4.0</td></tr><tr><td>Type:</td><td>Fælleskommunal Service</td></tr><tr><td>Udbyder:</td><td>Serviceplatformen</td></tr><tr><td>Dataafgrænsning:</td><td>dummy</td></tr><tr><td>Yderligere information:</td><td></td></tr></table>		Navn:	Person stamdata, udvidet (lokal) - v4.0	Type:	Fælleskommunal Service	Udbyder:	Serviceplatformen	Dataafgrænsning:	dummy	Yderligere information:	
Navn:	Person stamdata, udvidet (lokal) - v4.0											
Type:	Fælleskommunal Service											
Udbyder:	Serviceplatformen											
Dataafgrænsning:	dummy											
Yderligere information:												



Klik på Person stamdata informationsikonet ”Yderligere information” for at se *Entity ID* og kopier værdien. Denne værdi skal bruges i forespørgsel på udstedelse af token, der giver adgang til service:

Yderligere information	
UUID:	e6be2436-bf35-4df2-83fe-925142825dc2
EntityId:	http://cpr.serviceplatformen.dk/service/personbasedataextended/4
Navn:	Person stamdata, udvidet (lokal) - v4.0

Service Entity ID (nøgle der identificerer service, der anmodes om token til):

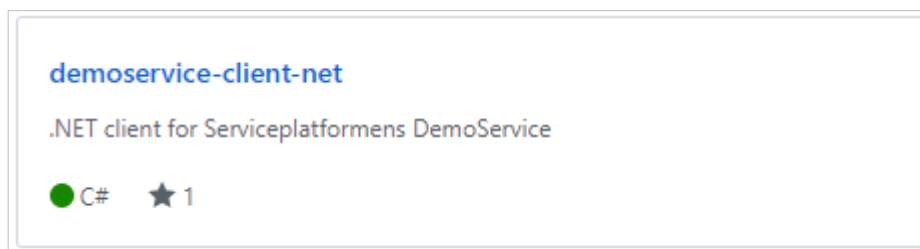
<http://cpr.serviceplatformen.dk/service/personbasedataextended/4>

4.3 Registrér funktionscertifikat

Du skal registrere den private version af dit funktionscertifikat på maskinen du skal afvikle koden fra. Dette er beskrevet i [KGIG-VEJL] - *certifikater* afsnit *Windows Certificate Store*. Du bør læse hele guiden, så du har den fornødne kendskab til anvendelse og konfiguration af certifikater. Inklusivt afsnit om Certification Authorities (CA), da vi ofte ser i praksis at Trust Store på maskine som koden afvikles på ikke er helt opdateret og kan mangle fornødne CA.

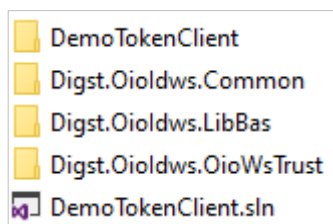
4.4 Hent demoservice-klienten

Tilslutning til Serviceplatformen kræver nøje forståelse for sikkerhedsmodellen og indlejring af token fra Security Token-servicen i det efterfølgende kald til Serviceplatformen, som beskrevet i *Programmernes Guide til Sikkerhed i den Fælleskommunale Infrastruktur*, der henvises til på [INTRO]. Dette er illustreret i disse kodeeksempler, som du skal hente fra [Github](#). Du skal hente *demoservice-client-net*:





Du skal anvende solution DemoTokenClient.sln, der er baseret på token sikkerhed, som ligger i folder \demoservice-client-net-master\DemoTokenClient:



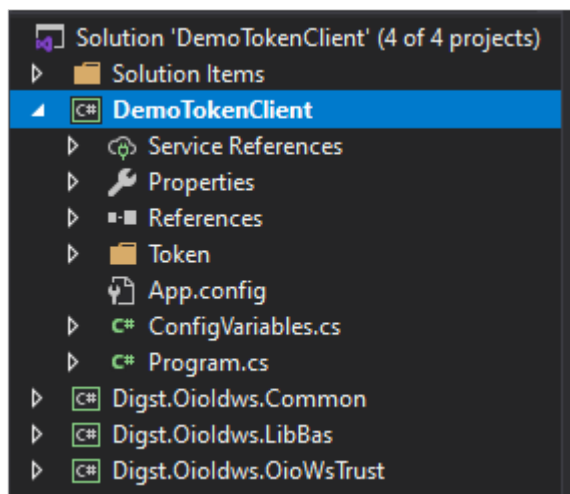
De tekniske forudsætninger for at kunne bygge og køre koden er beskrevet på [eksempelkodens introduktionsside](#).

Til denne guide har vi anvendt [Visual Studio Community](#), som indeholder alle forudsætninger for at kunne arbejde videre på koden. Vi antager, at du har et grundlæggende kendskab til Visual Studio og C# i det følgende. Du kan med fordel læse [readme.md](#) i solution, inden du starter.

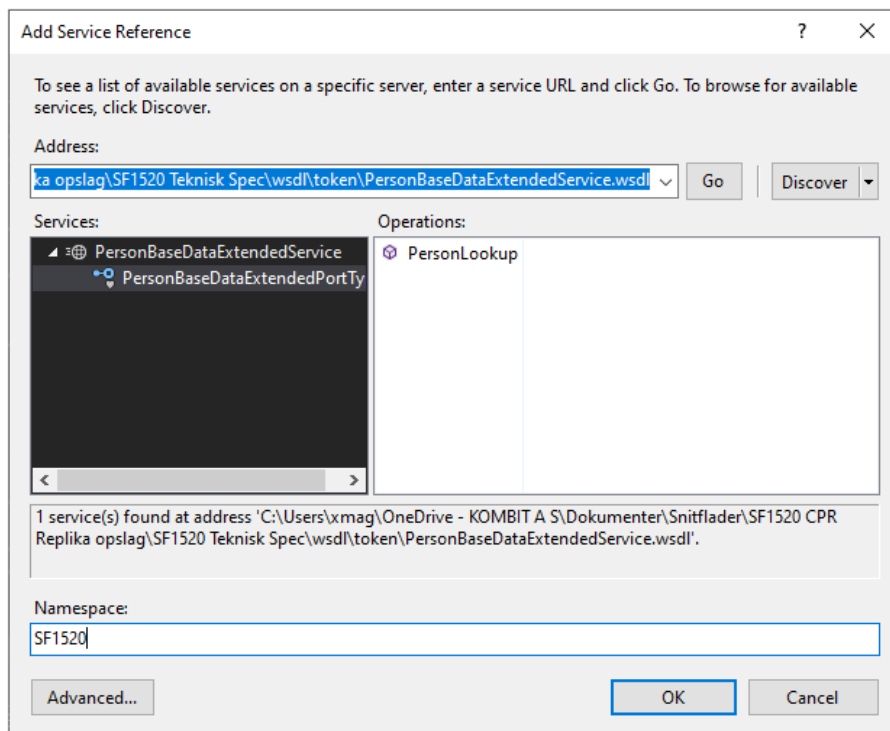
4.5 Byg Solution

Før du ændrer i koden, skal du køre et solution build, så referencer til Digitaliseringsstyrelsens OIOIDWS-komponenter fungerer. Ellers vil fx opdateringer til App.config ikke ske automatisk ved tilføjelse af ny service reference.

4.6 Tilføj reference til service



Højreklik på "Service References" eller "Connected Services" (afhængigt af version) og vælg den lokale WSDL fra dokumentationspakken. Angiv "SF1520" som namespace.



4.7 Opdatér kodens konfiguration

I App.config skal du opdatere endpoint konfigurationen for SF1520. Kopier værdier for "binding" samt "bindingConfiguration" fra demosevice endpoint. Tilføj attribut "behaviourConfiguration" fra demosevice endpoint. Endpoint adresse finder du i Digitaliseringskataloget under detaljer for servicen:

Eksternt testmiljø

Endpoint: <https://exttest.serviceplatformen.dk/service/CPR/PersonBaseDataExtended/4>

<https://exttest.serviceplatformen.dk:443/service/CPR/PersonBaseDataExtended/4>

Opdateret endpoint konfiguration::

```
<endpoint address="https://exttest.serviceplatformen.dk/service/CPR/PersonBaseDataExtended/4"
  behaviorConfiguration="LibBasBehaviourConfiguration" binding="LibBasBinding"
  bindingConfiguration="LibBasBindingConfiguration"
  contract="SF1520.PersonBaseDataExtendedPortType" name="PersonBaseDataExtendedPort" />
```

I ConfigVariables.cs skal du opdatere følgende parametre:

- ServiceEntityId
- ClientCertificateThumbprint
- ClientCertificateStoreLocation (hvis nødvendigt)



- ClientCertificateStoreName (hvis nødvendigt)
- Cvr (som serviceaftalen dækker)

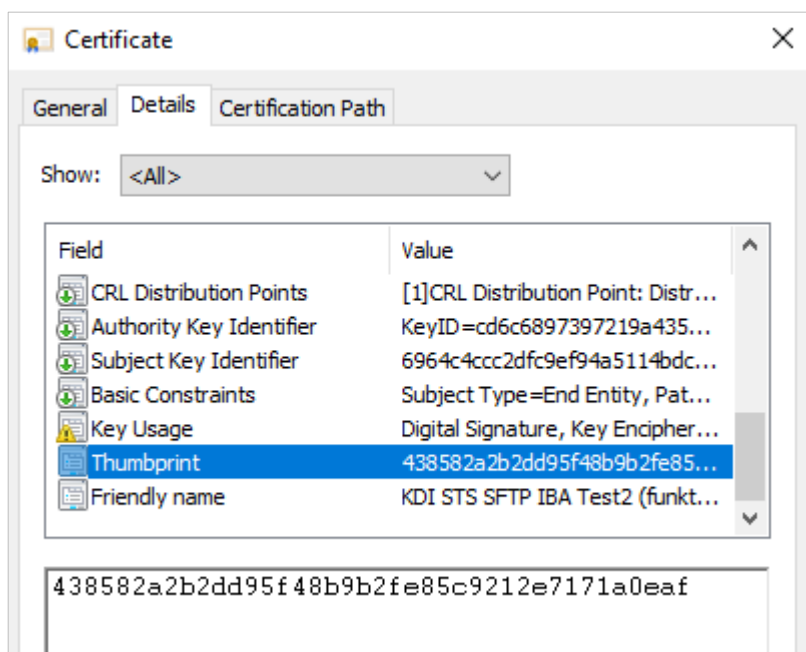
```
static class ConfigVariables
{
    // The alias used for Serviceplatformen endpoint identity check.
    public const string ServiceCertificateAlias = "kombit-sp-signing-test (funktionscertifikat)";

    // SHA-1 thumbprint of the client certificate to call STS and Serviceplatformen.
    public const string ClientCertificateThumbprint = "9B 6B 8E 98 30 9E 53 89 24 35 BA AB 1D 0B 86 C4 F3 38 A1 DF";

    public const StoreLocation ClientCertificateStoreLocation = StoreLocation.CurrentUser;

    public const StoreName ClientCertificateStoreName = StoreName.My;
}
```

Du finder Thumbprint for dit certifikat ved at åbne det og gå nederst på listen under "Details".



4.8 Opret en ny klasse SF1520Client.cs

Tilføj en ny SF1520Client.cs fil til projektet og kopier indholdet fra DemoServiceToken.cs. Dernæst skift reference til SF1520 og navnet på klassen til "SF1520Client".

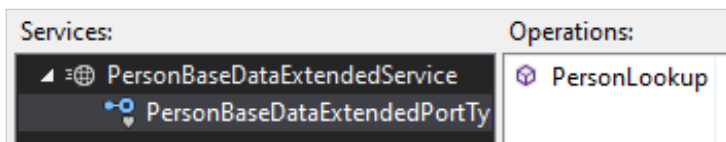


```
SF1520Client.cs* -> X DemoServiceToken.cs Program.cs App.config
DemoTokenClient
1 using System;
2 using System.IdentityModel.Tokens;
3 using System.Net;
4 using System.Net.Security;
5 using System.ServiceModel;
6 using DemoTokenClient.SF1520;
7
8 namespace DemoTokenClient.Token
9 {
10     Oreferences
11     public class SF1520Client
12     {
13         Oreferences
14         public string CallDemoServiceWithToken(string mess
15         {
16             var token = TokenFetcher.IssueToken(ConfigVar3
```

Fordi vi har skiftet reference til SF1520 service, vil IntelliSense markere, hvor vi skal opdatere:

```
var token = TokenFetcher.IssueToken(ConfigVariables.ServiceEntityId);
callDemoServiceRequest request = new callDemoServiceRequest
```

Du skal herefter erstatte "callDemoService" med "PersonLookup" samt "Demo" med "PersonBaseDataExtended" i variabel-deklarationer:



- callDemoServiceRequest -> PersonLookupRequest
- DemoPortType -> PersonBaseDataExtendedPortType
- callDemoServiceResponse -> PersonLookupResponse
- DemoPortTypeClient -> PersonBaseDataExtendedPortTypeClient



4.9 Tilføj funktion til serialisering

Tilføj følgende funktion til klassen, så vi kan returnere svaret som streng:

```
using System.Xml.Serialization;
using System.IO;

...

private static string SerializeObj<T>(T obj) {
    try {
        XmlSerializer xmlSerializer = new XmlSerializer(obj.GetType());
        using (StringWriter txtWriter = new StringWriter()) {
            xmlSerializer.Serialize(txtWriter, obj);
            return txtWriter.ToString();
        }
    }
    catch (Exception ex) {
        return "Unable to serialize: " + ex.Message;
    }
}
```

Og skift retur-værdien til streng-version af response object:

```
return SerializeObj(response.PersonLookupResponse1);
```

4.10 Opdatér program.cs

Vi mangler herefter blot at ændre startkoden i Program.cs til at kalde SF1520-klienten i stedet:

```
string endpointUrl = "https://exttest.serviceplatformen.dk/service/CPR/PersonBaseDataExtended/4";
string message = "2905690000";

Console.WriteLine("\nCalling SF1520 with token...");
string demoServiceResponse = new SF1520Client().CallDemoServiceWithToken(message, endpointUrl);
Console.WriteLine("Response from DemoService:\n");
Console.WriteLine(demoServiceResponse);

Console.WriteLine("\nPress any key to exit");
```

Som input kan du bruge et vilkårligt CPR-nummer fra listen af [CPR Arketyper](#) (se regnearket). Hvis din serviceaftale er gyldig, dit certifikat er validt og korrekt angivet i ADM, og alle konfigurationsparametre er opdateret korrekt, så ser svaret sådan ud, når programmet afvikles:



```
C:\Users\xmag\OneDrive - KOMBIT A S\Dokumenter\Demoklient\DemoTokenClient\DemoTokenClient\bin\Debug\DemoTokenClient.exe
Calling SF1520 with token...
Response from DemoService:
<?xml version="1.0" encoding="utf-16"?>
<PersonLookupResponseType xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
ema">
  <persondata xmlns="http://serviceplatformen.dk/xml/wsd1/soap11/CPR/PersonBaseDataExtended/4/">
    <personnummer>2905690000</personnummer>
    <gaeldendePersonnummer>2905690000</gaeldendePersonnummer>
    <navn>
      <personadresseringsnavn>Lone Hansentest</personadresseringsnavn>
      <fornavn>Lone</fornavn>
      <efternavn>Hansentest</efternavn>
      <navnestartdato>
        <dato>1969-08-23T13:00:00+02:00</dato>
      </navnestartdato>
    </navn>
    <foedselsdato>
      <dato>1969-05-29</dato>
    </foedselsdato>
    <foedselsregistreringsOplysninger>
      <foedselsregistreringsstedKode>8440</foedselsregistreringsstedKode>
      <foedselsregistreringsstedKodeMyndighedsnavn>Sæby, Frederikshavn</foedselsregistreringsstedKodeMyndighedsnavn>
      <supplerendeFoedselsregistreringssted>Sæby</supplerendeFoedselsregistreringssted>
    </foedselsregistreringsOplysninger>
    <startdato>
      <dato>1969-05-29</dato>
    </startdato>
    <alder>51</alder>
```

5 Kald CPR webservice - Java

5.1 Hent demoservice-klienten

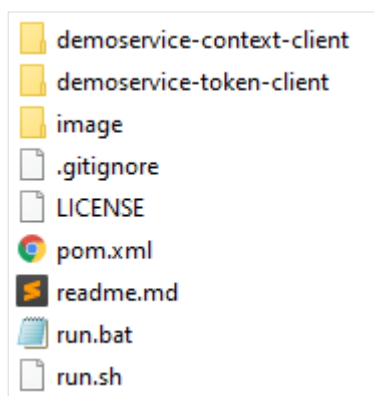
Hent Java kodeeksempel fra [Github](#):

demoservice-client-java

Java client for Serviceplatformens DemoService

● Java 3

Udpak filerne i din folder for Java-projekter. Indholdet i roden er som følger:





5.2 Forberedelse

Start med at udføre de første tre opgaver i forrige kapitel; afsnit 4.1, 4.2 og 4.3. Du har dermed en kopi af servicekontrakten, en serviceaftale til CPR-opslag service, samt dit funktionscertifikat klart.

Bemærk, at dit funktionscertifikat og certification authorities registreres i jks-filer, når koden afvikles i Java. Dette er beskrevet i [[KGIG-VEJL](#)] - *certifikater*.

Læs afsnit "PREREQUISITES" i readme.md filen i roden af projektet. Check at dit Java JDK er installeret og kørende:

```
C:\Users\xmag>javac -version
javac 1.8.0_271
```

Samt check at [Maven](#) er installeret og kørende:

```
C:\Users\xmag>mvn --version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: C:\Maven\bin\..
Java version: 1.8.0_271, vendor: Oracle Corporation, runtime: C:\Program
Files\Java\jdk1.8.0_271\jre
Default locale: en_GB, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

5.3 Tilføj certifikater

Referencer i det efterfølgende startende med "..\" er relative til:

```
\DemoserviceClient\demoservice-token-client\
```

Placering:

- ..\src\main\resources\
 - ▢ client.jks
 - ▢ trust.jks

Erstat funktionscertifikat i client.jks med dit eget. Sæt adgangskode på keystore filen til det samme som adgangskoden til den private nøgle.

Se [[KGIG-VEJL](#)] - *certifikater* for hjælp til det praktiske. Java har eget Trust Store, som beskrevet i vejledningen, og hvilke Certification Authorities (CA) der er inkluderet i dette afhænger af hvilken version af JRE og JDK du anvender. Dette er også grunden til at trust.jks, der kommer med demo-koden, ikke indeholder alle nødvendige CA, men blot dem der manglede i udviklers lokale Java Trust store. Vi ser derfor ofte i praksis, at der er behov for at tilføje manglende CA til trust.jks alternativt Java Trust Store, så vær opmærksom på dette.



5.4 Tilføj servicekontrakt

Placering:

- ..\src\main\resources\contracts\
 - ▢ ..
 - ▢ DemoServiceMsg.xsd

Start med at slette alle eksisterende filer i \contracts\ folder, som er servicekontrakt for demoservice. Dernæst kopier alle filer fra *PersonBaseDataExtendedService.zip* til folderen:

- ..\src\main\resources\contracts\
 - ▢ ..
 - ▢ PersonBaseDataExtendedServiceMsg.xsd

Vi skal da ændre referencen til den nye servicekontrakt.

Placering:

- ..\
 - ▢ pom.xml

Udcommenter eller fjern referencen til demoservice-kontrakten, og tilføj reference til CPR service:

```
<!--  
<properties>  
  <service.wsdl.name>DemoService.wsdl</service.wsdl.name>  
  <service.wsdl.path>/src/main/resources/contracts/wsdl/token</service.wsdl.path>  
</properties>-->  
<properties>  
  <service.wsdl.name>PersonBaseDataExtendedService.wsdl</service.wsdl.name>  
  <service.wsdl.path>/src/main/resources/contracts/wsdl/token</service.wsdl.path>  
</properties>
```

5.5 Generer serviceklient

I roden af projektet, kørs kommando "mvn generate-sources". Starten af output er som følger:

```
C:\Users\xmag\Java\DemoserviceClient>mvn generate-sources  
[INFO] Scanning for projects...  
[INFO] -----  
[INFO] Reactor Build Order:  
[INFO] -----  
[INFO] demoservice-client-parent [pom]  
[INFO] DemoService Context Client [jar]  
[INFO] DemoService Token Client [jar]  
[INFO] -----  
[INFO] -----< dk.kombit.serviceplatformen:demoservice-client-parent >-----  
[INFO] Building demoservice-client-parent 1.0 [1/3]  
[INFO] -----[ pom ]-----
```



Service klient/proxy for *SF1520 CPR-opslag* er da genereret i folder:

- ..\target\generated-sources\dk\serviceplatformen\xml\wsdl\soap11\cpr\personbasedataextended_4\
 - 📄 ...
 - 📄 PersonBaseDataExtendedPortType.java
 - 📄 PersonBaseDataExtendedService.java
 - 📄 ...

Hvor de to grundlæggende filer (*PortType* og *Service*) er fremhævet på listen foroven.

Fra *Service* filen aflæser vi:

package	dk.serviceplatformen.xml.wsdl.soap11.cpr.personbasedataextended._4;
targetNamespace	http://serviceplatformen.dk/xml/wsdl/soap11/CPR/PersonBaseDataExtended/4/
WebEndpoint	PersonBaseDataExtendedPort

Fra *PortType* filen aflæser vi:

operationName	PersonLookup
Request klasse	PersonLookupRequestType
Response klasse	PersonLookupResponseType

5.6 Opdater konfiguration

Placering:

- ..\src\main\resources\
 - 📄 client.properties
 - 📄 cxf.xml
 - 📄 token.properties

I *client.properties* angiver du adgangskode til dine key stores samt alias på dit funktionscertifikat:



```
org.apache.ws.security.crypto.merlin.keystore.type=jks
org.apache.ws.security.crypto.merlin.file=client.jks
org.apache.ws.security.crypto.merlin.keystore.password=Test!234
org.apache.ws.security.crypto.merlin.keystore.alias=kombit as - kdi sts sftp iba test2

org.apache.ws.security.crypto.merlin.truststore.type=jks
org.apache.ws.security.crypto.merlin.truststore.file=trust.jks
org.apache.ws.security.crypto.merlin.truststore.password=
```

I *cx.xml* under `<jaxws:client>` er angivet `targetNamespace` og `endpoint` for service, og disse skal erstattes med værdier for CPR-service som blev aflæst i forrige afsnit. Yderligere skal `sts.applies-to` erstattes med `entityId` for CPR-service, som vi fandt i afsnit 4.2:

```
<jaxws:client name="{#targetNamespace}#WebEndpoint#" createdFromAPI="true">
  <jaxws:properties>
    ...
    <entry key="ws-security.sts.applies-to" value="#service EntityId#"/>
    ...
  </jaxws:properties>
  ...
</jaxws:client>
```

`{#targetNamespace}#WebEndpoint#`

bliver således:

`{http://serviceplatformen.dk/xml/wsdL/soap11/CPR/PersonBaseDataExtended/4/}PersonBaseDataExtendedPort`

Nederst i *cx.xml* skal du erstatte `targetNamespace` og `endpoint` i `<http:conduit>` samt ændre parametre for certifikat Key Stores hvis nødvendigt:

```
<http:conduit name="{#targetNamespace}#WebEndpoint#.http-conduit">
  <http:tlsClientParameters disableCNCheck="false">
    <sec:keyManagers keyPassword="Test!234">
      <sec:keyStore type="JKS" password="Test!234" resource="client.jks"/>
    </sec:keyManagers>
    <sec:trustManagers>
      <sec:keyStore type="JKS" password="" resource="trust.jks"/>
    </sec:trustManagers>
  </http:tlsClientParameters>
  <http:client AutoRedirect="true" Connection="Keep-Alive"/>
</http:conduit>
```

I *token.properties* angiver du CVR-nummer for myndighed du har CPR-service serviceaftale for. Til vores test har vi anvendt "KOMBIT" kommune (19435075):

```
municipality.cvr=19435075
accounting.info=Something
callers.service.call.identifier=javaTokenDemoClientCall
on.behalf.of.user=Donald Duck
```




5.7 Opret metode til kald af CPR-service

Placering:

- ..\src\main\java\dk\serviceplatformen\demoservice\client\factories\
 - ▢ TokenRequestFactory.java

I starten af filen, udkommenter eller fjern import af DemoService package, og tilføj import af CPR-service package:

```
//import dk.serviceplatformen.xml.wsdl.soap11.sp.demo._1.CallDemoServiceRequestType;  
import dk.serviceplatformen.xml.wsdl.soap11.cpr.personbasedataextended._4.PersonLookupRequestType;
```

Fjern eller udkommenter den eksisterende funktion `getDemoServiceRequestType()`.

Tilføj en ny funktion `getCprServiceRequest()`, der anvender request type fra CPR-service, og som kalder `setPNR()`:

```
public PersonLookupRequestType getCprServiceRequest(String message) {  
    final PersonLookupRequestType request = new PersonLookupRequestType();  
    final CallContextType callContext = getCallContext();  
    request.setCallContext(callContext);  
    request.setPNR(message);  
    return request;  
}
```

5.8 Kald den nye metode

Placering:

- ..\src\main\java\dk\serviceplatformen\demoservice\client\
 - ▢ ApplicationRunner.java

Fjern import af demoservice klasser og tilføj import af CPR-service klasser:

```
/*  
import dk.serviceplatformen.xml.wsdl.soap11.sp.demo._1.CallDemoServiceRequestType;  
...  
*/  
import dk.serviceplatformen.xml.wsdl.soap11.cpr.personbasedataextended._4.PersonLookupRequestType;  
import dk.serviceplatformen.xml.wsdl.soap11.cpr.personbasedataextended._4.PersonLookupResponseType;  
import dk.serviceplatformen.xml.wsdl.soap11.cpr.personbasedataextended._4.PersonBaseDataExtendedPortType;  
import dk.serviceplatformen.xml.wsdl.soap11.cpr.personbasedataextended._4.PersonBaseDataExtendedService;
```

I klassen `ApplicationRunner {}`, skift endpoint adresse:

```
public class ApplicationRunner {  
    private static final String ENDPOINT_URL = "https://exttest.serviceplatformen.dk:443/service/CPR/PersonBaseDataExtended/4";
```



Og deklarerer ny variabel `cprPort`:

```
//private DemoPortType demoPort;  
private PersonBaseDataExtendedPortType cprPort;
```

I funktion `init()`, udskift port initiering:

```
private void init() {  
  
    //final DemoPortType port = new DemoService().getDemoPort();  
    final PersonBaseDataExtendedPortType port = new PersonBaseDataExtendedService().getPersonBaseDataExtendedPort();  
  
    ((BindingProvider) port).getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, ENDPOINT_URL);  
  
    //demoPort = port;  
    cprPort = port;  
}
```

I metoden `callWithToken()` erstatter du den eksisterende kode til at kalde CPR-service i stedet. Som eksempel på læsning af data i payload returnerer vi her personens fornavn:

```
String callWithToken(String message) {  
    try {  
        final PersonLookupRequestType request = tokenRequestFactory.getCprServiceRequest(message);  
        final PersonLookupResponseType response = cprPort.personLookup(request);  
        String resultFornavn = response.getPersondata().getNavn().getFornavn();  
        System.out.println("**** " + resultFornavn + " ****");  
        return resultFornavn;  
    }  
}
```

Nu er vi klar til at teste. Kør "run" og vælg "2":

```
C:\Users\xmag\Java\DemoserviceClient>run  
Please choose a client to execute:  
1 - Context client  
2 - Token client
```

Resultat:

```
[INFO] Scanning for projects...  
[INFO]  
[INFO] -----< dk.kombit.serviceplatformen:demoservice-token-client >-----  
[INFO] Building DemoService Token Client 1.0  
...  
...  
13:15:33.083 [main] INFO o.a.t.u.n.NioSelectorPool - Using a shared selector for servlet write/read  
13:15:33.131 [main] INFO o.s.b.c.e.t.TomcatEmbeddedServletContainer - Tomcat started on port(s): 8181 (http)  
Enter test message or 'q' for exit
```



Indtast et CPR-nummer fra listen af CPR arketyper, fx 2905690000. Du vil herefter ved succes se en lang udskrift der inkluderer dump af token-request og token, samt persondata xml. Her i forkortet version:

```
13:17:33.268 [main] WARN o.a.c.w.p.AssertionBuilderRegistryImpl - No assertion builder for
type {http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702}WssSam1V20Token11
registered.
13:17:35.506 [main] INFO o.a.c.s.S.C.IWSTrust13Sync - Outbound Message
-----
ID: 1
Address: https://adgangsstyring.eksterntest-
stoettesystemerne.dk/runtime/services/kombittrust/14/certificatemixed
Encoding: UTF-8
Http-Method: POST
...
...
</ns5:PersonLookupResponse>
</soap:Body>
</soap:Envelope>
-----
**** Lone ****
13:03:37.711 [main] INFO d.s.d.c.Application - Started Application in 13.36 seconds (JVM
running for 36.545)
13:03:37.713 [Thread-3] INFO o.s.b.c.e.AnnotationConfigEmbeddedWebApplicationContext -
Closing
org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@5336
926c: startup date [Fri Oct 30 13:03:24 CET 2020]; root of context hierarchy
13:03:37.716 [Thread-3] INFO o.s.j.e.a.AnnotationMBeanExporter - Unregistering JMX-exposed
beans on shutdown
```

6 Appendiks - Næste trin

Du har nu fået din første webservice-integration til at virke med eget anvendersystem og certifikat. Inden du kan benytte koden i dit fagsystem, er der flere essentielle tilføjelser, du skal sørge for:


- Request-token koden, hvis du anvender .NET, skal adskilles til separat klasse, så du kan trække token, uden at kalde service.
- Logik skal tilføjes, så koden kan håndtere et token per service og myndighed.
- Logik skal tilføjes, der tjekker, om et token stadig er gældende, eller om det skal fornyes. Hvis token stadig er gyldigt, skal det genanvendes.
- Applikationen skal genanvende gyldige tokens på tværs af sessioner.

7 Appendiks - Servicesystemroller

Serviceudbyder kan specificere granuleret adgang til funktioner og data ved at definere roller på deres service, også kaldet servicesystemroller. En rolle kan i sig selv involvere dataafgrænsning, og der kan også være specifikke dataafgrænsningsparametre specificeret for en rolle. I de tilfælde hvor der ikke er afgrænsning med roller, er der angivet en "dummy" rolle:

Navn:	IndkomstAbonnementVedligehold - v2.0
Type:	Fælleskommunal Service
Udbyder:	Serviceplatformen
Dataafgrænsning:	dummy
Yderligere information:	

I simple tilfælde kan der blot være specificeret en "Læs" og "Skriv" rolle. I det følgende mere avancerede tilfælde er der specificeret fire dataafgrænsningsparametre på rollen:

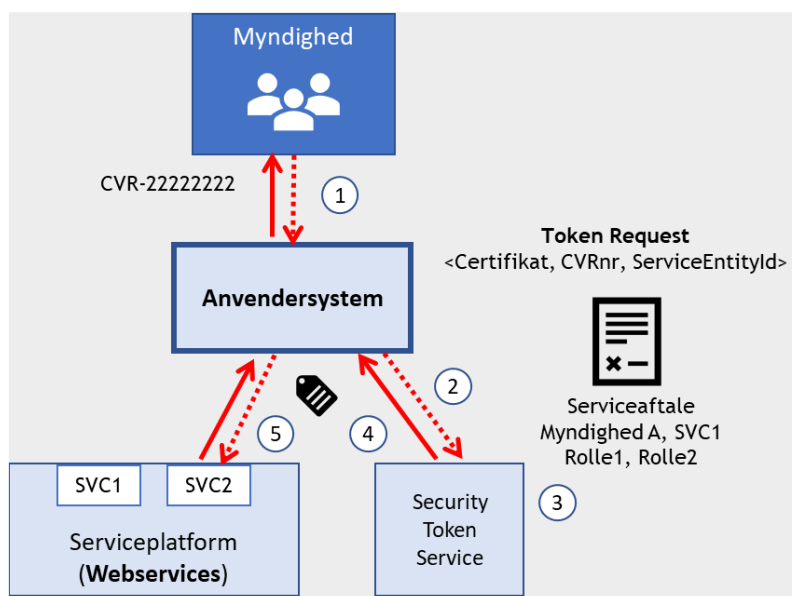
Navn:	afhent
Type:	Fælleskommunal Service
Udbyder:	Beskedfordeleren
Dataafgrænsning:	Modtag <ul style="list-style-type: none"> • Afsendende_myndighed: 19435075 • BeskedType: 7654895c-a3f3-4c77-8afd-4c77330f963c • Foelsomhed: 31c09910-e011-46a5-86fb-254374421fe8 • Kommunalt_forvaltningsomraade: *
Yderligere information:	

Hvis du er i tvivl om hvordan en serviceaftale skal udfyldes, så kontakt endeligt kdi@kombit.dk for kvalitetssikring. På den måde kan du være sikker på, at en serviceaftale er korrekt udfyldt, inden den sendes til godkendelse hos myndigheden.

8 Appendiks - Webservice Sikkerhed

Sikkerhedsmodellen ved kald til webservices er baseret på Security Tokens, der giver adgang til en specifik integration for en specifik myndighed og et specifikt anvendelsesystem (dit fagsystem har rollen "Anvendelsesystem" i denne sammenhæng).

Workflow er som følger:



1. En slutbruger fra en myndighed udfører en handling i Anvendelsesystemet (fagsystemet), der afstedkommer, at der skal foretages et kald til en webservice på infrastrukturen.
2. Anvendelsesystemet kalder Security Token Service (TOK) og anmoder om at få udstedt et token gældende:
 - a. Myndigheden (CVR-nummer)
 - b. Service identificeret ved service.EntityId
3. TOK kontrollerer om der foreligger en serviceaftale for angivne anvendelsesystem, myndighed og service.
4. TOK returnerer et token, som er gyldigt i otte timer.
5. Anvendelsesystem indlejrer det udstedte token i kaldet til den pågældende service.

Det er vigtigt at huske, at:

- Når en slutbruger udfører en handling i anvendelsesystemet, der medfører kald til en service, så skal der anvendes et token udstedt til pågældende myndighed.
- Tokens er gyldige i otte timer og skal genanvendes på tværs af applikationen. Du må ikke anmode om nyt token ved hvert kald til en service.

Dette er en kort beskrivelse af SAML token sikkerhed, som anvendes i de fleste udstillede webservices og dette eksempel. For mere information se [Fælleskommunalt Adgangsstyring for systemer](#).

9 Appendiks - Serviceaftaler

Alle forespørgsler til webservices sker på vegne af en myndighed. Der skal således eksistere en serviceaftale for dit anvendelsesystem for hver myndighed, du har behov for at kalde på vegne af. Det er en forudsætning for at kunne få udstedt et token, der giver adgang til at kunne kalde en service.

Du skal anmode hver myndighed om at få oprettet en serviceaftale, og hver myndighed skal godkende aftalen, før den virker. Du kan samle flere services i en anmodning, og du kan angive flere myndigheder i en anmodning.

Anmod om serviceaftale

Type	System	Myndigheder	Services	Parametre	Godkend
Serviceaftalstype: *	<input style="width: 100%;" type="text" value="Vælg fra liste"/> ?			Vælg serviceaftalstype og udfyld navn, gyldighedsperiode samt en begrundelse.	
Navn: *	<input style="width: 100%;" type="text"/>				
Gyldig fra: *	<input style="width: 100%;" type="text" value="2020-03-17"/>				
Gyldig til:	<input style="width: 100%;" type="text"/> ?				
Begrundelse: *	<input style="width: 100%;" type="text"/>				

Den detaljerede information finder du i [Vejledning til administrationsmodulet for leverandører](#). Visse integrationer udnytter mulighed for afgrænsning med parametre, som skal angives i serviceaftalen, og det er vigtigt, at de udfyldes korrekt for at fungere. Hvis du er i tvivl, bedes du kontakte kdi@kombit.dk og anmode om assistance, inden du opretter serviceaftalen.

10 Appendiks - Noter

10.1 Netværkssikkerhed

Demoservice eksempelkode tillader, at der anvendes gamle HTTPS-protokoller der ikke længere er understøttet. Siden koden blev lavet er de gamle protokoller ikke længere understøttet af serviceplatformen jf nyhed under [netværkssikkerhed](#).



```
DemoPortType channel = CreateChannel(token, endpointUrl);  
// Security protocols supported by the DemoService  
ServicePointManager.SecurityProtocol =  
    SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 | SecurityProtocolType.Ssl3;
```

Klienter kan da kun køre TLS 1.2 med server og du kan fjerne de gamle protokoller:

```
DemoPortType channel = CreateChannel(token, endpointUrl);  
// Security protocols supported by the DemoService  
ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
```

10.2 Fejl "The remote server returned an error: (417) Expectation Failed"

Denne fejl skyldes at klienten ved POST-request kan sende en request header "Expect: 100-continue" for at undersøge om server initielt svarer med HTTP 100 Continue inden HTTP 200 OK. Hvis server ikke svarer POST-requests med initielt HTTP 100, da kommer nævnte fejl. Fejlen fikses ved at slå dette check fra:

```
BeskedfordelerPortType channel = CreateChannelWithAuthorityContext();  
ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;  
ServicePointManager.Expect100Continue = false;
```