DIGITALISERINGS KATALOGET

KOM GODT I GANG

WEBSERVICE OIOIDWS

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

August 2021

KOMB:T
Kommunernes it-fællesskab

1 Introduktion

Denne guide tager udgangspunkt i en klient, der er etableret i henhold til guiden Kom-godt-i-gang *Webservice*, der er beregnet til webservices udstillet med sikkerhedsprofilen Liberty Basic SOAP Binding (LBSB). Guiden beskriver hvorledes du opgraderer klienten, til at kalde en webservice, der udstillet med sikkerhedsprofilen OIOIDWS. Dokumentet beskriver opgaverne for henholdsvis SOAP og REST services:

Kald OIOIDWS webservice SOAP/.NET

Kald OIOIDWS webservice SOAP/Java

Kald OIOIDWS webservice REST/.NET

Kald OIOIDWS webservice REST/Java

Der gælder ved skrivende stund, at Liberty Basic SOAP Binding anvendes på services udstillet af Serviceplatformen. OIOIDWS SOAP anvendes på de nye versioner af Klassifikation og Organisation, der tilgås direkte. OIOIDWS REST anvendes på den nye version af Digital Post (NGDP).

2 Baggrundsdokumentation

Gældende SOAP

Ændringer i sikkerhedsprofilen i forhold til LBSB er beskrevet i *OIO IDWS SOAP profile V1.1.pdf* som du finder i [OIOIDWS] under *Ressourcer*.

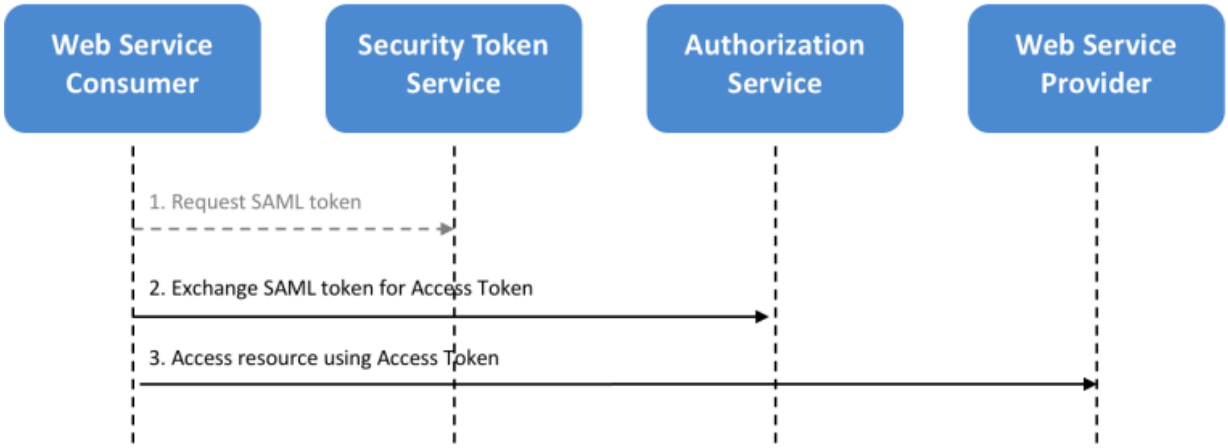
- *SOAP version 1.2 is used instead of version 1.1.*
- *The profile no longer mandates the specifics of SOAP faults.*
- *The Liberty framework header has been omitted.*
- *The header is no longer mandated.*
- *Requirements for a secure transport protocol has been added and message confidentiality using encryption of individual attributes in SAML Assertions is no longer recommended.*
- *Links to version 1.1.1 of the OASIS SAML Token Profile instead of version 1.0.*

Gældende REST

Her er sikkerhedsprofilen udstillet af webservice meget simpel. Webservice accepterer en *Authorization* http-header med et Access Token, der er en simpel nøgle:

```
GET /resource/1 HTTP/1.1
Authorization: Holder-of-key RsT5OjbzRn430zqMLgV3IaHjjuyG5
Host: dummy-site.net
```

Anvender trækker først som sædvanligt et SAML-token fra Security Token Service. Anvender kalder dernæst en Access Token service med SAML-token og får et Access Token tilbage. Sekvensen er beskrevet i *OIO IDWS REST profile V1_0.pdf*:



Det som ikke er illustreret på figuren er, at Webservice Provider efterfølgende laver opslag i Access Token Service og får information om servicesystemroller og eventuel dataafgrænsning, der var indeholdt i det oprindelige SAML-token.

Referencer

[OIOIDWS]	OIO Identity Based Web Services 1.2 (OIO IDWS) - OIO IDWS SOAP profile V1.1.pdf - OIO IDWS REST profile V1_0.pdf
[KGIG-WEBSVC]	Kom-godt-i-gang *Webservice*
[OIOIDWS.Net]	https://github.com/digst/OIOIDWS.Net
[DEMOSVC]	Serviceplatformen eksempelkode på Github
[OIOIDWS.Java.REST]	OIOIDWS.Java (REST)

For øvrige referencer, konfiguration og opgaver (certifikater, serviceaftaler, servicekontrakter, service entityIDs), venligst se [KGIG-WEBSVC], som du skal læse og gennemføre inden denne guide.

3 Kald OIOIDWS webservice SOAP/.NET

Opgaver du skal udføre, for at opgradere demoservice-client-net, til at kunne kalde en SOAP webservice udstillet med sikkerhedsprofil OIOIDWS:

- Fjern LBSB biblioteker fra Solution
- Hent OIOIDWS kildekode til .NET
- Opdater Digst.OioIdws projektfiler
- Tilføj Digst.OioIdws projekter til Solution
- Tilføj referencer til Digst.OioIdws biblioteker
- Opdater WCF konfiguration
- Opdater metode createChannel
- Opdater SoapBinding.cs

Det er en forudsætning, at du inden har udført opgaverne beskrevet i [KGIG-WEBSVC].

3.1 Note om OIOIDWS NuGet pakke

Digitaliseringsstyrelsen vedligeholder en NuGet pakke, der gør at man enkelt kan installere nødvendige biblioteker og konfigurere en klient til at kalde en webservice udstillet med OIOIDWS:

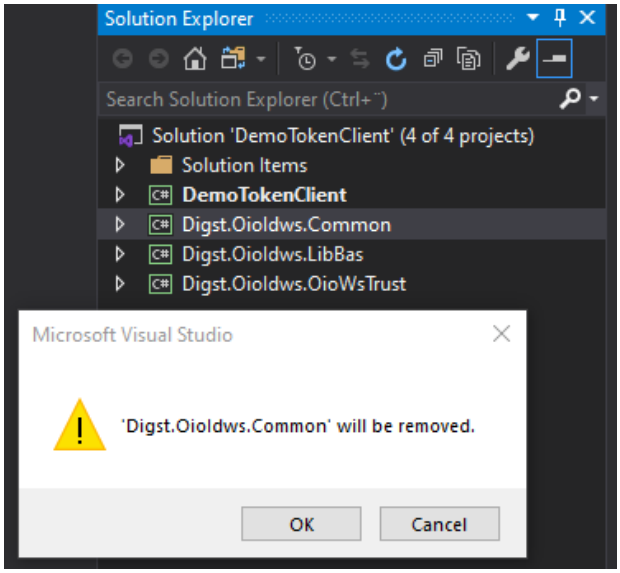
https://www.nuget.org/packages/Digst.OioIdws.Wsc/

Det er dog ikke muligt p.t. at benytte denne pakke, da den ikke tillader ændring af algoritme, der anvendes til signering af beskeder. Referencekoden fra Digitaliseringsstyrelsen benytter WCF standard-indstilling, mens at den fælleskommunale infrastruktur benytter SHA256.

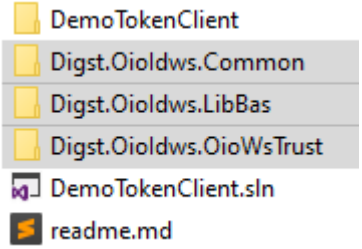
Fremgangsmåden her er således, at benytte kildekoden, der ligger til grund for NuGet pakken, så vi kan ændre signerings-algoritmen og kompilere OIOIDWS-bibliotekerne sammen med vores projekt.

3.2 Fjern LBSB biblioteker fra Solution

Højre-klik på de tre Digst.OioIdws projekter og vælg ”Remove”:

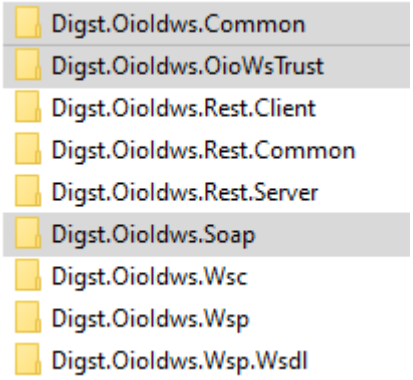


Slet dernæst de tre Digst.OioIdws foldere i filsystemet:



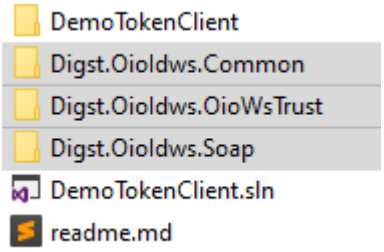
3.3 Hent OIOIDWS kildekode til .NET

Hent kildekoden her: https://github.com/digst/OIOIDWS.Net og udpak filerne i en midlertidig folder:



Kopier dernæst de tre markerede foldere (*Common, OioWsTrust, Soap*) over i din solution:

\DemoTokenClient



3.4 Opdater Digst.OioIdws projektfiler

Start med at se hvilken version af .NET du kører (i dette eksempel er *DemoTokenClient* hovedprojekt og det er sat til version 4.5.2):

\DemoTokenClient\DemoTokenClient\DemoTokenClient.csproj

```
    <AssemblyName>DemoTokenClient</AssemblyName>
    <TargetFrameworkVersion>v4.5.2</TargetFrameworkVersion>
    <FileAlignment>512</FileAlignment>
```

Du skal opdatere hver *Digst.OioIdws* projekt-fil:

\DemoTokenClient\Digst.OioIdws.Common\Digst.OioIdws.Common.csproj
\DemoTokenClient\Digst.OioIdws.OioWsTrust\Digst.OioIdws.OioWsTrust.csproj
\DemoTokenClient\Digst.OioIdws.Soap\Digst.OioIdws.Soap.csproj

Og:

- Sætte samme *TargetFrameworkVersion* som dit eget hovedprojekt
- Udkommentere eller fjerne reference til *PreBuild-targets.xml*

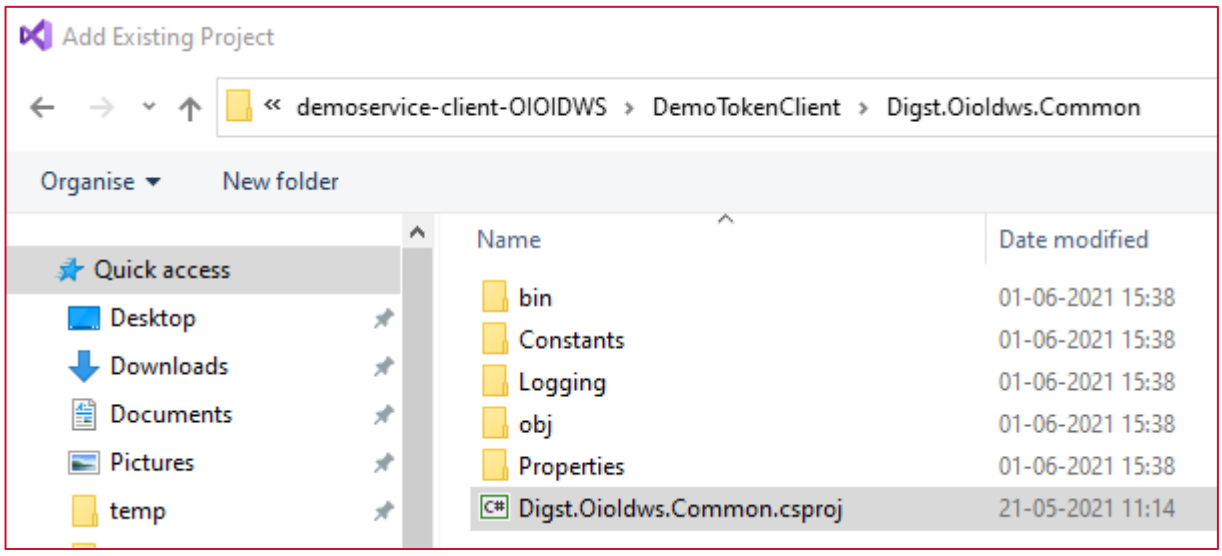
```
<Project ...>
   ...
   <PropertyGroup>
      ...
      <TargetFrameworkVersion>v4.5.2</TargetFrameworkVersion>
      ...
  <!-- <Import Project="..\..\build\PreBuild.targets.xml" /> -->
</Project>
```

3.5 Tilføj Digst.OioIdws projekter til Solution

Start solution:

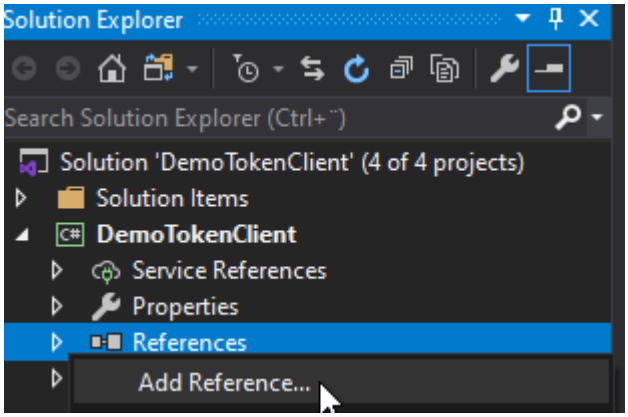
\DemoTokenClient\DemoTokenClient.sln

Højre-klik Solution i Visual Studio Solution Explorer og vælg "Add -> Existing Project". Tilføj alle tre projekter (*Common, OioWsTrust, Soap*) individuelt:

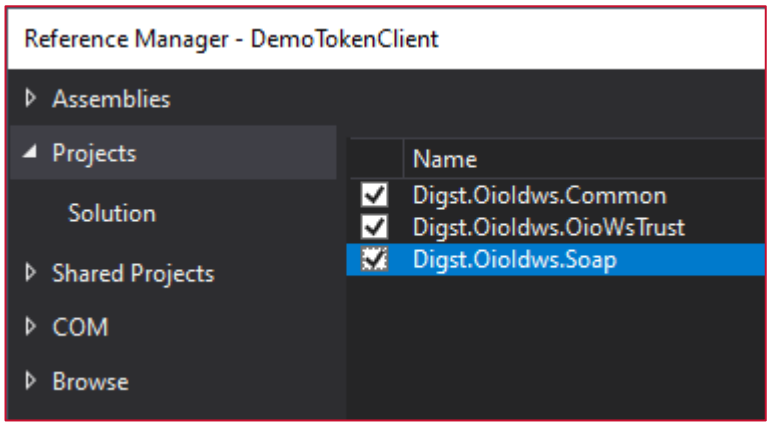


3.6 Tilføj referencer til Digst.OioIdws biblioteker

I hovedprojekt *DemoTokenClient*, højre-klik "References" og vælg "Add Reference":



Vælg “Projects” og marker de tre Digst.OioIdws projekter:



3.7 Opdater WCF konfiguration

Følgende er den komplette blok af <system.serviceModel> du skal benytte i App.config. Parametre du kan/skal tilpasse er markeret med gult, med tilhørende kommentarer efterfølgende.

```
  <system.serviceModel>
    <extensions>
      <bindingExtensions>
        <add name="SoapBinding"
type="Digst.OioIdws.Soap.Bindings.SoapBindingCollectionElement, Digst.OioIdws.Soap"/>
      </bindingExtensions>
      <behaviorExtensions>
        <add name="SoapBehavior"
type="Digst.OioIdws.Soap.Behaviors.SoapClientBehaviorExtensionElement, Digst.OioIdws.Soap"/>
      </behaviorExtensions>
    </extensions>
    <behaviors>
      <endpointBehaviors>
        <behavior name="SoapBehaviourConfiguration">
          <clientCredentials>
            <serviceCertificate>
              <defaultCertificate findValue="ec1f5b59d048e4e475ba1c49eab93e5432c2f387"
storeLocation="CurrentUser" storeName="My" x509FindType="FindByThumbprint"/>
            </serviceCertificate>
          </clientCredentials>
          <SoapBehavior/>
        </behavior>
      </endpointBehaviors>
    </behaviors>
    <bindings>
        <SoapBinding>
            <binding name="SoapBindingConfiguration" useHttps="true" />
        </SoapBinding>
    </bindings>
    <client>
        <endpoint address="https://klassifikation.eksterntest-
stoettesystemerne.dk/klassifikation_6"
```

```
            binding="SoapBinding" bindingConfiguration="SoapBindingConfiguration"
behaviorConfiguration="SoapBehaviourConfiguration"
            contract="SF1510v6.KlassifikationPortType" name="Klassifikation" />
    </client>
      ...
  </system.serviceModel>
```

Kommentarer til felter markeret med gult:

- ServiceCertificate angiver certifikat der anvendes til signering af svar. I vores eksempel svar fra Klassifikation v6 service, der signerer med “Klassifikation_T (funktionscertifikat)”.
- StoreLocation og storeName skal ændres, hvis din kode ikke kører i kontekst af lokale bruger (typisk under udvikling).
- Endpoint-konfiguration er i dette eksempel sat til kald af Klassifikation v6 webservice i testmiljøet. Skal erstattes med parametre for service du skal kalde.

Husk også at opdatere parametre i ConfigVariables.cs, som beskrevet i [KGIG-WEBSVC].

3.8 Opdater metode createChannel

I metoden, der opretter forbindelse til webservice, skal du ændre ProtectionLevel fra “Sign” til “None” (i dette eksempel for service *Klassifikation*):

\DemoTokenClient\Token\DemoServiceToken.cs

```
        private KlassifikationPortType CreateChannel(SecurityToken token)
        {
             ...
            client.Endpoint.Contract.ProtectionLevel = ProtectionLevel.None;

            return client.ChannelFactory.CreateChannelWithIssuedToken(token);
        }
```

Bemærk, at dette p.t. gør sig gældende for de OIOIDWS SOAP services, der er udstillet direkte af støttesystemerne (udenom serviceplatformen). Det kan tænkes, at fremtidige OIOIDWS SOAP webservices udstillet af serviceplatformen kræver *ProtectionLevel.Sign*. Så vær opmærksom på dette, hvis kald til nye OIOIDWS SOAP services fejler ved oprettelse af channel.

3.9 Opdater SoapBinding.cs

Du skal nu sætte signeringsalgoritme til SHA256. Tilføj linje markeret med fed til metoden CreateBindingElements() i SoapBinding.cs:

\DemoTokenClient\Digst.OioIdws.Soap\Bindings\SoapBinding.cs

```
public override BindingElementCollection CreateBindingElements() {
   ...
   asymmetric.ProtectTokens = true;
   asymmetric.DefaultAlgorithmSuite = SecurityAlgorithmSuite.Basic256Sha256;
```

Du har nu opdateret demoservice-client-net referencekoden, til at virke med en webservice udstillet med sikkerhedsmodel OIOIDWS SOAP i den fælleskommunale infrastruktur.

4 Kald OIOIDWS webservice SOAP/Java

Opgaver du skal udføre, for at opgradere demoservice-client-java, til at kalde en SOAP webservice udstillet med sikkerhedsprofil OIOIDWS::

- Opdater CXF konfiguration
- Opdater Security Token Service klient

Det er en forudsætning, at du inden har udført opgaverne beskrevet i [KGIG-WEBSVC].

4.1 Opdater CXF konfiguration

Når du har tilføjet reference til den nye WSDL i pom.xml og kørt ”mvn generate-sources”, for at generere service-klient proxy klasser, da skal du som sædvanligt angive det nye namespace, den nye port og Service EntityId. Dernæst skal du fjerne konfigurationen for de to message interceptors, da de er relateret til LBSB.

resources\cxf.xml

```
<jaxws:client name="{http://serviceplatformen.dk/xml/wsdl/soap11/SP/Demo/1/}DemoPort"
createdFromAPI="true">
   <jaxws:properties>
      <entry key="ws-security.encryption.properties" value="client.properties"/>
      <entry key="ws-security.signature.properties" value="client.properties"/>
      <entry key="ws-security.callback-handler"
value="dk.serviceplatformen.demoservice.client.sts.PasswordCallbackHandler"/>
      <entry key="ws-security.sts.applies-to" value="http://demo.prod-
serviceplatformen.dk/service/DemoService/1"/>
      <entry key="ws-security.asymmetric.signature.algorithm"
value="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
      <entry key="ws-security.sts.client">
         <ref bean="kombitSTSClient"/>
      </entry>
   </jaxws:properties>
   <jaxws:inInterceptors>   <!-- DENNE BLOK SKAL FJERNES  -->
      <bean id="FrameworkInInterceptor"
class="dk.serviceplatformen.demoservice.client.interceptors.UnderstandFrameworkHeaderInInter
ceptor"/>
   </jaxws:inInterceptors>
   <jaxws:outInterceptors>
      <bean id="FrameworkOutInterceptor"
class="dk.serviceplatformen.demoservice.client.interceptors.FrameworkHeaderOutInterceptor"/>
   </jaxws:outInterceptors>
</jaxws:client>

<http:conduit name="{http://serviceplatformen.dk/xml/wsdl/soap11/SP/Demo/1/}DemoPort.http-
conduit">
   <http:tlsClientParameters disableCNCheck="false">
      <sec:keyManagers keyPassword="********">
         <sec:keyStore type="JKS" password="********" 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>
```

4.2 Opdater Security Token Service klient

Følgende fejlbesked kom ved hentning af et token, da vi ændrede koden til at kalde en service udstillet med OIOIDWS:

ID3007: The element 'KeyType' with namespace 'http://www.w3.org/2006/05/addressing/wsdl' is unrecognized

Security Token Services (STS) kaldes før webservice, og CXF-konfiguration af STS er uændret. Det antages, at den ændrede sikkerhedsprofil i webservice vi kalder har fået CXF til at benytte andre standard-indstillinger, hvorefter kald til STS er ændret. Fejlen rettes ved tilføjelse af følgende to linjer markeret med fed til metoden issue() i STSClient.java:

java\dk\serviceplatformen\demoservice\client\sts\STSClient.java

```
@Override
protected STSResponse issue(String appliesTo, String action, String requestType, String
binaryExchange) throws Exception {
   ((HTTPConduit)
getClient().getConduit()).getClient().setContentType("application/soap+xml; charset=utf-8");
   this.template = null;
   this.wspNamespace = "http://schemas.xmlsoap.org/ws/2004/09/policy";
   return super.issue(appliesTo, action, requestType, binaryExchange);
}
```

Du har nu opdateret demoservice-client-java referencekoden, til at virke med en webservice udstillet med sikkerhedsmodel OIOIDWS SOAP i den fælleskommunale infrastruktur.

5 Kald OIOIDWS webservice REST/.NET

Opgaver du skal udføre, for at opgradere demoservice-client-net, til at kalde en REST webservice udstillet med sikkerhedsprofil OIOIDWS:

- Tilføj pakke Newtonsoft.Json
- Opret REST Klient klasse
- Tilføj hjælpefunktioner
- Tilføj funktion ConvertToAccessToken()
- Tilføj funktion SendRequest()
- Tilføj funktion CallService()
- Opdater Program.cs

Vores demoservice-client-net kode indeholder allerede kode og konfiguration, til at kalde STS og hente et SAML-token. Vi tilføjer blot en ny klient, der kan veksle SAML-token til et Access Token, og dernæst kalde REST-service med dette.

Der er lånt kode fra [OIOIDWS.Net] og dette er et simpelt eksempel til illustration af den grundlæggende funktionalitet. Når du efterfølgende i din egen kode skal håndtere fornyelse af tokens, da kan du overveje at indarbejde referencekoden fra Digitaliseringsstyrelsen. Koden angivet her har kun til formål, at illustrere hvorledes det virker.

Note: Koden er testet mod en lokalt kørende Webservice Provider og Access Token Service fra OIOIDWS.Java (REST). Konfiguration af disse er ikke gengivet her.

5.1 Tilføj pakke Newtonsoft.Json

I Visual Studio, vælg “Tools -> NuGet Package Manager -> Package Manager Console”. Udfør følgende kommando:

```
PM> Install-Package Newtonsoft.Json
```

5.2 Opret REST Klient klasse

Tilføj en ny *RESTClient.cs* fil til projektet med følgende indhold:

```
using System;
using System.Net;
using System.IdentityModel.Tokens;
using DemoTokenClient.Token;
using System.Text;
using System.IO;
using System.Xml;
using Newtonsoft.Json.Linq;

namespace DemoTokenClient
{
   class RESTClient
   {

   }
}
```

5.3 Tilføj hjælpefunktioner

Tilføj følgende funktioner til *RESTClient* klassen:

```
private string EncodeSAMLToken(GenericXmlSecurityToken token)
{
   string tokenXml = token.TokenXml.OuterXml;
   byte[] bytes = Encoding.UTF8.GetBytes(tokenXml);
   string encodedToken = Convert.ToBase64String(bytes);
   return System.Web.HttpUtility.UrlEncode(encodedToken);
}
```

```
private string ReadResponseData(Stream input)
{
   byte[] buffer = new byte[16 * 1024];
   using (MemoryStream ms = new MemoryStream())
   {
      int read;
      while ((read = input.Read(buffer, 0, buffer.Length)) > 0)
      {
         ms.Write(buffer, 0, read);
      }
      return Encoding.Default.GetString(ms.ToArray());
   }
}
```

```
public static string GetUTCDateTime()
{
   return DateTime.Now.ToString("yyyy-MM-ddTHH:mm:sszzz");
}
```

```
private class AccessToken
{
   public string Value { get; set; }
   public string Type { get; set; }
   public TimeSpan ExpiresIn { get; set; }
   public DateTime RetrievedAtUtc { get; set; }
   public bool IsValid()
   {
      return (RetrievedAtUtc + ExpiresIn) > DateTime.UtcNow;
   }
}
```

```
private void AddClientCertificate(ref HttpWebRequest request)
{
   request.ClientCertificates.Add(
      CertificateLoader.LoadCertificate(
         ConfigVariables.ClientCertificateStoreName,
         ConfigVariables.ClientCertificateStoreLocation,
         ConfigVariables.ClientCertificateThumbprint));
}
```

5.4 Tilføj funktion ConvertToAccessToken()

Denne metode kalder Access Token Service med SAML-token og returnerer et Access Token. Tilføj følgende funktion til *RESTClient* klassen:

```
private AccessToken ExchangeToAccessToken(GenericXmlSecurityToken token)
{
   HttpWebRequest webrequest =
(HttpWebRequest)WebRequest.Create("https://localhost:8443/auth");
   webrequest.Method = "POST";
   webrequest.ContentType = "application/x-www-form-urlencoded";
   AddClientCertificate(ref webrequest);

   string postData = "saml-token=" + EncodeSAMLToken(token);

   UTF8Encoding encoding = new UTF8Encoding();
   byte[] body = encoding.GetBytes(postData);

   webrequest.ContentLength = body.Length;
   Stream newStream = webrequest.GetRequestStream();
   newStream.Write(body, 0, body.Length);

   WebResponse response = webrequest.GetResponse();
   string responseData = ReadResponseData(response.GetResponseStream());

   JObject jsonValue = JObject.Parse(responseData);
   AccessToken accessToken = new AccessToken
   {
      Value = (string)jsonValue["access_token"],
      ExpiresIn = TimeSpan.FromSeconds((int)jsonValue["expires_in"]),
      RetrievedAtUtc = DateTime.UtcNow,
      Type = (string)jsonValue["token_type"]
   };

   return accessToken;
}
```

Du skal erstatte markeret med blåt:

- Adresse til Access Token Service

5.5 Tilføj funktion SendRequest()

Denne metode kalder REST-service med Access Token. Husk at sætte korrekte adresse markeret med blåt. Tilføj følgende funktion til *RESTClient* klassen:

```
private string SendRequest(AccessToken token, string method, string contentType, string
requestUri, string postData)
{
   HttpWebRequest webrequest = (HttpWebRequest)WebRequest.Create(requestUri);
   webrequest.Method = method;
   webrequest.ClientCertificates.Add(Util.GetClientCertificate());

   webrequest.Headers.Set(HttpRequestHeader.Authorization, "Holder-of-key " + token.Value);

   string guid = Guid.NewGuid().ToString();
   string timeStamp = GetUTCDateTime();

   webrequest.Headers.Add("x-TransaktionsId", guid);
   webrequest.Headers.Add("x-TransaktionsTid", timeStamp);

   if (method == "POST" || method == "PUT")
   {
      webrequest.ContentType = contentType;

      UTF8Encoding encoding = new UTF8Encoding();
      byte[] body = encoding.GetBytes(postData);

      webrequest.ContentLength = body.Length;
      Stream newStream = webrequest.GetRequestStream();
      newStream.Write(body, 0, body.Length);
   }

   WebResponse response = webrequest.GetResponse();

   responseData = ReadResponseData(response.GetResponseStream());

   return responseData;
}
```

Bemærk de to HTTP headers ”x-TransaktionsId” og ”x-TransaktionsTid”. De er påkrævet ved alle kald til REST-services udstillet i den fælleskommunale infrastruktur. Din applikation bør logge disse, således at de kan inkluderes i henvendelser til helpdesk i forbindelse med fejlsøgning.

5.6 Tilføj funktion CallService()

Dette er hoved-metoden i klassen. Tilføj følgende funktion til *RESTClient* klassen:

```
public void CallService()
{
   // Get SAML token
   ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
   GenericXmlSecurityToken token =
(GenericXmlSecurityToken)TokenFetcher.IssueToken(ConfigVariables.ServiceEntityId);
   Console.WriteLine("Token recevied: " + token.Id);

   // Convert to Access Token
   AccessToken accessToken = ConvertToAccessToken(token);
   Console.WriteLine("Access Token: " + accessToken.Value);

   // Call REST service
   string requestUri = "https://localhost:8443/api/hello?name=Joe";
   string responseData = SendRequest(accessToken, “GET”, “”, requestUri, “”);
   Console.WriteLine("Response:\n" + responseData);
}
```

5.7 Opdater Program.cs

Til sidst sætter vi programmet til at kalde vores REST-service i stedet for Demoservice:

```
namespace DemoTokenClient
{
    public class Program
    {
        public static void Main(string[] args)
        {
            try {
               RESTClient client = new RESTClient();
               client.CallService();
               Console.WriteLine("\nPress any key to exit");
               Console.ReadKey();
            }
            ...
        }
    }
}
```

Du har nu opdateret demoservice-client-net referencekoden, til at virke med en webservice udstillet med sikkerhedsmodel OIOIDWS REST i den fælleskommunale infrastruktur.

6 Kald OIOIDWS webservice REST/Java

Her tager vi udgangspunkt i referencekoden OIOIDWS.Java (REST) fra Digitaliseringsstyrelsen. Denne er konfigureret, til at anvende Nemlog-in Security Token Service (STS), og vi skal ændre kode og konfiguration til at virke med vores STS. Opgaver du skal udføre, for at opdatere OIOIDWS.Java (REST), til at virke med STS i den fælleskommunale infrastruktur:

- Opdater ClientCallbackHandler.java
- Opdater ClientConfig.java
- Tilføj ClaimsCallbackHandler.java
- Erstat STSClient.java
- Opdater TokenFetcher.java
- Opdater konfiguration
- Opdater Application.java

De resterende opgaver (*Opdater konfiguration*) er de samme som beskrevet i [KGIG-WEBSVC], og er gentaget her i kort version som huskeliste, da vi ikke benytter demoservice-client-java som udgangspunkt.

6.1 Opdater ClientCallbackHandler.java

\rest-client\src\main\java\client\callback\ClientCallbackHandler.java

Sæt adgangskode for dit funktionscertifikats private nøgle:

```
public void handle(Callback[] callbacks) throws IOException, UnsupportedCallbackException {
   for (int i = 0; i < callbacks.length; i++) {
      if (callbacks[i] instanceof WSPasswordCallback) {
         WSPasswordCallback pc = (WSPasswordCallback) callbacks[i];
         pc.setPassword("******");
      }
   }
}
```

6.2 Opdater ClientConfig.java

\rest-client\src\main\java\client\config\ClientConfig.java

Angiv sti til dit funktionscertifikat (placeret under \rest-client\src\main\resources) og sæt også her adgangskode til private nøgle:

```
public class ClientConfig {

   @Value("classpath:KDI_STS_SFTP_IBA_Test2.p12")
   ...
   public ClientHttpRequestFactory httpComponentsClientHttpRequestFactory() throws ...
      ks.load(inputStream, "******".toCharArray());
      builder.loadKeyMaterial(ks, "******".toCharArray());
```

6.3 Tilføj ClaimsCallbackHandler.java

\rest-client\src\main\java\client\sts\ClaimsCallbackHandler.java

Opret denne fil og indsæt indholdet forneden:

```
package client.sts;

import java.io.IOException;
import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;
import org.apache.cxf.helpers.DOMUtils;
import org.apache.cxf.ws.security.trust.claims.ClaimsCallback;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class ClaimsCallbackHandler implements CallbackHandler {
   private static String cvr = "19435075";

   @Override
   public void handle(Callback[] callbacks) throws IOException, UnsupportedCallbackException
{
      for (int i = 0; i < callbacks.length; i++) {
         if (callbacks[i] instanceof ClaimsCallback) {
            ClaimsCallback callback = (ClaimsCallback) callbacks[i];
            callback.setClaims(createClaims());
         }
         else {
            throw new UnsupportedCallbackException(callbacks[i], "Unrecognized Callback");
         }
      }
   }

   private static Element createClaims() {
      Document doc = DOMUtils.createDocument();
      Element claimsElement = doc.createElementNS("http://docs.oasis-open.org/ws-sx/ws-
trust/200512", "Claims");
      claimsElement.setAttributeNS(null, "Dialect", "http://docs.oasis-
open.org/wsfed/authorization/200706/authclaims");

      Element claimType = doc.createElementNS("http://docs.oasis-
open.org/wsfed/authorization/200706", "ClaimType");
      claimType.setAttributeNS(null, "Uri", "dk:gov:saml:attribute:CvrNumberIdentifier");
      claimsElement.appendChild(claimType);

      Element claimValue = doc.createElementNS("http://docs.oasis-
open.org/wsfed/authorization/200706", "Value");
      claimValue.setTextContent(cvr);
      claimType.appendChild(claimValue);

      return claimsElement;
   }

   public static String getCvr() {
      return cvr;
   }

   public static void setCvr(String cvr) {
      ClaimsCallbackHandler.cvr = cvr;
   }
}
```

Husk! at ændre CVR-nummer, markeret med blåt, til CVR-nummer du tester med og har serviceaftale for.

6.4 Erstat STSClient.java

Erstat hele STSClient.java med indholdet forneden.

\rest-client\src\main\java\client\sts\STSClient.java

```
package client.sts;

import java.security.cert.X509Certificate;
import org.apache.cxf.Bus;
import org.apache.cxf.staxutils.W3CDOMStreamWriter;
import org.apache.cxf.transport.http.HTTPConduit;
import org.apache.xml.security.utils.Base64;

public class STSClient extends org.apache.cxf.ws.security.trust.STSClient {

   public STSClient(Bus b) {
      super(b);
   }

   @Override
   protected STSResponse issue(String appliesTo, String action, String requestType, String
binaryExchange) throws Exception {
      // the STS does not like the CXF generated Content-Type header, so we overwrite with a
custom one that the STS likes
      ((HTTPConduit)
getClient().getConduit()).getClient().setContentType("application/soap+xml; charset=utf-8");

      // bit of a hack, but CXF adds a different namespace, probably due to some value in the
sts.wsdl file
      this.wspNamespace = "http://schemas.xmlsoap.org/ws/2004/09/policy";

      return super.issue(appliesTo, action, requestType, binaryExchange);
   }

   @Override
    protected void writeElementsForRSTPublicKey(W3CDOMStreamWriter writer, X509Certificate
cert) throws Exception {
        writer.writeStartElement("wst", "UseKey", namespace);
        writer.writeStartElement("wsse", "BinarySecurityToken", "http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd");
        writer.writeAttribute("EncodingType", "http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-soap-message-security-1.0#Base64Binary");
        writer.writeAttribute("ValueType", "http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509v3");

        // the KOMBIT STS requires that the UseKey contains a BinarySecurityToken, so the
usual X509Data that the CXF can supply will not work
        String encodedCert = Base64.encode(cert.getEncoded());
        writer.writeCharacters(encodedCert);

        writer.writeEndElement();
        writer.writeEndElement();
    }
}
```

Note: Identisk med STSClient.java fra demoservice-client-java, dog er følgende tilføjet til metoden *issue()* (markeret med blåt):

this.wspNamespace = "http://schemas.xmlsoap.org/ws/2004/09/policy";

6.5 Opdater TokenFetcher.java

Her angiver du sti til Access Token Service:

\rest-client\src\main\java\client\sts\TokenFetcher.java

```
headers.add("Content-Type", "application/x-www-form-urlencoded;charset=UTF-8");
ResponseEntity<AccessToken> authorizationServiceResponse =
restTemplate.exchange("https://localhost:8443/auth", HttpMethod.POST, new
HttpEntity<>(encodedToken, headers), AccessToken.class);
accessToken = authorizationServiceResponse.getBody();
```

6.6 Opdater konfiguration

\rest-client\src\main\resources\client.properties

- <tjek alle parametre>

\rest-client\src\main\resources\cxf.xml

- ws-security.signature.username[1]
- ws-security.sts.token.username[1]
- ws-security.encryption.username[2]

(1) Dit funktionscertifikat *Subject*

(2) STS Signering certifikat *Subject*

\rest-client\src\main\resources\sts.properties

- org.apache.ws.security.crypto.merlin.keystore.alias

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

- Service/STS/AS HTTPS certifikat
- STS signering certifikat
- Certification Authorities for ovenstående

6.7 Opdater Application.java

Angiv Service EntityID og REST-service endpoint her:

\rest-client\src\main\java\client\Application.java

```
public void run(String... args) throws Exception {

   // get the access token
   AccessToken accessToken = tokenFetcher.getAccessToken("<Service EntityID>");

   // setup request Authorization header
   HttpHeaders headers = new HttpHeaders();
   headers.add("Authorization", "Holder-of-key " + accessToken.getToken());

   // call service
   ResponseEntity<String> restServicResponse =
restTemplate.exchange("https://localhost:8443/api/hello?name=John", HttpMethod.GET, new
HttpEntity<>("", headers), String.class);

   // should print out "Hello John"
   logger.info(restServicResponse.toString());
}
```

Husk at CVR-nummer gældende serviceaftalen sættes i ClaimsCallbackHandler.java.

Husk at du også her skal sætte HTTP headers ”x-TransaktionsId” og ”x-TransaktionsTid”. Du kan se eksempel i afsnit 5.5.

Du har nu opdateret OIOIDWS.Java (REST) referencekoden, til at virke med en webservice udstillet med sikkerhedsmodel OIOIDWS REST i den fælleskommunale infrastruktur