PENSAB - lønservices (Notat)

AF / Tom Bisgård Sørensen, e-mail: tom.sorensen@cgi.com

Dette notat beskriver principperne for hvordan en klient til PENSABs eksterne lønservices kan konstrueres. Undervejs gennemgås kommunikations-, informations- og sikkerhedsmodellerne for de udstillede services og det eksemplificeres hvordan en klient til disse services konstrueres i Java.

1. Formål

PENSAB har behov for at lade lønservices indberette oplysninger om tjenestemænds ansættelsesforhold for efterfølgende at benytte disse til beregning af tjenestemænds pensionsalder mm.

PENSAB stiller derfor en række services (i praksis web services) til rådighed for at disse lønsystemer (herefter kaldet 'lønsystemet'), som giver mulighed for at aflevere oplysninger om hver enkelt tjenestemand og dennes stamoplysninger samt oplysninger om tjenestemandens ansættelsesperioder i form af periodens udstrækning, tjenestemandens ansættelsessted mm.

2. Generelt

De udstillede services vil generelt være tilgængelige i to adskilte PENSAB-miljøer, test og produktion.

3. Kommunikationsmodel

PENSAB tilbyder to forskellige typer services til at aflevere oplysninger om tjenestemænd – en synkron service, som giver mulighed for at aflevere oplysninger for én tjenestemand og få et svar på afleveringen tilbage synkront, samt en asynkron service, der giver mulighed for at aflevere oplysninger om flere tjenestemænd på én gang og efterfølgende forespørge på status for indlæsningen via et andet servicekald.

Servicen LoenIndberetningWSX udstiller i alt følgende service-operationer:

- *indberet* – synkron indberetning af oplysningerne for én tjenestemand
- *indberetBatch* – asynkron indberetning af oplysninger på flere tjenestemænd på en gang
- *slet* – synkron sletning af oplysninger for én tjenestemand – anvendes typisk til at slette tidligere indberettede oplysninger, som viser sig at være forkerte
- *sletBatch* – asynkron sletning af oplysninger for flere tjenestemænd.
- *status* – forespørgsel på status for tidligere kald af typen indberetBatch og sletBatch.

(de to slet-operationer sletter tjenestemænd fra PENSAB).
De to typer services – synkrone og asynkrone – giver to forskellige kalde-scenarier:

Synkront

Lønsystemet ønsker at aflevere oplysninger om en enkelt tjenestemand og dennes perioder til PENSAB. Lønsystemet kalder den synkrone service-operation 'indberet' med en PENSABLoenIndberetningStruktur (se afsnit 4 for en nærmere forklaring af data-modellen) og modtager – efter endt processering i PENSAB - et svar i form af en PENSABLoenIndberetningResponsStruktur bl.a. indeholdende en statusindikator, der fortæller om indberetningen gik godt. Hvis indberetningen gik godt, er oplysningerne korrekt afleveret i

PENSAB. I modsat fald vil svaret fra PENSAB indeholde en eller flere fejlmeddelelser, som beskriver fejlen.

Asynkront

Lønsystemet ønsker at aflevere oplysninger om flere tjenestemænd på én gang og vælger derfor at kalde service-operationen 'indberetBatch' med en PENSABLoenIndberetningBatchStruktur'. I modsætning til PENSABLoenIndberetningStruktur, kan denne struktur indeholde flere PENSABLoenIndberetningTransaktionStruktur-elementer hver indeholdende oplysninger om en tjenestemand. Da der potentielt kan afleveres mange oplysninger i et sådant kald, vil processeringen af disse oplysninger kunne tage så lang tid, at det ikke er hensigtsmæssigt at den kaldende part skal vente. Denne operation vil derfor blot gemme de indsendte oplysninger og processere dem senere. Operationen vil derfor returnere status MODTAGET sammen med en entydig identifikator PENSABLoenIndberetningIdentifikator.

Lønsystemet kan nu anvende denne PENSABLoenIndberetningIdentifikator ifbm. efterfølgende forespørgsler på status på indberetningens status. Dette foregår via kald til operationen 'status'.

4. Datamodel

Forespørgsler og svar hhv. til og fra PENSAB lønservices er defineret ved følgende strukturer (to beregnet til at indberette ny/opdatere tjenestemand, to til at slette eksisterende tjenestemand samt én til at hente status):

Operation	Input	Output
indberet	PENSABLoenIndberetningStruktur	PENSABLoenIndberetResponsStruktur
indberetBatch	PENSABLoenIndberetningBatchStruktur	PENSABLoenIndberetResponsStruktur
slet	PENSABLoenIndberetningSletStruktur	PENSABLoenIndberetResponsStruktur
sletBatch	PENSABLoenIndberetningSletBatchStruktur	PENSABLoenIndberetResponsStruktur
status	PENSABLoenIndberetningStatusForespoergselStruktur	PENSABLoenIndberetResponsStruktur

I det følgende gennemgås PENSABLoenIndberetningStruktur og PENSABLoenIndberetningResponsStruktur, som er de centrale strukturer, vha. eksempler (Selve indholdet af de enkelte felter i eksemplerne giver sikkert ikke mening i en konkret indberetning. Eksempler her har udelukkende til formål at illustrere strukturen).

PENSABLoenIndberetningStruktur

```
<?xml version="1.0" encoding="UTF-8"?> <tns:PENSABLoenIndberetningStruktur
                xmlns:cpr="http://rep.oio.dk/cpr.dk/xml/schemas/core/2005/03/1
8/"             xmlns:oes="http://rep.oio.dk/oes.dk/xml/schemas/2006/11/24"
                xmlns:tns="http://www.logica.com/pensab/external"
                xmlns:tns1="http://rep.oio.dk/oes.dk/xml/schemas/2006/06/01"
                xmlns:tns2="http://rep.oio.dk/cvr.dk/xml/schemas/2005/03/22/"
                xmlns:tns3="http://rep.oio.dk/ebxml/xml/schemas/dkcc/2003/02/1
3/"             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="PENSAB_LoenIndberetningStruktur.xsd ">
  <tns:PENSABLoenIndberetningIdentifikator>SLS0123456789</tns:PENSABLoenIndberetningIdentifikator>
  <tns:PENSABLoenIndberetningDato>2001-01-31</tns:PENSABLoenIndberetningDato>
  <tns:PENSABLoenIndberetningTransaktionStruktur>
    <tns:PENSABLoenIndberetningTransaktionIdentifikator>SLS0123456789-
01</tns:PENSABLoenIndberetningTransaktionIdentifikator>
    <oes:ExtendedPersonCivilRegistrationIdentifierStructure>
      <cpr:PersonCivilRegistrationIdentifier>0000000000</cpr:PersonCivilRegistrationIdentifier>
    </oes:ExtendedPersonCivilRegistrationIdentifierStructure>
    <tns3:PersonSurnameName>Hansen</tns3:PersonSurnameName>
    <tns3:PersonGivenName>Hans</tns3:PersonGivenName>
    <tns:PENSABPensionLoenrammeIdentifikator>51</tns:PENSABPensionLoenrammeIdentifikator>
    <tns:PENSABPensionSkalaTrinIdentifikator>12</tns:PENSABPensionSkalaTrinIdentifikator>
    <tns:PENSABDelregnskabIdentifikator>1601010</tns:PENSABDelregnskabIdentifikator>
    <tns:PENSABDelregnskabTekst>Ombudsmanden</tns:PENSABDelregnskabTekst>
    <tns:PENSABAnsaettelsesomraadeIdentifikator>9</tns:PENSABAnsaettelsesomraadeIdentifikator>
    <tns:PENSABAnsaettelsesomraadeTekst>Statens Arkiver</tns:PENSABAnsaettelsesomraadeTekst>
<tns:PENSABLoenPeriodeStruktur>
      <tns2:CVRnumberIdentifier>63890812</tns2:CVRnumberIdentifier>
      <tns2:ProductionUnitIdentifier>638908120000</tns2:ProductionUnitIdentifier>
      <tns:PENSABIkrafttraedelseDato>2001-01-01</tns:PENSABIkrafttraedelseDato>
      <tns:PENSABPersonalekategoriIdentifikator>22</tns:PENSABPersonalekategoriIdentifikator>
      <tns:PENSABSkalaTrinIdentifikator>51</tns:PENSABSkalaTrinIdentifikator>
      <tns:PENSABStatusIdentifikator>0</tns:PENSABStatusIdentifikator>
      <tns:PENSABBeskaeftigelsesgradRate>1.0</tns:PENSABBeskaeftigelsesgradRate>
    </tns:PENSABLoenPeriodeStruktur>
  </tns:PENSABLoenIndberetningTransaktionStruktur>
</tns:PENSABLoenIndberetningStruktur>
```

Figur 1

Figur 1 viser et eksempel på den informationsstruktur, som kan indberettes via operationen 'indberet'. Læg mærke til brugen af PENSABLoenIndberetningIdentifikator og PENSABLoenIndberetningTransaktionIdentifikator – indholdet af disse bestemmes af lønsystemet og bruges til at referere hhv. en konkret indberetning og en specifik transaktion i en indberetning ifbm. logning og fejlfinding i PENSAB, samt i forbindelse med indikation af fejl ifbm. indlæsningen.

Det anbefales derfor på det kraftigste, at PENSABLoenIndberetningIdentifikator er entydigt indenfor det kaldende lønsystem og at PENSABLoenIndberetningTransaktionIdentifikator er entydig givet en konkret PENSABLoenIndberetningIdentifikator. Dermed kan en eventuel fejlmeddelelse, som måtte blive returneret, entydigt referere en given transaktion i en given indberetning. I ovenstående tilfælde er PENSABLoenIndberetningTransaktionIdentifikator lig PENSABLoenIndberetningIdentifikator efterstillet med et løbenummer. Dette er ikke et krav.

PENSABLoenIndberetningBatchStruktur er helt analog til ovenstående, men kan blot indeholde flere instanser af PENSABLoenIndberetningTransaktionStruktur.

Alle operationer returnerer samme struktur – PENSABLoenIndberetningResponsStruktur – denne struktur har til formål at formidle status og eventuelle informationer om fejl tilbage til lønsystemet.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:PENSABLoenIndberetningResponsStruktur xmlns:tns="http://www.logica.com/pensab/external"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="PENSAB_LoenIndberetningResponsStruktur.xsd ">
 <tns:PENSABLoenIndberetningIdentifikator>SLS0123456789</tns:PENSABLoenIndberetningIdentifikator>
  <tns:PENSABLoenIndberetningStatusIndikator>OK</tns:PENSABLoenIndberetningStatusIndikator>
  <tns:PENSABLoenIndberetningMeddelelseStruktur>
    <tns:MeddelelseType>INFO</tns:MeddelelseType>
    <tns:MeddelelseKode>PEN0642</tns:MeddelelseKode>
    <tns:MeddelelseTekst>Alt gik fint</tns:MeddelelseTekst>
  </tns:PENSABLoenIndberetningMeddelelseStruktur>
  <tns:PENSABLoenIndberetningTransaktionResponsStruktur>
    <tns:PENSABLoenIndberetningTransaktionIdentifikator>SLS0123456789-
01</tns:PENSABLoenIndberetningTransaktionIdentifikator>
<tns:PENSABLoenIndberetningTransaktionStatusIndikator>OK</tns:PENSABLoenIndberetningTransaktionStatusIndik
ator>
    <tns:PENSABLoenIndberetningMeddelelseStruktur>
      <tns:MeddelelseType>INFO</tns:MeddelelseType>
      <tns:MeddelelseKode>PEN0642</tns:MeddelelseKode>
      <tns:MeddelelseTekst>Alt gik fint</tns:MeddelelseTekst>
    </tns:PENSABLoenIndberetningMeddelelseStruktur>
  </tns:PENSABLoenIndberetningTransaktionResponsStruktur>
</tns:PENSABLoenIndberetningResponsStruktur>
```

Figur 2

Figur 2 illustrerer et fiktivt svar på en tidligere afsendt PENSABLoenIndberetningStruktur. Strukturen indeholder en PENSABLoenIndberetningIdentifikator hvis indhold refererer den tilsvarende identifikator i det kald, som ovenstående er et svar på.
PENSABLoenIndberetningStatusIndikator indeholder det overordnede svar på om processering af indberetning er gået godt eller ej – hvis denne status ikke er OK, er ingen af de indberettede oplysninger indlæst i PENSAB.
Der returneres 0 til mange meddelelser af typen INFO, ADVARSEL eller FEJL. Hvis PENSABLoenIndberetningStatus ikke er OK, vil der altid blive returneret mindst én meddelelse af typen FEJL, der forklarer fejlårsagen. Hvis alle indeholdte transaktioner er indlæst uden bemærkninger, returneres ingen meddelelser.

Tilsvarende leveres der – på transaktionsniveau – en status og 0 til mange meddelelser.

5. Sikkerhedsmodel

Sikkerhedsmodellen omfatter både kryptering og signering. Kryptering anvendes på transportniveau i form af SSL for at forhindre uvedkommende adgang til de data, som transporteres til og fra PENSABs lønservices.
Til authentikering og signering anvendes X.509-baserede funktionscertifikater udstedt af DANID. Dele af requestet til services signeres for at sikre at PENSAB med sikkerhed kan identificere den kaldende part.

Der anvendes følgende principper/standarder

- WS Security 1.1 X.509 Token Policy med 'asymmetric binding'
- Signering af message body
- Signering af timestamp
- Signering af binary security token

I praksis betyder dette, at de pågældende services forventer requests med følgende WSsecurity elementer fra figur 3:

```
<wsse:Security
  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
  S:mustUnderstand="1">
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    Id="Signature-2">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod
        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
      <ds:Reference URI="#Timestamp-1">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <ds:DigestValue />
      </ds:Reference>
      <ds:Reference URI="#id-3">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <ds:DigestValue />
      </ds:Reference>
      <ds:Reference URI="#CertId-482C7293AB5DDD244C13291403817411">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <ds:DigestValue />
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue />
    <ds:KeyInfo Id="KeyId-482C7293AB5DDD244C13291403817512">
      <wsse:SecurityTokenReference
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
        wsu:Id="STRId-482C7293AB5DDD244C13291403817513"
        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
        <wsse:KeyIdentifier
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary"
          ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#ThumbprintSHA1" />
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </ds:Signature>
  <wsse:BinarySecurityToken
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary"
    ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
    wsu:Id="CertId-482C7293AB5DDD244C13291403817411"
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" />
  <wsu:Timestamp
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    wsu:Id="Timestamp-1">
    <wsu:Created />
    <wsu:Expires />
  </wsu:Timestamp>
</wsse:Security>
```

Figur 3

Responset fra services vil indeholde samme WS-security elementer inklusiv en SignatureConfirmation værdi, som vil være signaturen fra det pågældende request.

6. Konstruktion af klient

I dette afsnit gennemgås hvordan en stand-alone klient til PENSAB web services kan konstrueres i Java.

1. Rekvirér en interface-pakke hos undertegnede indeholdende WSDL- og XSD-filer, som beskriver interfacet til de udstillede web services. Pakken indeholder ydermere PENSABs offentlige certifikat, som benyttes til signering af svaret fra PENSABs lønservices. WSDLfilerne vil desuden indeholde de specifikke end-points, som services er udstillet på. Pakken indeholder desuden to eksempel-klienter, som begge er Java-baserede. Den ene klient er Oracle WebLogic-specifik og benytter specifikke features derfra til håndteringen af sikkerhedsmodellen, hvilket gør håndteringen af denne meget simpel. Den anden klient er baseret på WSS4J og kræver lidt flere kodelinier til håndteringen af sikkerhedsmodellen, men er til gengæld mere uafhængig af hvilken Java-platform den kaldende part måtte anvende. Begge klienter gennemgås i større detalje i de kommende afsnit.
2. Kontakt undertegnede med henblik for at få registreret din eksterne IP-adresse på vores positiv-liste. Positiv-listen repræsenterer de IP-adresser, som lukkes igennem vores firewall og ind til PENSAB lønservices.
3. Rekvirér et funktionscertifikat hos DANID gennem din Lokale Registrerings Autoritet (LRA) – angiv et funktionsnavn eksempelvis PENSAB el.lign. Din LRA kan nu bestille funktionscertifikat hos DANID – dette funktionscertifikat vil være underordnet din virksomheds virksomhedscertifikat og vil dermed indeholde virksomhedens CVR-nummer. Når funktionscertifikatet er udstedt kan det hentes hos DANID via et link, som din LRA vil give dig. Når du henter funktionscertifikatet, skal du undervejs angive en passphrase til den private key, som udgør den ene (private) del af certifikatet. Det anbefales at gemme funktionscertifikatet som et Java Key Store, når det downloades hvis det (som i dette eksempel) skal anvendes i Java.

Generer JAX-WS artefakter vha. Apache Ant og wsimport

Wsimport er et værktøj, som er pakket med i JAX-WS RI. Wsimport anvendes hovedsagelig til at generere service endpoint interfaces (SEI) samt mapning mellem Java klasser og XSD skema typer (via JAXB) på baggrund af WSDL definitioner. Følg figur 4 for at anvende wsimport fra Ant:

```
<taskdef name="wsimport" classname="com.sun.tools.ws.ant.WsImport">
  <classpath>
    <fileset dir="path/to/jaxb" includes="*.jar" />
    <fileset dir="path/to/jaxws" includes="*.jar" />
  </classpath>
</taskdef>

<target name="generate-artifacts">
  <wsimport debug="true"
            destdir="src"
            wsdl="http://location/serviceName?WSDL"
            xendorsed="true"
            verbose="true"
            keep="true">
  </wsimport>
</target>
```

Figur 4

Scriptet fra figur 4 kan anvendes både til den Weblogic-specifikke klient og den ikke-Weblogicspecifikke klient.

Kald web service via SEI

Den pågældende web service kan kaldes via det JAX-WS genererede SEI samt javax.xml.ws.Service.

```
package com.logica.pensab;

import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
import your.generated.SEI;

public class WsseClient {
  public static void main(String[] args) throws Exception {
    // you want to change these values
    URL wsdlLocation = new URL("http://location/serviceName?WSDL");
    QName serviceName = new QName("namespaceURI", "serviceName");

    // create port to service
    Service service = Service.create(wsdlLocation, serviceName);
    SEI port = service.getPort(SEI.class);

    // WS-security will be configured here…

    // invoke method on port
    port.method();
  }

}
```

Figur 5

Dette gælder både for den Weblogic-specifikke og den ikke-Weblogic-specifikke klient, se figur 5.

Håndter WS-security for request og response

Weblogic-specifik

Den Weblogic–specifikke klient konfigurerer både udgående og indgående WS-security ved hjælp af en CredentialProvider. Figur 6 viser, hvordan klientens private nøgle udpeges til signering, og hvordan serverens offentlige certifikat udpeges til validering af responset. Vær opmærksom på, at sti til servercertifikat og konfiguration af CredentialProvider naturligvis skal udskiftes med gældende værdier.

```
package com.logica.pensab.external;

import java.io.FileInputStream;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import javax.xml.ws.BindingProvider;
import weblogic.security.SSL.TrustManager;
import weblogic.wsee.security.bst.ClientBSTCredentialProvider;
import weblogic.xml.crypto.wss.WSSecurityContext;

public class WsseClient {
  public static void main(String[] args) throws Exception {
    . . .
    // port is created
    . . .

    // grap the server's certificate
    FileInputStream serverCertificate = new FileInputStream("path/to/server-certificate.cer");
    CertificateFactory cf = CertificateFactory.getInstance("X.509");
    X509Certificate cert = (X509Certificate)cf.generateCertificate(serverCertificate);
    serverCertificate.close();

    // create CredentialProvider to sign the request
    ClientBSTCredentialProvider cp = new ClientBSTCredentialProvider("path/to/client-keystore.jks",
                                                                     "keystorePassword",
                                                                     "privateKeyAlias",
                                                                     "privateKeyPassword");
    // tell the CredentialProvider to verify
    // the response signature with this certificate
    cp.setServerCertificate(cert);
    List credProviders = new ArrayList();
    credProviders.add(cp);

    // attach CredentialProvider and TrustManager to requst context.
    // the TrustManager ensures that the server certificate is verified.
    Map<String, Object> requestContext = ((BindingProvider)port).getRequestContext();
    requestContext.put(WSSecurityContext.CREDENTIAL_PROVIDER_LIST, credProviders);
    requestContext.put(WSSecurityContext.TRUST_MANAGER, new TrustManager() {
      public boolean certificateCallback(X509Certificate[] chain, int validateErr) {
        return true;
      }
    });


    // this call will throw a weblogic.xml.crypto.wss.WSSecurityException,
    // if the response signatture cannot be validated.
    port.method();
  }
}
```

Figur 6

Ikke-Weblogic-specifik

Den ikke-Weblogic-specifikke klient håndterer WS-security ved hjælp af en javax.xml.ws.handler.Handler. Ved at hæfte en Handler på SEI porten bliver vi i stand til at berige requestet med WS-security samt at verificere WS-security på responset. Figur 7 viser, hvordan en handler hæftes på SEI porten.

```
package com.logica.pensab;

import java.util.List;
import javax.xml.ws.Binding;
import javax.xml.ws.BindingProvider;
import javax.xml.ws.handler.Handler;
import com.logica.pensab.WsseHandler;

public class WsseClient {
  public static void main(String[] args) throws Exception {
    . . .
    // port is created
    . . .

    // append security handler to handler chain
    Binding binding = ((BindingProvider) port).getBinding();
    List<Handler> handlerChain = binding.getHandlerChain();
    handlerChain.add(new WsseHandler());

    // this call is required to configure the binding instance with the new chain
    binding.setHandlerChain(handlerChain);

    port.indberet(type);
  }
}
```

Figur 7

Opbygning og signering af web service requestet i Handler klassen er implementeret ift. figur 8 og 9. Vær opmærksom på, at konfiguration af keystore samt udpegning af privat nøgle til signering udskiftes med gældende værdier.

```
package com.logica.pensab;

import java.security.KeyStore;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.cert.X509Certificate;
import java.util.HashSet;
import java.util.Properties;
import java.util.Set;
import java.util.Vector;

import javax.security.cert.CertificateEncodingException;
import javax.xml.bind.DatatypeConverter;
import javax.xml.namespace.QName;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.handler.MessageContext;
import javax.xml.ws.handler.soap.SOAPHandler;
import javax.xml.ws.handler.soap.SOAPMessageContext;

import org.apache.ws.security.WSConstants;
import org.apache.ws.security.WSEncryptionPart;
import org.apache.ws.security.WSSecurityEngine;
import org.apache.ws.security.WSSecurityException;
import org.apache.ws.security.components.crypto.Crypto;
import org.apache.ws.security.components.crypto.CryptoFactory;
import org.apache.ws.security.message.WSSecHeader;
import org.apache.ws.security.message.WSSecSignature;
import org.apache.ws.security.message.WSSecTimestamp;
import org.apache.ws.security.util.XMLUtils;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class WsseHandler implements SOAPHandler<SOAPMessageContext> {

  public boolean handleOutboundMessage(SOAPMessageContext smc) {
    try {
      Properties props = new Properties();
      props.setProperty("org.apache.ws.security.crypto.merlin.file",
                        "path/to/truststore.jks");
      props.setProperty("org.apache.ws.security.crypto.merlin.keystore.password",
                        "truststorePassword");
      props.setProperty("org.apache.ws.security.crypto.provider",
                        "org.apache.ws.security.components.crypto.Merlin");
      props.setProperty("org.apache.ws.security.crypto.merlin.keystore.type",
                        "JKS");
      props.setProperty("private.key.alias",
                        "privateKeyAlias");
      props.setProperty("private.key.password",
                        "privateKeyPassword");
      Crypto crypto = CryptoFactory.getInstance(props);

      KeyStore.PrivateKeyEntry keyEntry = (KeyStore.PrivateKeyEntry) crypto.getKeyStore().getEntry(
                                          props.getProperty("private.key.alias"),
                                            new KeyStore.PasswordProtection(
                                              props.getProperty("private.key.password").toCharArray()));
      X509Certificate certificate = (X509Certificate) keyEntry.getCertificate();

      SOAPPart soapPart = smc.getMessage().getSOAPPart();
      SOAPEnvelope env = soapPart.getEnvelope();
      Document doc = env.getOwnerDocument();

      // insert WSSecUsernameToken to SOAPEnvelope's WSSecHeader
      WSSecHeader secHeader = new WSSecHeader();
      secHeader.insertSecurityHeader(doc);

      // create Timestamp
      WSSecTimestamp secTimestamp = new WSSecTimestamp();
      secTimestamp.setTimeToLive(300);
      secTimestamp.prepare(doc);
      // prepend the Timestamp to the security header
      secTimestamp.prependToHeader(secHeader);
```

Figur 8

```
      // create Signature and choose Binary Security Token as key identifier
      WSSecSignature secSignature = new WSSecSignature();
      secSignature.setKeyIdentifierType(WSConstants.BST_DIRECT_REFERENCE);
      // give credentials to signing key
      secSignature.setUserInfo(props.getProperty("private.key.alias"),
                               props.getProperty("private.key.password"));

      // prepare so that we can reference the signature element before signing the lot
      secSignature.prepare(doc, crypto, secHeader);

      // prepend the Binary Security Token to the security header
      secSignature.prependBSTElementToHeader(secHeader);
      secSignature.prependToHeader(secHeader);

      // create new KeyIdentifier with ThumbPrintSHA1 value.
      Element secKeyIdentifier = doc.createElementNS(WSConstants.WSSE_NS, "KeyIdentifier");
      secKeyIdentifier.setPrefix("wsse");
      secKeyIdentifier.setAttribute("EncodingType", "http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary");
      secKeyIdentifier.setAttribute("ValueType", "http://docs.oasis-open.org/wss/oasis-wss-soap-message-
security-1.1#ThumbprintSHA1");
      secKeyIdentifier.setTextContent(getThumbPrint(certificate));

      // replace existing KeyIdentifier with new one
      Element securityTokenReference =
(Element) secHeader.getSecurityHeader().getElementsByTagNameNS(WSConstants.WSSE_NS,
                                                               "SecurityTokenReference").item(0);
      Element defaultReference =
(Element) securityTokenReference.getElementsByTagNameNS(WSConstants.WSSE_NS,
                                                        "Reference").item(0);
      securityTokenReference.replaceChild(secKeyIdentifier, defaultReference);

      // point out which elements should be signed
      Vector<WSEncryptionPart> wsEncryptionParts = new Vector<WSEncryptionPart>();
      // add Timestamp for signing
      wsEncryptionParts.add(new WSEncryptionPart(secTimestamp.getId()));
      // add Body for signing
      wsEncryptionParts.add(new WSEncryptionPart("Body", WSConstants.URI_SOAP11_ENV, "Content"));
      // add BinarySecurityToken for signing
      wsEncryptionParts.add(new WSEncryptionPart("BinarySecurityToken", WSConstants.WSSE_NS, "Element"));
      secSignature.addReferencesToSign(wsEncryptionParts, secHeader);

      // do the actual signing
      secSignature.computeSignature();
      System.out.println(XMLUtils.PrettyDocumentToString(doc));
    } catch (Exception e) {
      // catch-all!
      e.printStackTrace();
      return false;
    }
    return true;
  }

  private String getThumbPrint(X509Certificate cert) throws Exception {
    MessageDigest md = MessageDigest.getInstance("SHA-1");
    byte[] der = cert.getEncoded();
    md.update(der);
    byte[] digest = md.digest();
    return DatatypeConverter.printBase64Binary(digest);
  }

  . . .
  // some other methods
  . . .

}
```

Figur 9

Verificering af web service responset er implementeret i Handler klassen ift. figur 10. Vær opmærksom på, at konfiguration af keystore naturligvis skal udskiftes med gældende værdier.

```
package com.logica.pensab;

import java.security.KeyStore;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.cert.X509Certificate;
import java.util.HashSet;
import java.util.Properties;
import java.util.Set;
import java.util.Vector;

import javax.security.cert.CertificateEncodingException;
import javax.xml.bind.DatatypeConverter;
import javax.xml.namespace.QName;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.handler.MessageContext;
import javax.xml.ws.handler.soap.SOAPHandler;
import javax.xml.ws.handler.soap.SOAPMessageContext;

import org.apache.ws.security.WSConstants;
import org.apache.ws.security.WSEncryptionPart;
import org.apache.ws.security.WSSecurityEngine;
import org.apache.ws.security.WSSecurityException;
import org.apache.ws.security.components.crypto.Crypto;
import org.apache.ws.security.components.crypto.CryptoFactory;
import org.apache.ws.security.message.WSSecHeader;
import org.apache.ws.security.message.WSSecSignature;
import org.apache.ws.security.message.WSSecTimestamp;
import org.apache.ws.security.util.XMLUtils;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class WsseHandler implements SOAPHandler<SOAPMessageContext> {

  . . .
  // some other method
  . . .

  public boolean handleInboundMessage(SOAPMessageContext smc) {
    Properties props = new Properties();
    props.setProperty("org.apache.ws.security.crypto.merlin.file",
                      "path/to/truststore.jks");
    props.setProperty("org.apache.ws.security.crypto.merlin.keystore.password",
                      "truststorePassword");
    props.setProperty("org.apache.ws.security.crypto.provider",
                      "org.apache.ws.security.components.crypto.Merlin");
    props.setProperty("org.apache.ws.security.crypto.merlin.keystore.type",
                      "JKS");
    Crypto crypto = CryptoFactory.getInstance(props);

    Document doc = smc.getMessage().getSOAPPart();
    System.out.println(XMLUtils.PrettyDocumentToString(doc));
    WSSecurityEngine secEngine = new WSSecurityEngine();
    try {
      // this call will throw a WSSecurityException,
      // if the envelope cannot be validated.
      secEngine.processSecurityHeader(doc, null, null, crypto);
    } catch (WSSecurityException e) {
      e.printStackTrace();
      return false;
    }
    return true;
  }
}
```

Figur 10

Krævede binaries

- WSS4J 1.5.12 (http://www.apache.org/dyn/closer.cgi/ws/wss4j/1_5_12/wss4j-bin-1.5.12.zip)
- JAXWS 2.2 (http://jax-ws.java.net/2.2/)

Følgende gælder kun for den ikke-Weblogic-specifikke klient:

- JAXB 2.2.3 (http://jaxb.java.net/2.2.3/)
- Apache Commons logging (http://commons.apache.org/logging/download_logging.cgi)

7. Services med reduceret sikkerhedsmodel

Det har vist sig problematisk at implementere den beskrevne sikkerhedsmodel fuldt og helt på CICS-platformen i det den signerings- og krypteringskomponent, som anvendes her, ikke understøtter eller kommer til at understøtte signering af header elementer.

Der er derfor implementeret et andet sæt services, som semantisk fungerer 100% som de allerede beskrevne services, blot med en reduceret sikkerhedsmodel.

Klientdelen til disse services implementeres med udgangspunkt i LoenIndberetningNHSWSX.wsdl i stedet for LoenIndberetningWSX.wsdl.
NHS står i denne forbindelse for No Header Signing (NHS) og hentyder til, at to af kravene til requestet, som sendes til servicen, ikke valideres:

1. Signering af Timestamp - der skal ikke længere indsættes Timestamp og som heraf skal dette naturligvis heller ikke signeres.
2. Signeringen af certifikatet / Binary Security Token
3. SHA1-Thumbprint reference til Binary Security Token

Security-delen af et request efter NHS-modellen kunne se ud som følger - bemærk, at forskellen ift. det tidligere beskrevne indhold af security-delen under den fulde sikkerhedsmodel er, at dsig:SignedInfo nu kun indeholder signaturen for Body og ikke for elementerne Timestamp og BST.

8. Returkoder fra PENSAB

Lønservices kan returnere følgende beskeder:

Kode	Type	Tekst
PEN1049	FEJL	Certifikatet udpeger ikke et lønsystem. Kontakt leverandøren.
PEN1053	FEJL	Det er ikke muligt at oprette den nye tjenestemand
PEN1054	FEJL	Det er ikke muligt at opdatere tjenestemanden
PEN1055	FEJL	Det er ikke muligt at oprette ny periode for tjenestemand
PEN1056	FEJL	Tjenestemanden findes ikke i PENSAB og kan derfor ikke slettes
PEN1057	FEJL	Kan ikke slette tjenestemanden
PEN1067	FEJL	Den angivne statuskode findes ikke i PENSAB
PEN1068	FEJL	Den angivne personalekategori findes ikke i PENSAB
PEN1069	FEJL	Det angivne CVR-nr. og P-nr. hører ikke sammen
PEN1070	INFO	Tjenestemandens produktionsenhed er ændret til den nyeste periodes produktionsenhed
PEN1071	INFO	CVR-nr. er ændret for perioden og derfor har PENSAB nulstillet seniorbeskæftigelsesgraden
PEN1072	FEJL	Der er ingen indberetning med denne PENSABLoenIndberetningIdentifikator fra en leverandør med dette certifikat
PEN1102	FEJL	Det angivne ansættelsesområde og evt. delregnskab findes ikke i PENSAB
PEN1150	INFO	Der er oprettet en tjenestemand med CPR-nr. <cpr-nr>
PEN1151	INFO	Skalatrin er ændret fra <trin> til <trin>
PEN1152	INFO	Beskæftigelsesgrad er ændret fra <beskgrad> til <beskgrad>
PEN1153	INFO	Ansættelsesområdet er ændret fra <Ans.kode> til <Ans.kode>
PEN1154	INFO	Virksomheden er ændret fra <CVR-nr> til <CVR-nr>
PEN2001	FEJL	Ukendt fejl ved indberetning - kontakt CGI - tom.sorensen@cgi.com
PEN2002	FEJL	Personen med dette CPR-nr findes ikke i PENSAB og der er ikke rettighed til at oprette, bruger: <brugerid>
PEN2003	FEJL	Der er ikke rettighed til at slette tjenestemanden fra PENSAB, bruger: <brugerid>
PEN2704	INFO	Statuskode ændret fra x til y for periode med start d