Written by René Hjortskov Nielsen @ SKAT Copenhagen, 21 - 08 - 2012

Introduction

This guideline will demonstrate how to connect a Windows Communication Foundation (WCF) .NET c# client with a typical B2B endpoint in the Danish Tax department.

There are four steps that are needed in order to create the web service client.

1. Know your tools
2. Web service contract handling
3. Business wiring
4. Web service security

Content

Know your tools

Visual Studio 2008 and above inserts custom xml in the SOAP Header. This breaks the security processing so this has to be removed.

Furthermore, it is vital to be able to see messages preferable on the wire, but otherwise as close as possible.

Visual Studio fix

To remove 'VsDebuggerCausalityData' you need stop de Visual Studio Diagnostic for WCF using this command:

VS 2008 -> c:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE>vsdiag_regwcf.exe -u

VS 2010 -> c:\Program Files\Microsoft Visual Studio 10.0\Common7\IDE>vsdiag_regwcf.exe -u

Debug logging

Visual Studio got a nice tool to browse and format logs. This tool is called SvcTraceViewer.exe, which can be found here: C:\Program Files\Microsoft SDKs\Windows\v7.0A\bin

The following app.config create a message log and a very detailed log.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.diagnostics>
    <!--
    Logging is important when debugging web service security protocols
    -->
    <sources>
      <source name="System.ServiceModel.MessageLogging">
        <listeners>
          <add name="messages"
          type="System.Diagnostics.XmlWriterTraceListener"
          initializeData="c:\temp\messages.svclog " />
        </listeners>
      </source>
      <source name="System.ServiceModel" switchValue="All">
        <listeners>
          <add name="identity" type="System.Diagnostics.XmlWriterTraceListener"
initializeData="c:\temp\identity.xml" />
        </listeners>
      </source>
      <source name="System.ServiceModel.MessageLogging" switchValue="All">
        <listeners>
          <add type="System.Diagnostics.DefaultTraceListener" name="Default" >
            <filter type="" />
          </add>
          <add name="ServiceModelMessageLoggingListener">
            <filter type="" />
          </add>
        </listeners>
      </source>
      </sources>
    <sharedListeners>
      <add initializeData="C:\temp\PersonKontrolOplysningHent.svclog"
          type="System.Diagnostics.XmlWriterTraceListener, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089"
          name="ServiceModelMessageLoggingListener" traceOutputOptions="Timestamp">
        <filter type="" />
      </add>
      <add initializeData="C:\temp\PersonKontrolOplysningHent.svclog"
                      type="System.Diagnostics.XmlWriterTraceListener, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
                      name="ServiceModelTraceListener" traceOutputOptions="Timestamp">
        <filter type="" />
      </add>
    </sharedListeners>
    <trace autoflush="true" />
  </system.diagnostics>

  <system.serviceModel>
    <diagnostics>
      <messageLogging
           logEntireMessage="true"
           logMalformedMessages="true"
           logMessagesAtServiceLevel="true"
           logMessagesAtTransportLevel="true"
           maxMessagesToLog="30000"
           maxSizeOfMessageToLog="20000000"/>
    </diagnostics>

  </system.serviceModel>

<!—Disanle this if you are not behind a proxy -->
  <system.net>
    <defaultProxy enabled="true" useDefaultCredentials="true" />
  </system.net>
</configuration>
```

Web service contract handling

It is important to know your way around handling a web service contract. SKAT is very strict about contract-first.

This section describes various normal tasks when working with contracts.

After this lesson you will know how to create the first artifact – the web service stub.

WSDL

The example web service we are going to connect to is called PersonKontrolOplysningHent. The demo version is running at this endpoint:
https://services.extranet.demo.skat.dk/vericert/services/PersonKontrolOplysningHentServicePort

Fetching the contract with some tool, e.g. svcutil, or SOAP UI will just result in errors since the endpoint does not expose the full contract with all references schema files.

Since SKAT enforces contract-first we need to fetch the contract from some yellow page repository. SKAT typical expose the wsdls for B2B on our external wiki site:

http://85.81.229.78/services/demo/PersonKontrolOplysningHent/wsdl/PersonKontrolOplysningHent.wsdl

This contract is, however, an abstract contract meaning there is no service binding. Furthermore, the contract does not expose the Web Service Policy to be applied, neither does the demo endpoint.

Downloading the contract can be a cumbersome job. Luckily, the author of this guide has written a contract tool helping downloading the abstract contract as a local copy, with relative paths. The tool can be found here: http://85.81.229.78/contract

Just point the tool to the wsdl file for an online contract and the result is an offline copy you can use directly in your development and control version system.

There can be some situations to remedy, however.

One situation could be a missing service binding, which is why it is an abstract contract. Copy the missing binding from the demo endpoint in order to remedy this.

Typically the demo endpoint also includes a Web service security wsdl section (WSP). This is not the case for our demo service, though.

SKAT used the same WSP for all B2B services and therefore it can just be copied from this document:

```
  <wsp:UsingPolicy s1:Required="true"/>
  <wsp:Policy s0:Id="SkatGatewayAuthentication" >
    <wssp:Identity>
      <wssp:SupportedTokens>
        <wssp:SecurityToken TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3">
        </wssp:SecurityToken>
      </wssp:SupportedTokens>
    </wssp:Identity>
    <wssp:Integrity>
      <wssp:SignatureAlgorithm URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <wssp:CanonicalizationAlgorithm URI="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      <wssp:Target>
        <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <wssp:MessageParts
Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part">wsp:Body()</wssp:MessageParts>
      </wssp:Target>
      <wssp:Target>
        <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <wssp:MessageParts
Dialect="http://www.bea.com/wls90/security/policy/wsee#part">wls:SecurityHeader(wsu:Timestam
p)</wssp:MessageParts>
      </wssp:Target>
      <wssp:SupportedTokens>
        <wssp:SecurityToken IncludeInMessage="true" TokenType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
      </wssp:SupportedTokens>
    </wssp:Integrity>
  </wsp:Policy>
  <wsp:Policy s0:Id="Sign.xml">
```

```
    <wssp:Integrity>
      <wssp:SignatureAlgorithm URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <wssp:CanonicalizationAlgorithm URI="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      <wssp:Target>
        <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <wssp:MessageParts Dialect="http://www.bea.com/wls90/security/policy/wsee#part">
        wls:SystemHeaders()
        </wssp:MessageParts>
      </wssp:Target>
      <wssp:Target>
        <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <wssp:MessageParts Dialect="http://www.bea.com/wls90/security/policy/wsee#part">
        wls:SecurityHeader(wsu:Timestamp)
        </wssp:MessageParts>
      </wssp:Target>
      <wssp:Target>
        <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <wssp:MessageParts Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part">
      wsp:Body()
        </wssp:MessageParts>
      </wssp:Target>
      <wssp:SupportedTokens>
        <wssp:SecurityToken IncludeInMessage="true" TokenType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3">

<wssp:TokenIssuer>CN=sktpens01web02.csc.dk,O=Skat,C=DK,CN=B2BGateway,O=Skat,C=DK,CN=sktpens0
1web02.csc.dk,O=Skat,C=DK,CN=sktpens01web02.csc.dk,O=Skat,C=DK,CN=TDC OCES
CA,O=TDC,C=DK,</wssp:TokenIssuer>
        </wssp:SecurityToken>
      </wssp:SupportedTokens>
    </wssp:Integrity>
    <wssp:MessageAge Age="60"/>
  </wsp:Policy>
```

The definition line has to get the following namespaces declared:

```
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wssp="http://schemas.xmlsoap.org/ws/2002/12/secext" xmlns:s1="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:s0=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
```

```
Please note that this web service policy is for signing only. SKAT also uses policies for
encryption sometimes, but the endpoints inline the wsp so there is no need to put it here.

Now we got a concrete offline contract ready to begin the actual development with.
```

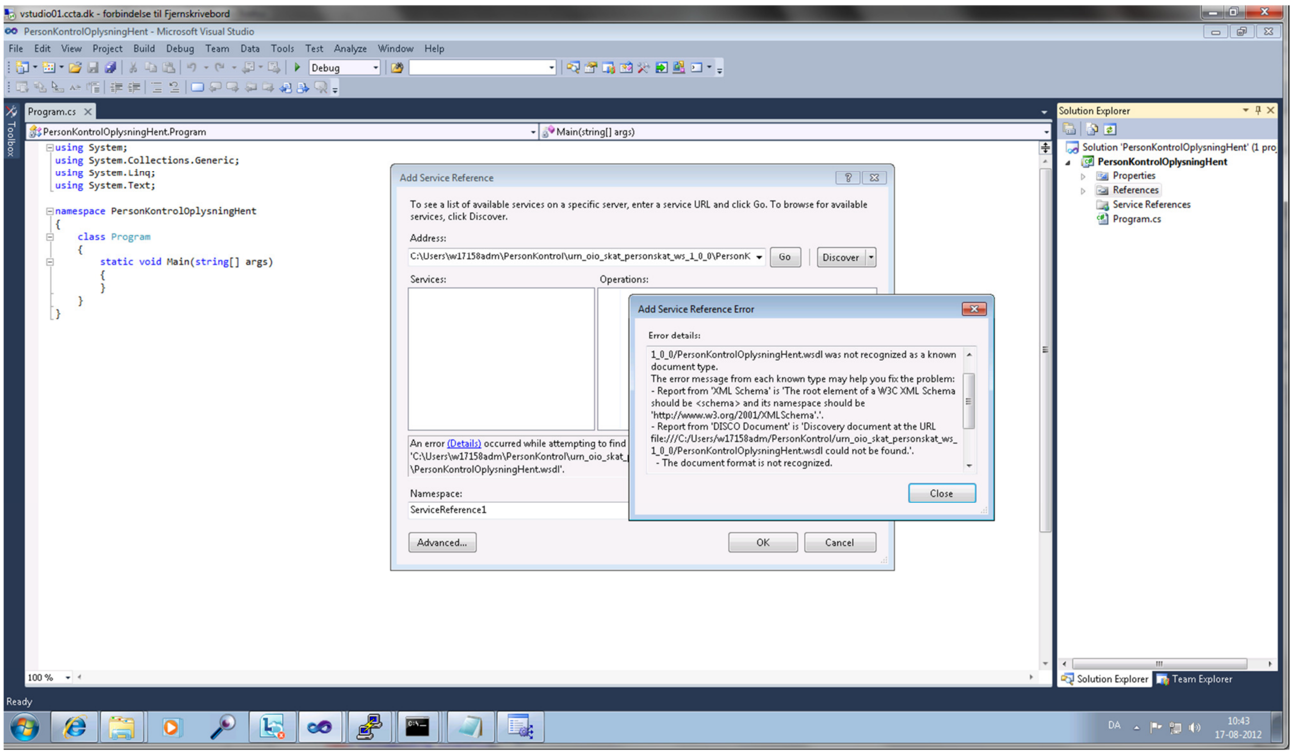
Svcutil

This paragraph is about generating the web service client stub. Microsoft has not put too much effort into the handling of contract except their own mex endpoint discovery mechanism which is basically used exclusively in WCF to WCF communication.

SKAT uses open standards in order to enable all vendors / tools to participate in the process of building web service clients.

The older Visual Studio 2005 used Webservice Security Enhancement 3.0 (WSE 3.0) which in Visual Studio 2008 and above is substituted with Windows Communication Foundation (WCF). This also implies a move from .NET 2.0 to .NET 3.5 and above. Another important change is the stub type, which is called Web service reference in .NET 2.0. This way of building web service is still possible in VS2010, however, this guide looks into WCF and thus we also need to use the new stub type Service reference.

VS2010 has a wizard for adding both stub types, however, these tools lacks support for handling nested contracts with multiple files, i.e. wsdl and schema files. The only valid way to remedy this is to generate the stub with svcutil.exe. The error is shown in the picture below:


vstudio01.ccta.dk - forbindelse til Fjernskrivebord
PersonKontrolOplysningHent - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Test Analyze Window Help
Debug
Program.cs
PersonKontrolOplysningHent.Program
Main(string[] args)
Solution Explorer
Solution 'PersonKontrolOplysningHent' (1 pro
PersonKontrolOplysningHent
Properties
References
Service References
Program.cs
Solution Explorer
Team Explorer
100 %
Ready
DA
10:43
17-08-2012

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace PersonKontrolOplysningHent
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Add Service Reference

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address: C:\Users\w17158adm\PersonKontrol\urn_oio_skat_personskat_ws_1_0_0\PersonK

Services:

Operations:

An error (Details) occurred while attempting to find 'C:\Users\w17158adm\PersonKontrol\urn_oio_skat_
\PersonKontrolOplysningHent.wsdl'.

Namespace: ServiceReference1

Add Service Reference Error

Error details:

```
1_0_0/PersonKontrolOplysningHent.wsdl was not recognized as a known
document type.
The error message from each known type may help you fix the problem:
- Report from 'XML Schema' is 'The root element of a W3C XML Schema
should be <schema> and its namespace should be
'http://www.w3.org/2001/XMLSchema'.'.
- Report from 'DISCO Document' is 'Discovery document at the URL
file:///C:/Users/w17158adm/PersonKontrol/urn_oio_skat_personskat_ws_
1_0_0/PersonKontrolOplysningHent.wsdl could not be found.'.
  - The document format is not recognized.
```

Figure 1 – Error while trying to perform Add service reference in Visual Studio 2010

It would be nice if the tool supported just pointing out the top wsdl file, but in the absence of this feature we need to supply all files on the command line.

For the demo endpoint we can execute the following – please don’t be worried about the security warning, which we will fix in a moment.

```
cd C:\Program Files\Microsoft SDKs\Windows\v7.0A\bin
```

```
svcutil.exe /language:cs /out:c:\Users\w17158adm\generatedProxy.cs
/config:c:\Users\w17158adm\app.config
pk\urn_oio_skat_personskat_ws_1_0_0\PersonKontrolOplysningHent.wsdl
pk\urn_oio_skat_personskat_ws_1_0_0\PersonKontrolOplysningHent_IType.xsd
pk\urn_oio_skat_personskat_ws_1_0_0\PersonKontrolOplysningHent_OType.xsd
pk\urn_oio_skat_personskat_ws_1_0_0\PersonKontrolOplysningHentInterface.xsd
pk\urn_oio_skat_personskat_ws_1_0_0\RapportOplysningAdresseStrukturType.xsd
pk\urn_oio_skat_personskat_ws_1_0_0\RapportOplysningBeloebStrukturType.xsd
pk\urn_oio_skat_personskat_ws_1_0_0\RapportOplysningGruppeStrukturType.xsd
pk\urn_oio_skat_personskat_ws_1_0_0\RapportOplysningIdentifikationStrukturType.xsd
```

pk\urn_oio_skat_personskat_ws_1_0_0\RapportOplysningKvantumStrukturType.xsd
pk\urn_oio_skat_personskat_ws_1_0_0\RapportOplysningMarkeringStrukturType.xsd
pk\urn_oio_skat_personskat_ws_1_0_0\RapportOplysningPeriodeStrukturType.xsd
pk\urn_oio_skat_personskat_ws_1_0_0\RapportOplysningProcentStrukturType.xsd
pk\urn_oio_skat_personskat_ws_1_0_0\RapportOplysningReferenceStrukturType.xsd
pk\urn_oio_skat_personskat_ws_1_0_0\RapportOplysningSekvensStrukturType.xsd
pk\urn_oio_skat_personskat_ws_1_0_0\RapportOplysningStrukturType.xsd
pk\urn_oio_skat_personskat_1_0_0\PostAdresseSjetteLinjeTekst.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningBeloeb.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningBeloebKvalifikatorKode.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningBeloebKvalifikatorTekst.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningGruppeIdentifikator.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningGruppeNavn.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningIdentifikationBoNavn.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningIdentifikationKvalifikatorKode.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningIdentifikationKvalifikatorTekst.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningIdentifikationNavn.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningKvantumEnhedKode.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningKvantumKvalifikatorKode.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningKvantumKvalifikatorTekst.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningKvantumMaal.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningMarkeringKode.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningMarkeringKvalifikatorKode.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningMarkeringKvalifikatorTekst.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningMarkeringTekst.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningNavn.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningNummerIdentifikator.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningPeriodeDato.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningPeriodeKvalifikatorKode.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningPeriodeKvalifikatorTekst.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningPeriodeSlutAarIdentifikator.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningPeriodeStartAarIdentifikator.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningProcent.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningProcentKvalifikatorKode.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningProcentKvalifikatorTekst.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningReferenceKvalifikatorKode.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningReferenceKvalifikatorNavn.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningReferenceTekst.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningSekvensNavn.xsd
pk\urn_oio_skat_personskat_1_0_0\RapportOplysningSekvensNummerIdentifikator.xsd
pk\urn_oio_skat_personskat_1_0_0\SkattecenterKodeIdentifikator.xsd
pk\rep_oio_dk\cpr_dk\xml\schemas\core\2005\05\19\CPR_CompletePostalLabelText.xsd
pk\rep_oio_dk\oib\dato_tid_maal\xml_schema\AarIdentifikator_20080901.xsd

pk\rep_oio_dk\oib\dato_tid_maal\xml_schema\MaanedKode_20080901.xsd
pk\rep_oio_dk\oib\oekonomi_skat\xml_schema\KvartalKode_20090201.xsd
pk\rep_oio_dk\xkom_dk\xml\schemas\2007\09\01\XKOM_DateInterval.xsd
pk\rep_oio_dk\xkom_dk\xml\schemas\2007\09\01\XKOM_DurationMeasure.xsd
pk\rep_oio_dk\cvr_dk\xml\schemas\2005\03\22\CVR_CVRnumberIdentifier.xsd
pk\rep_oio_dk\cpr_dk\xml\schemas\core\2002\06\28\CPR_AuthorityCode.xsd
pk\rep_oio_dk\ebxml\xml\schemas\dkcc\2005\05\19\DKCC_PostalAddressFifthLineText.xsd
pk\rep_oio_dk\ebxml\xml\schemas\dkcc\2005\05\19\DKCC_PostalAddressFirstLineText.xsd
pk\rep_oio_dk\ebxml\xml\schemas\dkcc\2005\05\19\DKCC_PostalAddressFourthLineText.xsd
pk\rep_oio_dk\ebxml\xml\schemas\dkcc\2005\05\19\DKCC_PostalAddressSecondLineText.xsd
pk\rep_oio_dk\ebxml\xml\schemas\dkcc\2005\05\19\DKCC_PostalAddressThirdLineText.xsd
pk\rep_oio_dk\xkom_dk\xml\schemas\2006\09\01\XKOM_StartDate.xsd
pk\rep_oio_dk\xkom_dk\xml\schemas\2007\04\15\XKOM_EndDate.xsd
pk\rep_oio_dk\skat_dk\motor\class\virksomhed\xml\schemas\20080401\SKAT_VirksomhedSENummerIdentifikator.xsd pk\rep_oio_dk\skat_dk\TSE\angivelse\xml\schemas\2006\09\01\SKAT_ValutaKode.xsd
pk\rep_oio_dk\skat_dk\eindkomst\class\alternativadresse\xml\schemas\20071202\SKAT_AlternativAdresseAnvendelseKode.xsd
pk\rep_oio_dk\skat_dk\basis\kontekst\xml\schemas\2006\09\01\SKAT_AdvisIdentifikator.xsd
pk\rep_oio_dk\skat_dk\basis\kontekst\xml\schemas\2006\09\01\SKAT_AdvisStruktur.xsd
pk\rep_oio_dk\skat_dk\basis\kontekst\xml\schemas\2006\09\01\SKAT_AdvisTekst.xsd
pk\rep_oio_dk\skat_dk\basis\kontekst\xml\schemas\2006\09\01\SKAT_FejlIdentifikator.xsd
pk\rep_oio_dk\skat_dk\basis\kontekst\xml\schemas\2006\09\01\SKAT_FejlStruktur.xsd
pk\rep_oio_dk\skat_dk\basis\kontekst\xml\schemas\2006\09\01\SKAT_FejlTekst.xsd
pk\rep_oio_dk\skat_dk\basis\kontekst\xml\schemas\2006\09\01\SKAT_HovedOplysninger.xsd
pk\rep_oio_dk\skat_dk\basis\kontekst\xml\schemas\2006\09\01\SKAT_HovedOplysningerSvar.xsd
pk\rep_oio_dk\skat_dk\basis\kontekst\xml\schemas\2006\09\01\SKAT_ServiceIdentifikator.xsd
pk\rep_oio_dk\skat_dk\basis\kontekst\xml\schemas\2006\09\01\SKAT_SvarStruktur.xsd
pk\rep_oio_dk\skat_dk\basis\kontekst\xml\schemas\2006\09\01\SKAT_TransaktionIdentifikator.xsd
pk\rep_oio_dk\skat_dk\basis\kontekst\xml\schemas\2006\09\01\SKAT_TransaktionTid.xsd
pk\rep_oio_dk\cpr_dk\xml\schemas\core\2005\03\18\CPR_PersonCivilRegistrationIdentifier.xsd
pk\rep_oio_dk\ebxml\xml\schemas\dkcc\2003\02\13\DKCC_CountryIdentificationCode.xsd
pk\rep_oio_dk\itst_dk\xml\schemas\2006\01\17\ITST_PersonName.xsd

Execution of the svcutil yields:

```
Microsoft (R) Service Model Metadata Tool

[Microsoft (R) Windows (R) Communication Foundation, Version 3.0.4506.2152]

Copyright (c) Microsoft Corporation.  All rights reserved.



Warning: Følgende politikantagelser blev ikke importeret:
  XPath:
//wsdl:definitions[@targetNamespace='urn:oio:skat:personskat:ws:1.0.0']/wsdl:binding[@name='PersonKo
ntrolOplysningHentServiceBinding']/wsdl:operation[@name='getPersonKontrolOplysningHent']/ws

dl:input

  Antagelser:

    <wssp:Integrity xmlns:wssp='http://schemas.xmlsoap.org/ws/2002/12/secext'>..</wssp:Integrity>

    <wssp:MessageAge
xmlns:wssp='http://schemas.xmlsoap.org/ws/2002/12/secext'>..</wssp:MessageAge>

    <wssp:Identity xmlns:wssp='http://schemas.xmlsoap.org/ws/2002/12/secext'>..</wssp:Identity>

    <wssp:Integrity xmlns:wssp='http://schemas.xmlsoap.org/ws/2002/12/secext'>..</wssp:Integrity>

Generating files...

c:\Users\w17158adm\generatedProxy.cs

c:\Users\w17158adm\app.config
```

This leaves us with two artifacts describing a web service client configuration file and a proxy c# file:

- App.config
- generatedProxy.cs

Business wiring

Now the time has come to wire the generated web service proxy with some business logic.

This step requires some valid test data.

System owners in SKAT can provide the test data.

The code for our PersonKontrolOplysningHent demo service would look something like this:

```
            // Call the service operations with business document.
            HovedOplysningerType ho = new HovedOplysningerType();
            ho.TransaktionIdentifikator = "518fb9c7-72d4-4726-be8a-80e854846d18";
            ho.TransaktionTid = new DateTime(2012, 3, 15, 9, 0, 45, 950, DateTimeKind.Utc);
            ho.TransaktionTidSpecified = true;
            PersonKontrolOplysningHent_IType doc = new PersonKontrolOplysningHent_IType();
            doc.HovedOplysninger = ho;
            PersonKontrolOplysningHent_ITypePersonAar paar = new
PersonKontrolOplysningHent_ITypePersonAar();
            paar.PersonCivilRegistrationIdentifier = "2201031438";
            paar.AarIdentifikator = "2011";
            doc.PersonAar = paar;
            PersonKontrolOplysningHent_OType resp = null;
            try
            {
                resp = client.getPersonKontrolOplysningHent(doc);
                client.Close();
            }
            catch (FaultException e)
            {
                Console.WriteLine("SOAPFault->");
                Console.WriteLine("Code: "+e.Code);
                Console.WriteLine("Message: " + e.Message);
                Console.WriteLine("Reason: " + e.Reason);
                if (resp != null)
                {
                    Console.WriteLine(resp.HovedOplysningerSvar.SvarStruktur.ToString());
                }
            }
            catch (MessageSecurityException e)
            {
                Console.WriteLine(e.Message);
                p.printError(e);
            }
```

Technical errors are reported as SOAP Faults, and Business Errors and advertisements ought to be reported in the SOAP Response within the `HovedOplysninger.`

Web service Security

Actually, we got a working web service client at this point in time, however, the authentification with certificates is missing before we can retrieve a positive SOAP Response.

System owners in SKAT can provide the demo certificates used. There are two certificates which are important.

1. Public / Private key VOCES/MOCES test certificate (often a .p12 file)
2. Public key test certificate denoting the demo server (often a .cer file)

The latter is optional and typically should only be applied when encryption is needed. However, malformed implementation also requires this key for verifying the SOAP Response server signature.

Now it is time for us to enable signing and/or encryption. This is a bit akward since the svcutil didn't understand our nice little Webservice Security Policy.

Open the generatedProxy.cs file and change the following:

```
[System.ServiceModel.ServiceContractAttribute(Namespace =
"urn:oio:skat:personskat:ws:1.0.0", ConfigurationName =
"PersonKontrolOplysningHentServicePortType")]
public interface PersonKontrolOplysningHentServicePortType

to

// Added manually since WCF does not handle WSE wwebservice policies in the svcutil tool
[System.ServiceModel.ServiceContractAttribute(Namespace =
"urn:oio:skat:personskat:ws:1.0.0", ConfigurationName =
"PersonKontrolOplysningHentServicePortType", ProtectionLevel =
System.Net.Security.ProtectionLevel.Sign
)]
```

```
public interface PersonKontrolOplysningHentServicePortType
```

Visual Studio 2010 now supports the standard security we use in SKAT, but still it has to be added as a so called custom binding.

The next section list the complete demo program.cs. You will have to add the appropriate references and rename a few things, add proper error handling and business wiring to your backend. I will not go into details about how the security, please refer to the section Further reading.

The only security parts you have to adapt are the DnsEndpointIdentity, the client and server certificates. This should suffice for most web service b2b client.

Full program.cs demo listing

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml;
using System.Security;
using System.Security.Cryptography.X509Certificates;
using System.ServiceModel.Description;
using System.ServiceModel;
using System.ServiceModel.Security;
using System.ServiceModel.Channels;
using System.Net;
using System.Net.Security;
using System.ServiceModel.Security.Tokens;

namespace PersonKontrolOplysningHent
{
    /*
     * This is the main program used for calling the webservice PersonKontrolOplysningHent
with an appropriate security context
     *
     * The security is added with a CustomBinding.
     *
     * Values are hardcoded, e.g. clientcertificate, servicecertificate, server ssl
certificate dns, and endpoint
     *
     * @author René Hjortskov Nielsen, SKAT
     * @version 1.0
     */
    class Program
    {
        static void Main(string[] args)
        {
            Program p = new Program();
            Console.WriteLine("Default encoding: "+Encoding.Default);

            // Ignore resolving SSL server certificate
            // Furthermore, we use One-Way SSL, menaning that the client should not identify
itself at the transport level via SSL certificate
            ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(IgnoreCertificateErrorHandler);

            //Step 1: Create an instance of the WCF proxy.
            // We use HTTPS
            // Client certificate with serial number 40 37 d9 b3
            DnsEndpointIdentity dnsendpointidentity = new DnsEndpointIdentity("SKAT - SKAT
virsomhedscert test");
            EndpointAddress endpointAddress = new EndpointAddress(new
Uri("https://services.extranet.demo.skat.dk/vericert/services/PersonKontrolOplysningHentServ
icePort"), dnsendpointidentity, new AddressHeaderCollection());
            PersonKontrolOplysningHentServicePortTypeClient client = new
PersonKontrolOplysningHentServicePortTypeClient(CreateCustomBinding(), endpointAddress);
            //var factory = new
ChannelFactory<PersonKontrolOplysningHentServicePortType>(CreateCustomBinding(),
endpointAddress);

            // Add Message inspector
            client.Endpoint.Behaviors.Add(new MyBehavior());
            //factory.Endpoint.Behaviors.Add(new MyBehavior());

            //Step 2: Attach client certificate used in securing SOAP message at the message
level - Only integrity is used by means of signing

//factory.Credentials.ClientCertificate.SetCertificate(StoreLocation.CurrentUser,
StoreName.My, X509FindType.FindBySerialNumber, "40 37 d9 b3");

//factory.Credentials.ServiceCertificate.Authentication.CertificateValidationMode =
System.ServiceModel.Security.X509CertificateValidationMode.None;

//factory.Credentials.ServiceCertificate.SetDefaultCertificate(StoreLocation.CurrentUser,
StoreName.My, X509FindType.FindBySerialNumber, "40 37 db 39");
            //var wcfClientChannel = factory.CreateChannel();
            // Certificate used for signing

client.ClientCredentials.ClientCertificate.SetCertificate(StoreLocation.CurrentUser,
StoreName.My, X509FindType.FindBySerialNumber, "40 37 d9 b3");

client.ClientCredentials.ServiceCertificate.Authentication.CertificateValidationMode =
System.ServiceModel.Security.X509CertificateValidationMode.None;
            // Certificate used for encryption and signing response validation

client.ClientCredentials.ServiceCertificate.SetDefaultCertificate(StoreLocation.CurrentUser,
StoreName.My, X509FindType.FindBySerialNumber, "40 37 db 39");

            // Step 3: Call the service operations with business document.
            HovedOplysningerType ho = new HovedOplysningerType();
            ho.TransaktionIdentifikator = "518fb9c7-72d4-4726-be8a-80e854846d18";
            ho.TransaktionTid = new DateTime(2012, 3, 15, 9, 0, 45, 950, DateTimeKind.Utc);
            ho.TransaktionTidSpecified = true;
            PersonKontrolOplysningHent_IType doc = new PersonKontrolOplysningHent_IType();
            doc.HovedOplysninger = ho;
            PersonKontrolOplysningHent_ITypePersonAar paar = new
PersonKontrolOplysningHent_ITypePersonAar();
            paar.PersonCivilRegistrationIdentifier = "2201031438";
            paar.AarIdentifikator = "2011";
            doc.PersonAar = paar;
            PersonKontrolOplysningHent_OType resp = null;
            try
            {
                //getPersonKontrolOplysningHentRequest request = new
getPersonKontrolOplysningHentRequest(doc);
                //getPersonKontrolOplysningHentResponse response =
wcfClientChannel.getPersonKontrolOplysningHent(request);//
```

```
                resp = client.getPersonKontrolOplysningHent(doc);
                // Step 3: Closing the client gracefully closes the connection and cleans up 
resources.
                client.Close();
            }
            catch (FaultException e)
            {
                Console.WriteLine("SOAPFault->");
                Console.WriteLine("Code: "+e.Code);
                Console.WriteLine("Message: " + e.Message);
                Console.WriteLine("Reason: " + e.Reason);
                if (resp != null)
                {
                    Console.WriteLine(resp.HovedOplysningerSvar.SvarStruktur.ToString());
                }
            }
            catch (MessageSecurityException e)
            {
                Console.WriteLine(e.Message);
                p.printError(e);
            }

            Console.ReadKey();
        }

        private void printError(Exception e)
        {
            Console.WriteLine("############################################################ 
ERROR #######################################################");
            Console.WriteLine("" + e.ToString());
            Console.WriteLine("############################################################ 
ERROR #######################################################");
        }

        /**
         * For demo system we accept all SSL certificates, no matter what
         */
        public static bool IgnoreCertificateErrorHandler(object sender,
              X509Certificate certificate, X509Chain chain, SslPolicyErrors sslPolicyErrors)
        {
            Console.WriteLine("IgnoreCertificateErrorHandler");
            return true;
        }

        /**
         * SKAT uses:
         *  SOAP 1.1
         *  UTF-8
         *  HTTP with one-way SSL (Only server needs to be validated)
         *  Signing and/or Encryption of SOAP message parts, i.e. TimeStamp, Body, 
BinarySecurityToken
         *  Policy http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3
         */
        private static CustomBinding CreateCustomBinding()
        {
            var res = new CustomBinding();

            //http://msdn.microsoft.com/en-us/library/aa738565.aspx
            AsymmetricSecurityBindingElement sec =
(AsymmetricSecurityBindingElement)SecurityBindingElement.CreateMutualCertificateBindingEleme
nt(

MessageSecurityVersion.WSSecurity10WSTrustFebruary2005WSSecureConversationFebruary2005WSSecu
rityPolicy11BasicSecurityProfile10);
            sec.MessageProtectionOrder = MessageProtectionOrder.SignBeforeEncrypt;

            /**
             * http://support.microsoft.com/kb/971493
             * A hotfix that enables WCF to send secured messages and to receive unsecured
responses,
             * and to send unsecured messages and to receive secured responses,
             * is available for the .NET Framework 3.5 SP1
             */
            sec.EnableUnsecuredResponse = false;

            /**
             * SKAT does not expects responses to be neither signed nor encrypted
             */
            sec.AllowSerializedSigningTokenOnReply = true;
            sec.RequireSignatureConfirmation = false;
            sec.AllowInsecureTransport = true;
            sec.IncludeTimestamp = true;

            res.Elements.Clear();
            res.Elements.Add(sec);
            sec.SetKeyDerivation(false);
            sec.DefaultAlgorithmSuite = SecurityAlgorithmSuite.TripleDesRsa15;
            /**
             * Timestamp shall be written in the end
             */
            sec.SecurityHeaderLayout = SecurityHeaderLayout.LaxTimestampLast;
            sec.EndpointSupportingTokenParameters.Signed.Add(new
X509SecurityTokenParameters(X509KeyIdentifierClauseType.SubjectKeyIdentifier));

            res.Elements.Add(new TextMessageEncodingBindingElement()
            {
                MessageVersion = MessageVersion.CreateVersion(EnvelopeVersion.Soap11,
AddressingVersion.None), // MessageVersion.Soap11,
                WriteEncoding = new System.Text.UTF8Encoding()
            });

            /**
             * HTTP with one-way SSL (i.e only server needs to be validated) binding
             */
            HttpsTransportBindingElement httpsbinding = new HttpsTransportBindingElement();
            httpsbinding.RequireClientCertificate = false;
            httpsbinding.AuthenticationScheme = AuthenticationSchemes.Anonymous;
            httpsbinding.MaxReceivedMessageSize = 1024 * 1024;
            res.Elements.Add(httpsbinding);

            return res;
        }

    }
}
```

Further reading

Tuturial http://msdn.microsoft.com/en-us/library/ms733133.aspx

WCF support for WSE 3.0 http://msdn.microsoft.com/en-us/library/ms730299

Custom certificates in signing and encryption http://msdn.microsoft.com/en-us/library/ms729856

WCF Message logging http://msdn.microsoft.com/en-us/library/ms730064.aspx

Migrating from WSE 3.0 to WCF http://msdn.microsoft.com/en-us/library/ms732008.aspx

WCF App.config http://msdn.microsoft.com/en-us/library/ms733932.aspx

```
Good description of signing and encryption
 http://webservices20.blogspot.dk/2010/09/wcf-server-signs-response-with.html
```