Chapter 5
WS Security in GlassFish ESB

Michael.Czapski@sun.com, Revision 0.3.0, October, 2009

In this document I explore the effects of selected web services security policies on SOAP message exchange in the GlassFish ESB v2.1.

To provide early access I intend to release revisions of this document as significant new sections become available.

This is a work-in-progress document.

Rev 0.1: Content

- Assumptions and Notes
- Person Service XML Schema and WSDL Interface
- Common XML Project
- PersonSvc BPEL Module
- PersonCli BPEL Module
- JBI-based Person Service – Plain End-to-End
- JBI-based Person Service – SSL with Server-side Authentication

Rev 0.2: Additional Content

- JBI-based Person Service – SSL with Mutual Authentication (broken)
- EJB-based Person Service – No security
- EJB-based Person Service – SSL with Server-side Authentication

Rev 0.3: Additional Content

- EJB-based Person Service – SSL with Mutual Authentication
- JBI-based Person Service - Exploring WS-Addressing

Table of Content

5.1 Chapter Content

GlassFish ESB v2.1 uses the Metro stack for web services standards support.

This chapter explores selected methods of applying security to the channel over which SOAP messages are exchanged and the SOAP messages themselves, using a basic BPEL 2.0-based invoker and provider set.

A pair of projects, an invoker and a provider, are used to provide the logic. Composite Applications are used to apply different variants of security policies. There will be one pair of Composite Applications for each security policy to demonstrate that security is a non-functional requirement and to show how security policy and application logic can be separated such that change in one does not require change to the other.

The following security policies are explored:

- None
- HTP BC Channel Security - SSL / TLS with Server-side Authentication
- HTTP BC Channel Security - SSL / TLS with Mutual Authentication
- EJB Channel Security - SSL / TLS with Server-side Authentication
- EJB Channel Security - SSL / TLS with Mutual Authentication
- JBI-based Service – Exploring WS-Addressing
-
- Message Encryption
-

For each variant an end-to-end solution will be built and exercised. Server.log traces from both sides will be inspected and discussed as necessary to clarify what is happening during the process.

5.2 Assumptions and Notes

To explore different options for securing web services in GlassFish ESB v2.1 it is necessary to obtain and install the GlassFish ESB v2.1 software. Distribution, as at September 2009, can be downloaded from https://open-esb.dev.java.net/Downloads.html. Since installation of GlassFish

ESB is adequately documents, see http://wiki.open-esb.java.net/Wiki.jsp?page=UsingTheGlassFishESBInstallationGUI, installation instructions are not repeated here. It is assumed that the GlassFish ESB v2.1 installation exists and is functional. This also assumes that the NeBeans 6.5.1 IDE, distributed as part of GlassFish ESB, is installed and operational. Issues have been reported with different version of the JDK 6. GlassFish ESB installation used for examples developed in this chapter uses JDK 1.6.0_16.

Exploration of channel security, while possible with a single installation of GlassFish ESB, will be easier if two instances of GlassFish ESB, on two different hosts, are available. If not, it will be hard for the reader to follow SSL Handshake log messages and figure out which are produced by the invoker and which are produced by the provider. Examples in this chapter will use two separate hosts for these projects. Typically, the server-side implementation will be deployed to a host whose fully qualified domain name (FQDN) is orad1.ssc and client-side implementation will be deployed to a host whose FQDN is mcz02.aus.sun.com.

It is assumed that the NetBeans IDE and one instance of the GlassFish Application Server are co-hosted on the same machine. Each time I use the expression “the local instance of the GlassFish” I mean the instance of the GlassFish which is resident on the same host as the NetBeans IDE used for development. If this is not the environment you use adjust as required.

5.3 *Person Service XML Schema and WSDL Interface*

A basic web service provider, which we will develop and use to explore web services security, will accept a request with a person identifier and will return a small set of person details for the selected person. This is a request/reply service. Since data returned by the service is not of importance we will not bother with details such as searching a database for person details. We will simply hardcode the response.

The request and response messages will conform to the XML Schema shown in Listing 5.3.1.

Listing 5.3.1 *Person XML Schema*

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/schema/Person"
    xmlns:tns="http://xml.netbeans.org/schema/Person"
    elementFormDefault="qualified"
    >

    <xsd:element name="PersonReq">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="PersonID" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="PersonRes">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="PersonID" type="xsd:string"/>
                <xsd:element name="FamilyName" type="xsd:string"/>
                <xsd:element name="MiddleInitials"
```

```
                             type="xsd:string" minOccurs="0"/>
                <xsd:element name="GivenName" type="xsd:string"/>
                <xsd:element name="Gender" type="xsd:string" minOccurs="0"/>
                <xsd:element name="AddressDetails" minOccurs="0">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="StreetAddress"
                                         type="xsd:string" minOccurs="0"/>
                            <xsd:element name="CityTown"
                                         type="xsd:string" minOccurs="0"/>
                            <xsd:element name="PostCode"
                                         type="xsd:string" minOccurs="0"/>
                            <xsd:element name="StateProvince"
                                         type="xsd:string" minOccurs="0"/>
                            <xsd:element name="Country"
                                         type="xsd:string" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="CreditCardDetails" minOccurs="0">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="CardType"
                                         type="xsd:string" minOccurs="0"/>
                            <xsd:element name="CardNumber"
                                         type="xsd:string" minOccurs="0"/>
                            <xsd:element name="ExpiryDate"
                                         type="xsd:string" minOccurs="0"/>
                            <xsd:element name="SecurityCode"
                                         type="xsd:string" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>

            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="PersonFlt">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="PersonID" type="xsd:string"/>
                <xsd:element name="FaultDetail" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

</xsd:schema>
```

The service interface will conform to the WSDL interface document shown in Listing 5.3.2. This service uses messages defined in the Person XML Schema shown in Listing 5.3.1.

Listing 5.3.2 ***PersonAbsSvc WSDL Interface Document***

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
    name="PersonAbsSvc"
    targetNamespace="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc"
    xmlns:ns="http://xml.netbeans.org/schema/Person"
    xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype">
    <types>
        <xsd:schema
            targetNamespace="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc">
            <xsd:import
                namespace="http://xml.netbeans.org/schema/Person"
                schemaLocation="Person.xsd"/>
        </xsd:schema>
    </types>
    <message name="getPersonDetailsRequest">
        <part name="msgPersonDetailsReq" element="ns:PersonReq"/>
    </message>
    <message name="getPersonDetailsResponse">
        <part name="msgPersonDetailsRes" element="ns:PersonRes"/>
    </message>
    <message name="getPersonDetailsFault">
        <part name="msgPersonDetailsFlt" element="ns:PersonFlt"/>
    </message>
    <portType name="PersonAbsSvcPortType">
        <operation name="getPersonDetails">
            <input name="input1" message="tns:getPersonDetailsRequest"/>
            <output name="output1" message="tns:getPersonDetailsResponse"/>
            <fault name="fault1" message="tns:getPersonDetailsFault"/>
        </operation>
    </portType>
    <plnk:partnerLinkType name="PersonAbsSvc">
        <plnk:role name="PersonAbsSvcPortTypeRole"
                   portType="tns:PersonAbsSvcPortType"/>
    </plnk:partnerLinkType>
</definitions>
```

Note that this WSDL document only defines the Abstract part of the interface. We will add concrete part for each project variant we will explore. This WSDL definition will be named PersonAbsSvc.

To save the effort the client implementation will be exposed as a web service and will be triggered using a SoapUI web service testing project. The interface definition, TriggerCon, is shown in Listing 5.3.3.

Listing 5.3.3 ***TriggerCon WSDL Interface Document***

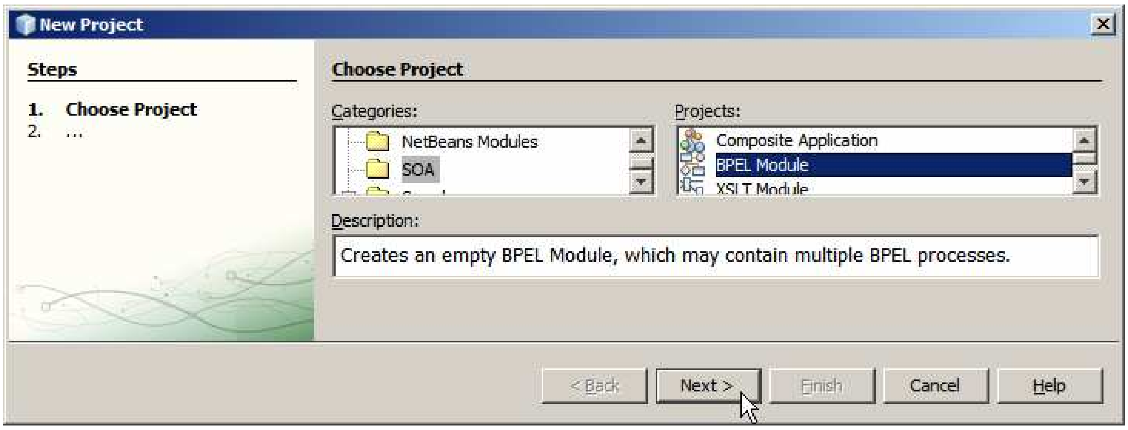
```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
    name="TriggerCon"
    targetNamespace="http://j2ee.netbeans.org/wsdl/CommonXML/TriggerCon"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://j2ee.netbeans.org/wsdl/CommonXML/TriggerCon"
    xmlns:ns="http://xml.netbeans.org/schema/Person"
    xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
    <types>
        <xsd:schema
            targetNamespace="http://j2ee.netbeans.org/wsdl/CommonXML/TriggerCon">
            <xsd:import
                namespace="http://xml.netbeans.org/schema/Person"
                schemaLocation="Person.xsd"/>
        </xsd:schema>
    </types>
    <message name="triggerPersonRequest">
        <part name="msgPersonDetailsReq" element="ns:PersonReq"/>
    </message>
    <message name="triggerPersonResponse">
        <part name="msgPersonDetailsRes" element="ns:PersonRes"/>
    </message>
    <message name="triggerPersonFault">
        <part name="msgPersonDetailsFlt" element="ns:PersonFlt"/>
    </message>
    <portType name="TriggerConPortType">
        <operation name="triggerPerson">
            <input name="input1" message="tns:triggerPersonRequest"/>
            <output name="output1" message="tns:triggerPersonResponse"/>
            <fault name="fault1" message="tns:triggerPersonFault"/>
        </operation>
    </portType>
    <binding name="TriggerConBinding" type="tns:TriggerConPortType">
        <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="triggerPerson">
            <soap:operation/>
            <input name="input1">
                <soap:body use="literal"/>
            </input>
            <output name="output1">
                <soap:body use="literal"/>
            </output>
            <fault name="fault1">
                <soap:fault use="literal" name="fault1"/>
            </fault>
        </operation>
    </binding>
    <service name="TriggerConService">
        <port name="TriggerConPort" binding="tns:TriggerConBinding">
            <soap:address location=
            "http://localhost:${HttpDefaultPort}/TriggerConService/TriggerConPort"/>
        </port>
    </service>
    <plnk:partnerLinkType name="TriggerCon">
        <plnk:role name="TriggerConPortTypeRole" portType="tns:TriggerConPortType"/>
    </plnk:partnerLinkType>
</definitions>
```

This is a concrete interface. Remember to change the FQDN of the host in the WSDL to that of your host.

5.4 Common XML Project

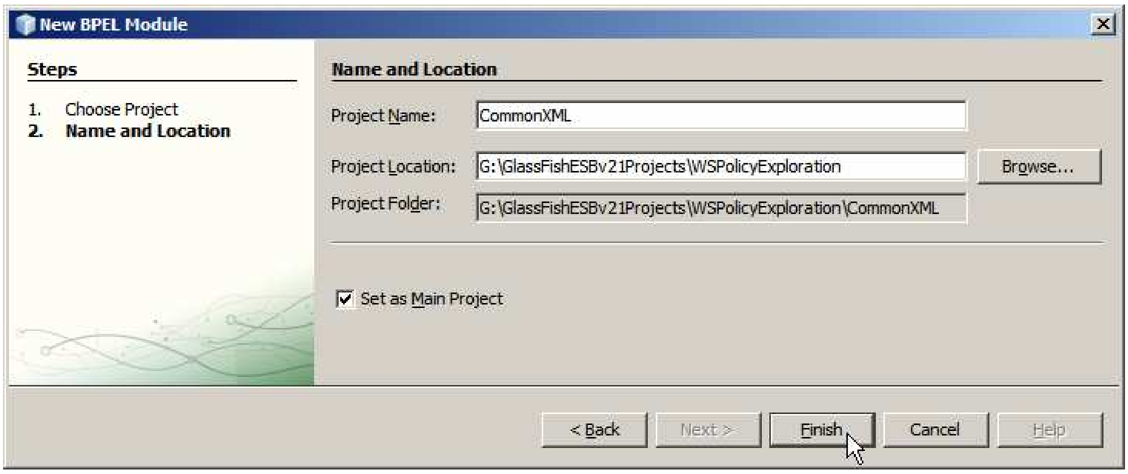
Let’s create a Project Group to contain projects developed in this chapter. This project group will be called WSPolicyExploration and will contain the common XML artifacts, the Person XML Schema, the PersonAbsSvc WSDL and the TriggerCon WSDL.

Let’s create a New Project … -> SOA -> BPEL Module, named CommonXML. Figures 5.4.1 and 5.4.2 show important steps in the process. This BPEL project is just a convenient location for the XML documents we will be creating.



Description: Creates an empty BPEL Module, which may contain multiple BPEL processes.

Figure 5.4.1 *Create BPEL Module*



Project Name: CommonXML

Project Location: G:\GlassFishESBv21Projects\WSPolicyExploration

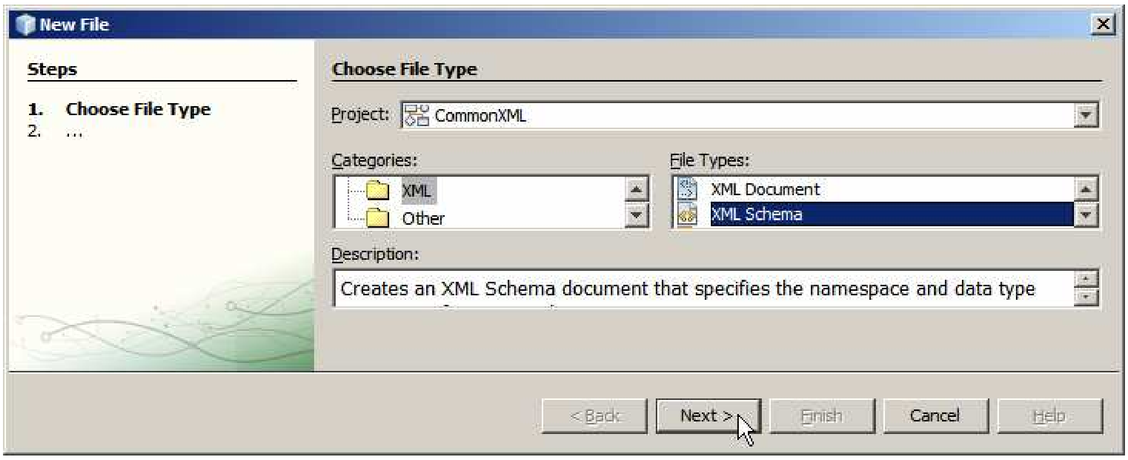
Project Folder: G:\GlassFishESBv21Projects\WSPolicyExploration\CommonXML

☑ Set as Main Project

Figure 5.4.2 *Naming the project and specifying location*

The skeleton BPEL process model, commonXML.bpel, can be deleted since it will not be used.

Right-click the project name and choose New … -> Other … -> XML -> XML Schema, Figure 5.4.3.



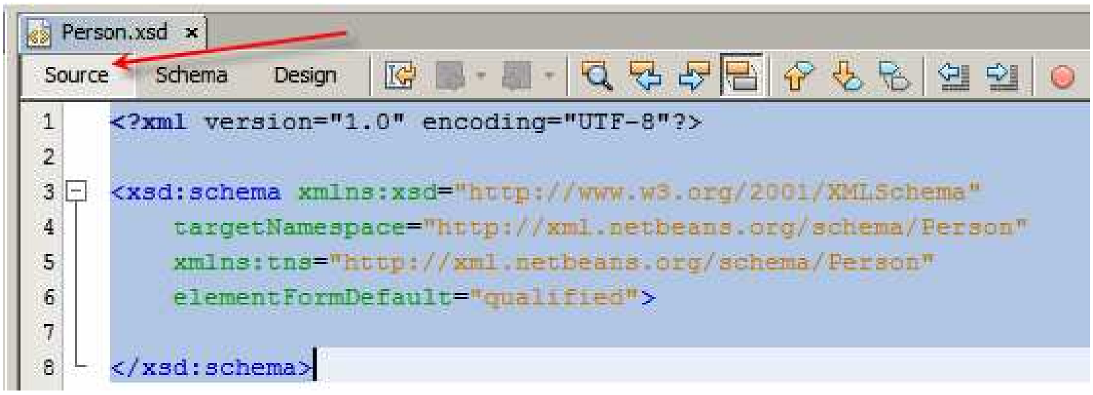
Project: CommonXML

Description: Creates an XML Schema document that specifies the namespace and data type

Figure 5.4.3 *Create a new XML Schema*

Name this schema Person and click Finish.

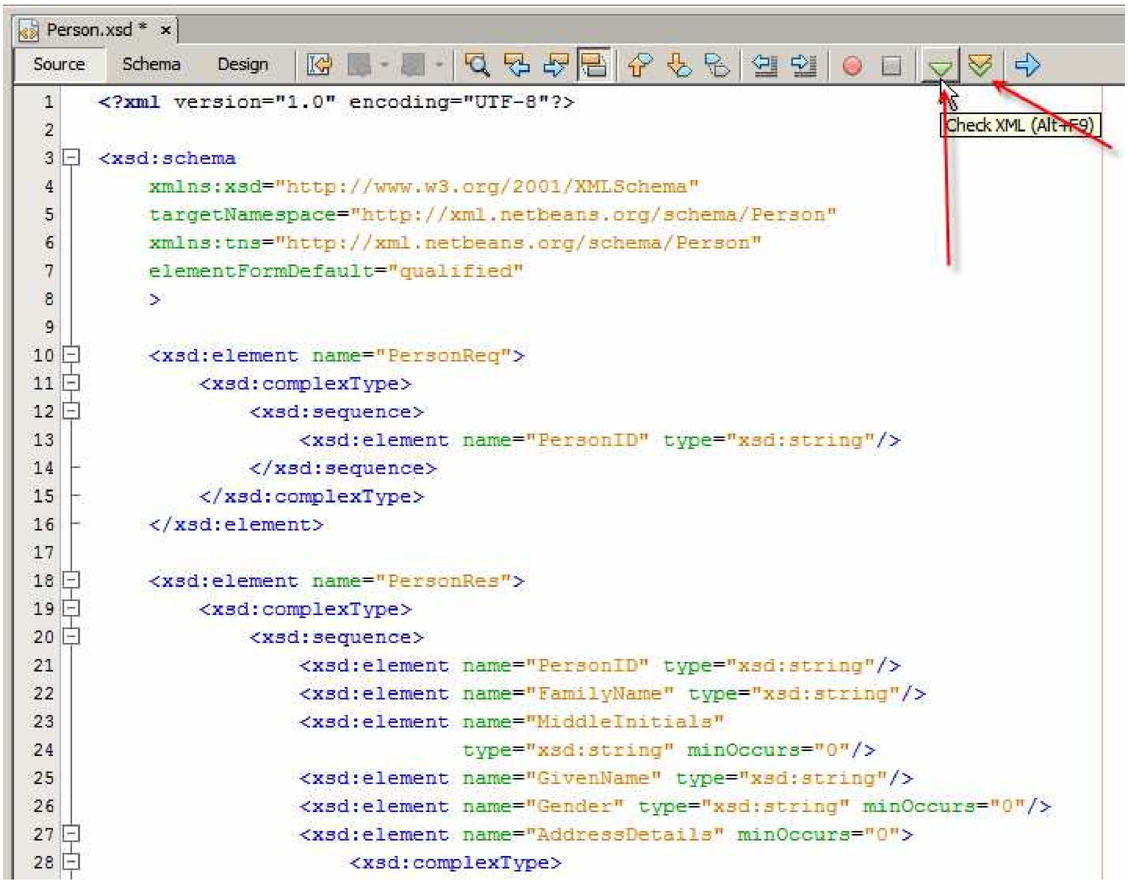
When the new XML Schema document opens in the editor window, switch to the Source view and select all the text. Figure 5.4.4 illustrates this.



```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/schema/Person"
    xmlns:tns="http://xml.netbeans.org/schema/Person"
    elementFormDefault="qualified">

</xsd:schema>
```

Figure 5.4.4 *Select the content of the new XML Schema document*

Paste XML Schema text from Listing 5.3.1 in place of the selected text. Check XML and Validate XML, illustrated in Figure 5.4.5, and resolve any issues that might have arisen.



```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/schema/Person"
    xmlns:tns="http://xml.netbeans.org/schema/Person"
    elementFormDefault="qualified"
    >

    <xsd:element name="PersonReq">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="PersonID" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="PersonRes">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="PersonID" type="xsd:string"/>
                <xsd:element name="FamilyName" type="xsd:string"/>
                <xsd:element name="MiddleInitials"
                             type="xsd:string" minOccurs="0"/>
                <xsd:element name="GivenName" type="xsd:string"/>
                <xsd:element name="Gender" type="xsd:string" minOccurs="0"/>
                <xsd:element name="AddressDetails" minOccurs="0">
                    <xsd:complexType>
```

Figure 5.4.5 *Check and Validate*

Save the new schema.

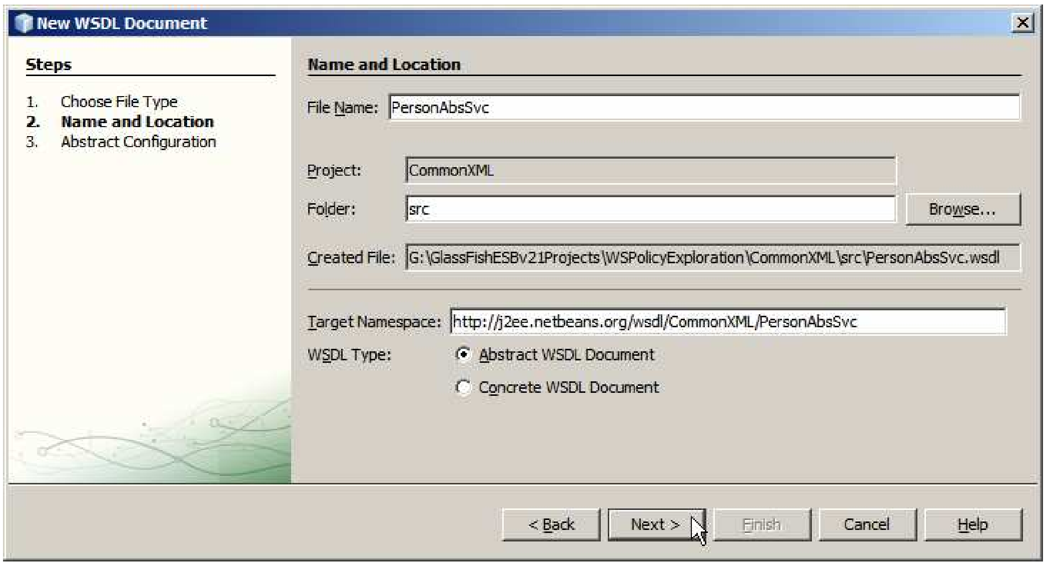
Create a New … -> WSDL Document …, named PersonAbsSvc. This will be an Abstract WSDL Document. Figure 5.4.6 illustrates the first dialogue box involved in the process.

Click Next to advance to the next panel.

Change the name of the operation to getPersonDetails. Change names of Input and Output message parts to msgPersonDetailsReq and msgPersonDetaislRes respectively. Add a new Fault message part and name it msgPersonDetaislFlt. Figure 5.4.7 illustrated the dialogue box at this point in the process.

For each message part click the small ellipsis button and choose appropriate element from the Person XML Schema. For msgPersonDetailsReq it will be PersonReq, for msgPersonDetailsRes it will be PersonRes and for msgPersonDetailsFlt it will be PersonFlt. Figure 5.4.8 illustrates a step in this process. Figure 5.4.9 shows the dialogue box with all parts with correct elements.

Click Finish to complete the wizard. The resulting WSDL should look like that shown in Listing 5.3.2.



New WSDL Document

Steps

1. Choose File Type
2. **Name and Location**
3. Abstract Configuration

Name and Location

File Name: PersonAbsSvc

Project: CommonXML

Folder: src

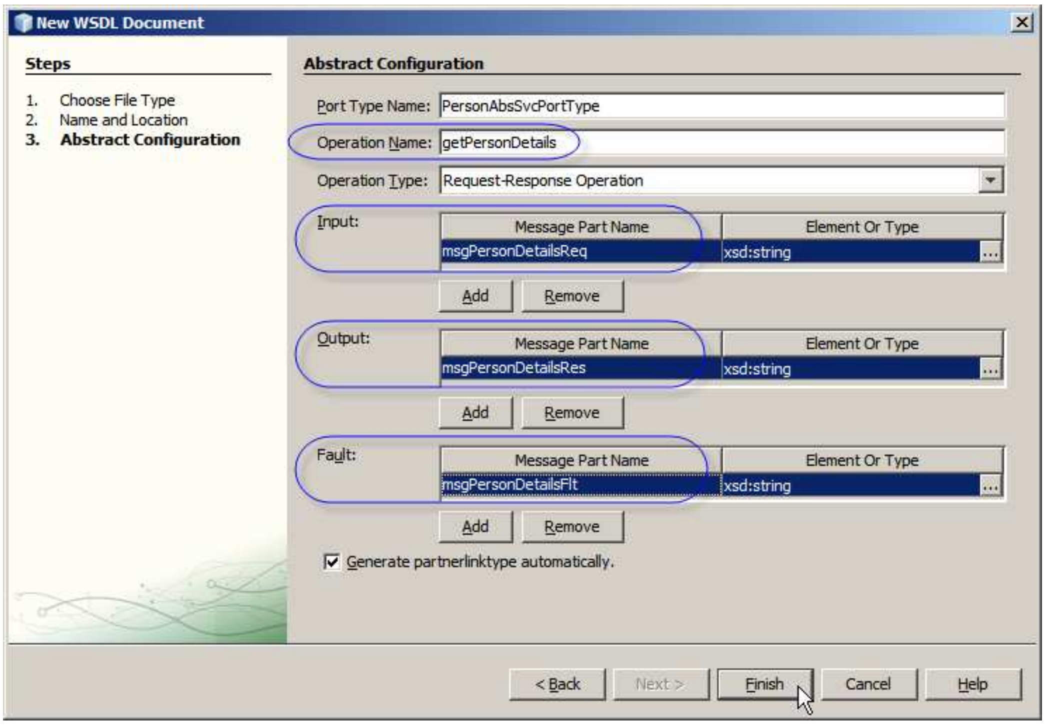
Created File: G:\GlassFishESBv21Projects\WSPolicyExploration\CommonXML\src\PersonAbsSvc.wsdl

Target Namespace: http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc

WSDL Type: ● Abstract WSDL Document

○ Concrete WSDL Document

Figure 5.4.6 *Create a new Abstract WSDL, step 1*



New WSDL Document

Steps

1. Choose File Type
2. Name and Location
3. **Abstract Configuration**

Abstract Configuration

Port Type Name: PersonAbsSvcPortType

Operation Name: getPersonDetails

Operation Type: Request-Response Operation

Input:

Message Part Name	Element Or Type
msgPersonDetailsReq	xsd:string

Output:

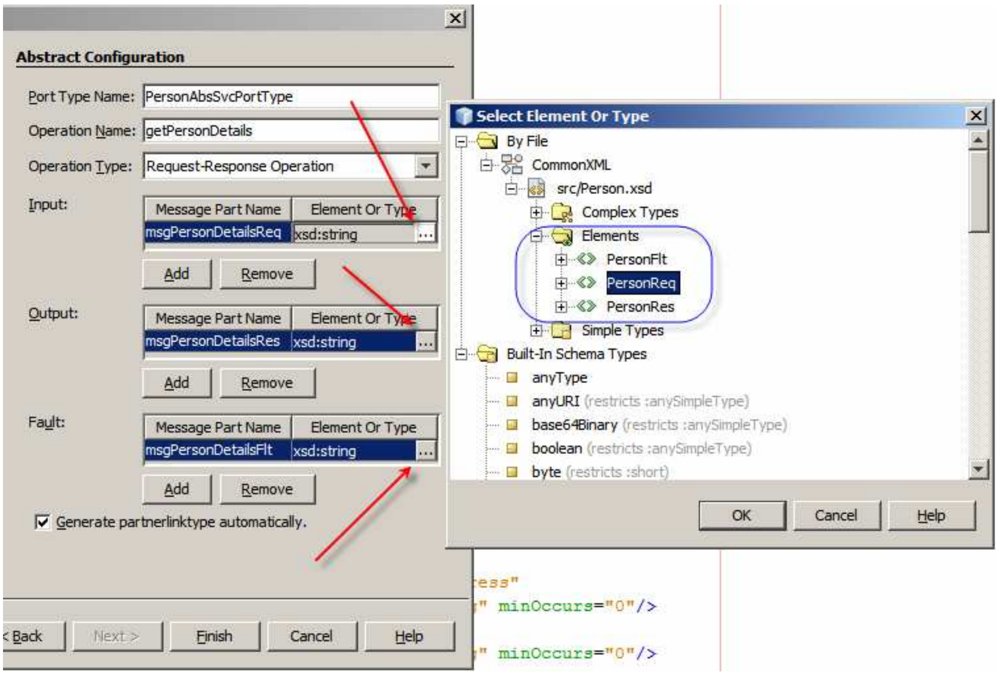
Message Part Name	Element Or Type
msgPersonDetailsRes	xsd:string

Fault:

Message Part Name	Element Or Type
msgPersonDetailsFlt	xsd:string

☑ Generate partnerlinktype automatically.

Figure 5.4.7 *Name operation and message parts*



Abstract Configuration

Port Type Name: PersonAbsSvcPortType

Operation Name: getPersonDetails

Operation Type: Request-Response Operation

Input:

Message Part Name	Element Or Type
msgPersonDetailsReq	xsd:string

Output:

Message Part Name	Element Or Type
msgPersonDetailsRes	xsd:string

Fault:

Message Part Name	Element Or Type
msgPersonDetailsFlt	xsd:string

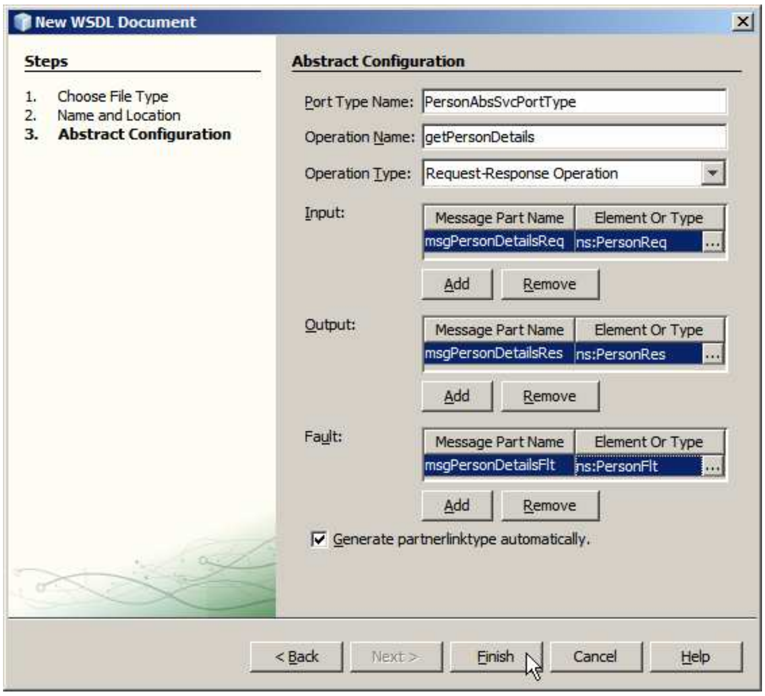
☑ Generate partnerlinktype automatically.

Select Element Or Type

```
By File
  CommonXML
    src/Person.xsd
      Complex Types
      Elements
        PersonFlt
        PersonReq
        PersonRes
      Simple Types
Built-In Schema Types
  anyType
  anyURI (restricts :anySimpleType)
  base64Binary (restricts :anySimpleType)
  boolean (restricts :anySimpleType)
  byte (restricts :short)
```

```
ess"
" minOccurs="0"/>
" minOccurs="0"/>
```

Figure 5.4.8 *Change message part types*


New WSDL Document

Steps

1. Choose File Type
2. Name and Location
3. **Abstract Configuration**

Abstract Configuration

Port Type Name: PersonAbsSvcPortType

Operation Name: getPersonDetails

Operation Type: Request-Response Operation

Input:

Message Part Name	Element Or Type
msgPersonDetailsReq	ns:PersonReq

Output:

Message Part Name	Element Or Type
msgPersonDetailsRes	ns:PersonRes

Fault:

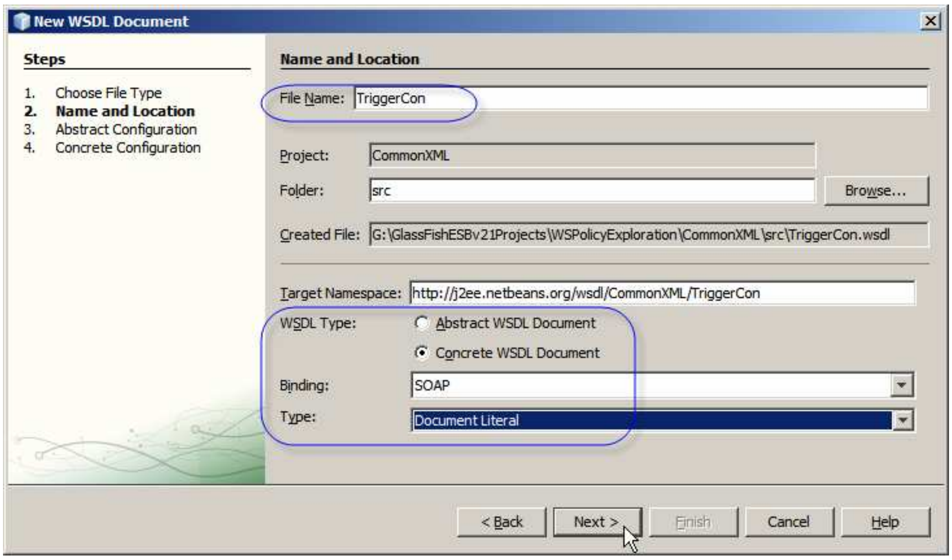
Message Part Name	Element Or Type
msgPersonDetailsFlt	ns:PersonFlt

☑ Generate partnerlinktype automatically.

Figure 5.4.9 *Completed Abstract Configuration*

Finally, let’s create the TriggerCon WSDL, which will be used to expose the client implementation as a web service so it can be triggered by a SopaUI web service testing project.

Create a New -> WSDL Document, names TriggerCon. It will be a Concrete WSDL, SOAP Binding, Document/Literal Type. Figure 5.4.10 shows the dialogue panel at this step in the process.


New WSDL Document

Steps

1. Choose File Type
2. **Name and Location**
3. Abstract Configuration
4. Concrete Configuration

Name and Location

File Name: TriggerCon

Project: CommonXML

Folder: src

Created File: G:\GlassFishESBv21Projects\WSPolicyExploration\CommonXML\src\TriggerCon.wsdl

Target Namespace: http://j2ee.netbeans.org/wsdl/CommonXML/TriggerCon

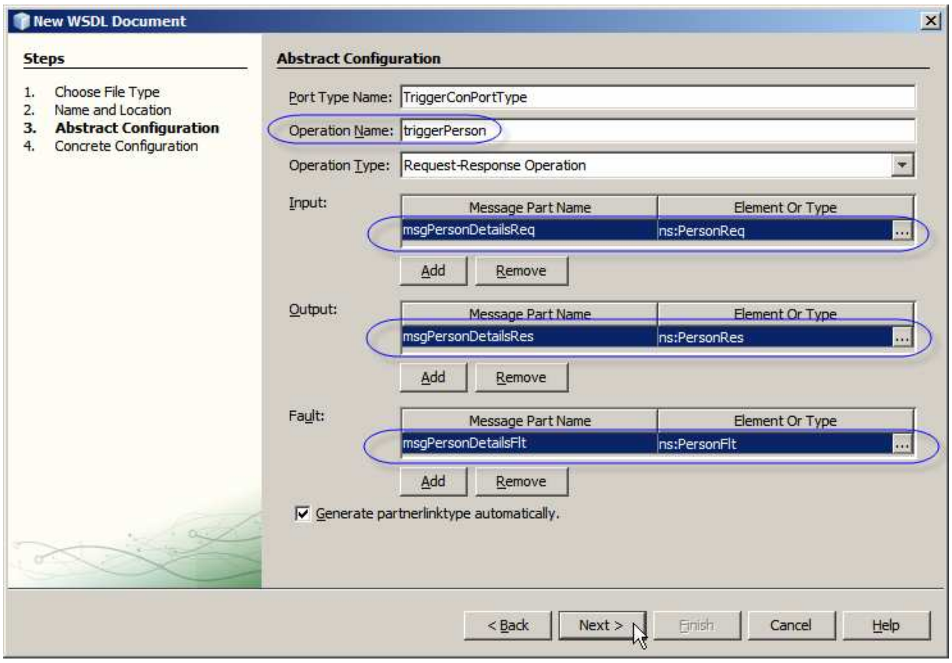
WSDL Type: ○ Abstract WSDL Document ● Concrete WSDL Document

Binding: SOAP

Type: Document Literal

Figure 5.4.10 *Concrete WSDL, SOAP, Document/Literal*

Click Next to advance o the next panel. Change operation name to triggerPerson. Change message part names to msgPersonDetailsReq, msgPersonDetailsRes, add a Fault part and name it msgPersonDetailsFlt. Change “Element or Type” for the message parts to Personreq, PersonRes and PersonFlt, much the same way as was done for the PersonAbsSvc WSDL earlier. Figure 5.4.11 illustrates the final panel.



1. Choose File Type
2. Name and Location
3. **Abstract Configuration**
4. Concrete Configuration

Abstract Configuration

Port Type Name: TriggerConPortType

Operation Name: triggerPerson

Operation Type: Request-Response Operation

Input:

Message Part Name	Element Or Type
msgPersonDetailsReq	ns:PersonReq

Add Remove

Output:

Message Part Name	Element Or Type
msgPersonDetailsRes	ns:PersonRes

Add Remove

Fault:

Message Part Name	Element Or Type
msgPersonDetailsFlt	ns:PersonFlt

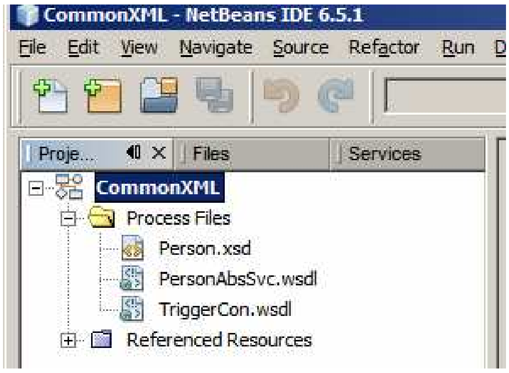
Add Remove

☑ Generate partnerlinktype automatically.

Figure 5.4.11 *TriggerCon WSDL abstract configuration*

Click Next, accept defaults and click Finish.

Our project, CommonXML, should look like that shown in Figure 5.4.12.



- CommonXML
  - Process Files
    - Person.xsd
    - PersonAbsSvc.wsdl
    - TriggerCon.wsdl
  - Referenced Resources

Figure 5.4.12 XML *Schema and WSDL in CommonXML*

5.5 PersonSvc BPEL Module

Let’s create the BPEL Module project, PersonSvc, to implement, in BPEL 2.0, the service whose interface is defined by the PersonSvc WSDL, in CommonXML project.

Right-click in any blank area of the Project Explorer window and choose New Project … -> SOA -> BPEL Module. Name the project PersonSvc.

Expand the node tree to "Referenced Resources". Right-click "Referenced Resources" and choose Add -> Project Resource …, as illustrated in Figure 5.5.1.

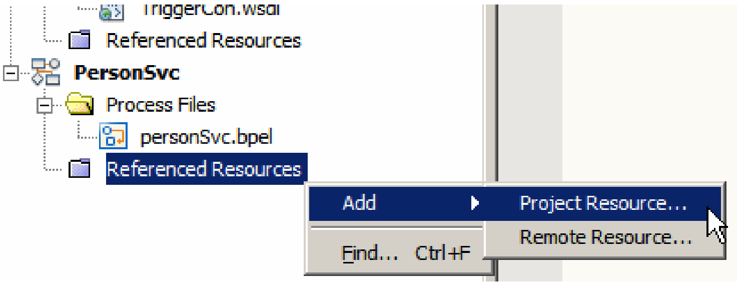
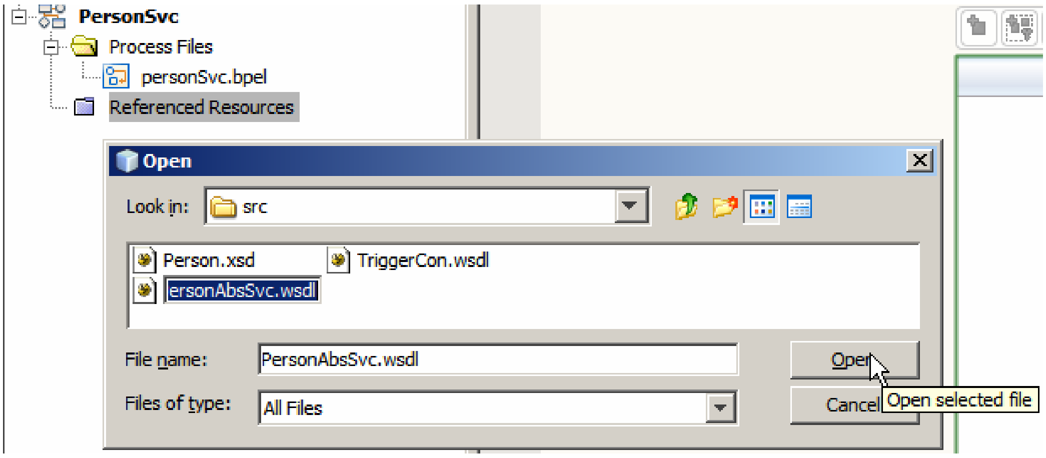


Figure 5.5.1 *Add Project Resource*

Locate the WSDL PersonAbsSvc and click Open, as illustrated in Figure 5.5.2.



File name: PersonAbsSvc.wsdl

Files of type: All Files

Figure 5.5.2 *Locate PersonAbsSvc WSDL*

Open the BPEL process, personSvc.bpel, if it is not already open, and drag the reference CommonXML/PersonAbsSvc onto the target marker in the left-hand swim line, as shown in Figure 5.5.3.

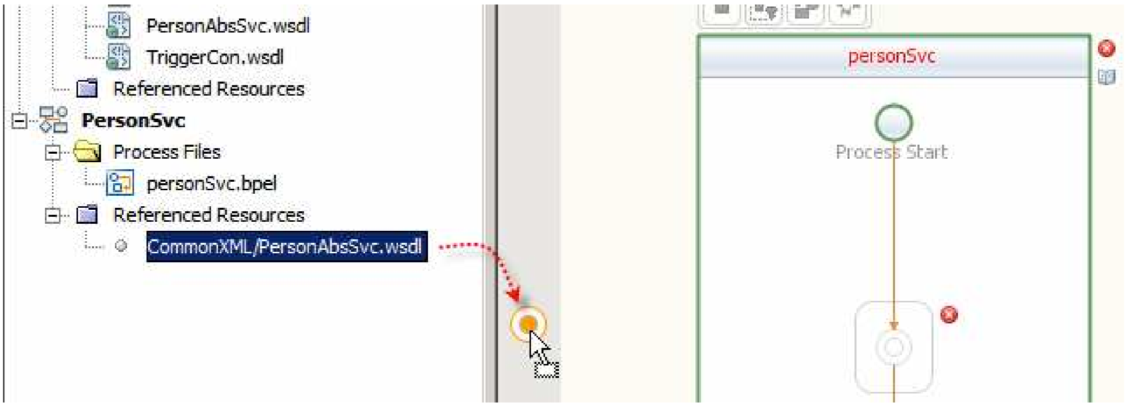


Figure 5.5.3 *Drag PersonAbsSvc WSDL reference onto the process canvas*

Name the partner link PersonRR.

From the Web Service Palette drag Receive, Assign and Reply activities onto the target markers inside the personSvc process scope, as shown in Figure 5.5.4.

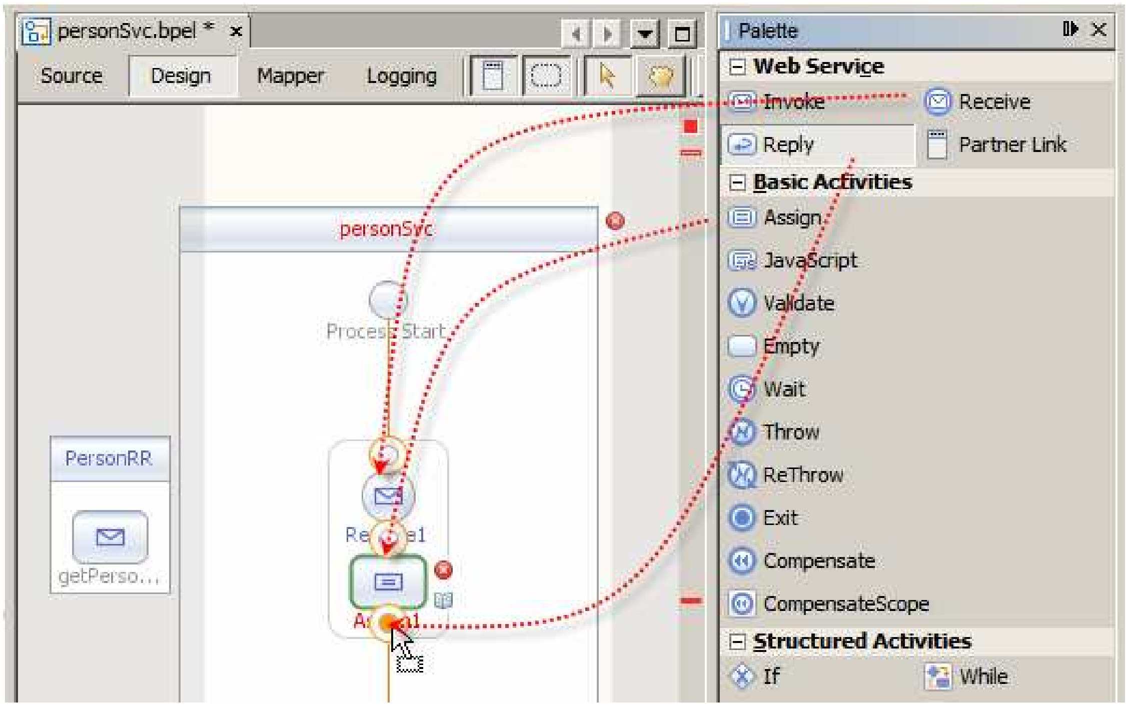


Figure 5.5.4 *Add Receive, Assign and Invoke activities*

Connect Receive and Reply activities to the PersonRR partner Link, as illustrated in Figure 5.5.5.

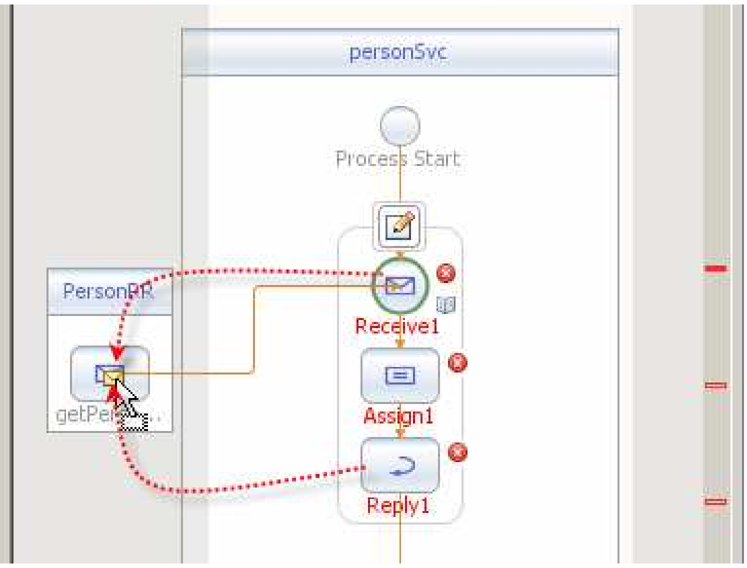
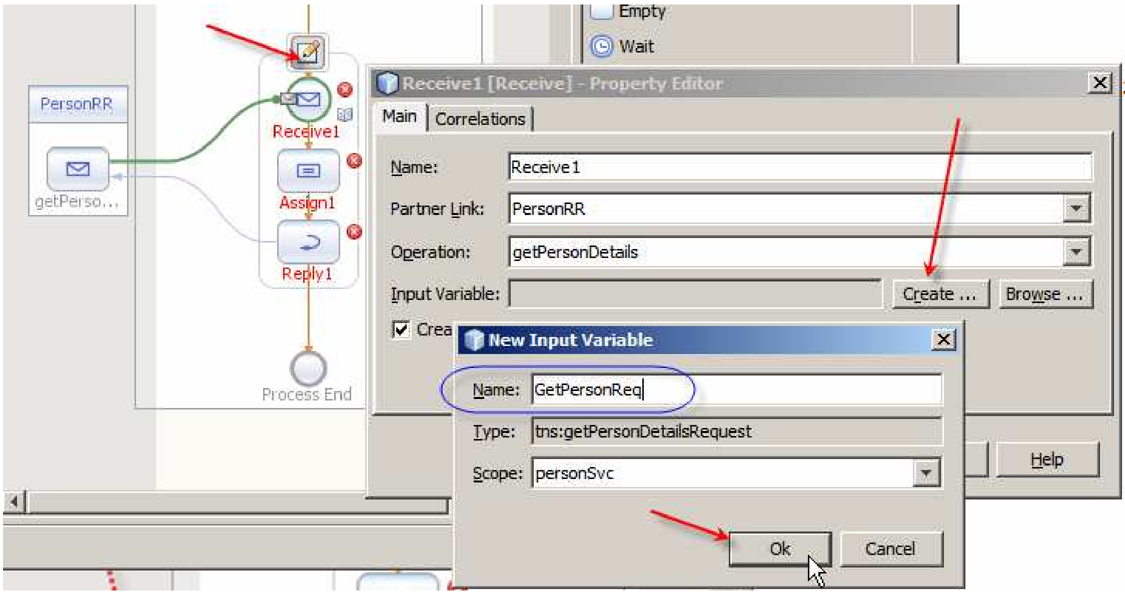


Figure 5.5.5 *Connect Receive and Reply activities to the Partner Link*

Select the Receive activity, click the Edit icon, click the Create button alongside the "Input variable", change the name of the variable to GetPersonReq, click OK. This will add a variable, GetPersonreq, which will contain the request message. Figure 5.5.6 illustrates the interesting points.

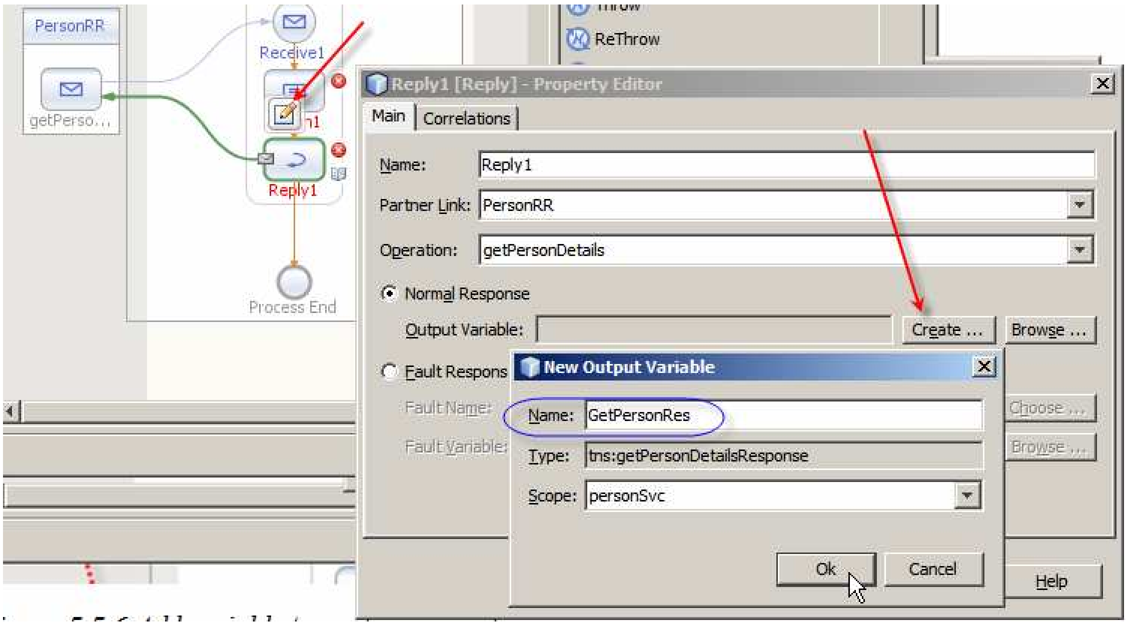


Name: Receive1
Partner Link: PersonRR
Operation: getPersonDetails
Input Variable:

Name: GetPersonReq
Type: tns:getPersonDetailsRequest
Scope: personSvc

Figure 5.5.6 *Add variable to contain request message*

Repeat the process for the Reply activity, naming the variable GetPersonRes, as shown in Figure 5.5.7.



Reply1 [Reply] - Property Editor

Main | Correlations

Name: Reply1

Partner Link: PersonRR

Operation: getPersonDetails

● Normal Response

Output Variable:

○ Fault Respons

Fault Name:

Fault Variable:

New Output Variable

Name: GetPersonRes

Type: tns:getPersonDetailsResponse

Scope: personSvc

Figure 5.5.7 *Add variable GetPersonRes to the reply activity*

Double-click the Assign activity, or select the Assign activity and switch to Mapper mode. When in Mapper, map the request values and literal to the appropriate nodes of the response message. Figure 5.5.8 illustrates the mapping. Feel free to provide your own values for the literals.

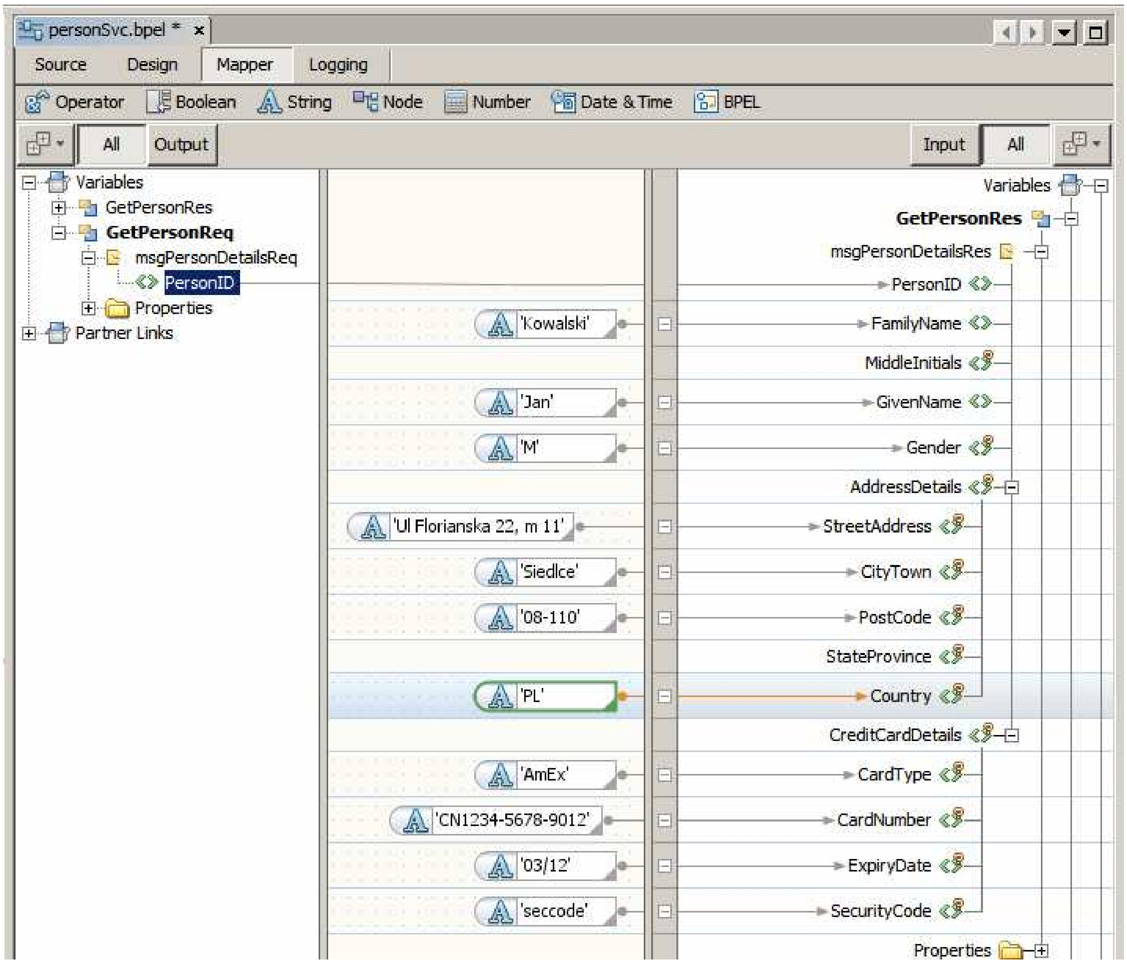


Figure 5.5.8 *Mapping response values*

Right-click the name of the project and choose Build. Figure 5.5.9 illustrates this.

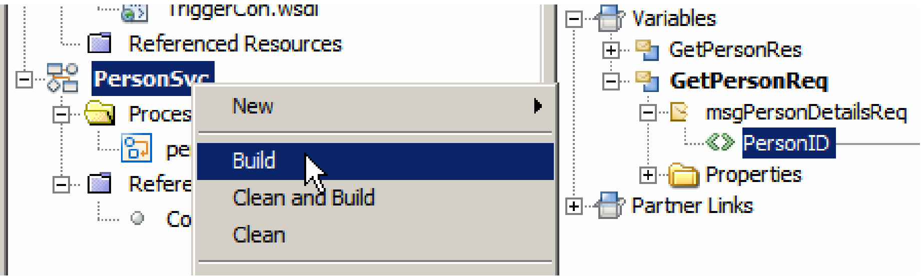


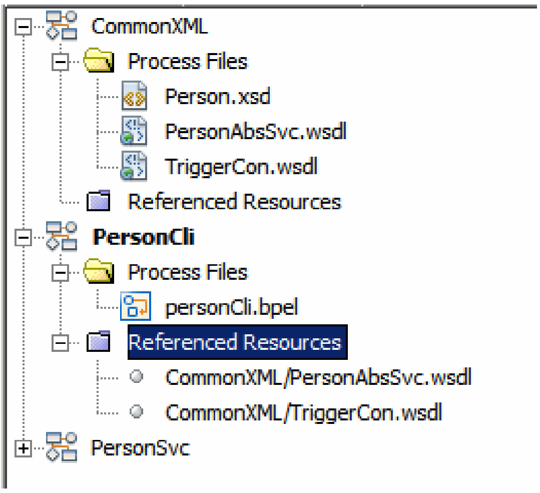
Figure 5.5.9 *Build the project*

The PersonSvc project, which implements service logic, is ready. We will develop the composite application that will encapsulate this logic and deploy it to runtime in subsequent sections.

5.6 *PersonCli BPEL Module*

Let's create the BPEL Module project, PersonCli, to implement, in BPEL 2.0, the invoker of the PersonSvc service. The BPEL process, implemented in this project, will be exposed as a web service using the TriggerCon WSDL, developed earlier. This process will, in turn, invoke the PersonSvc service using the Abstract WSDL interface defined in PersonAbsSvc.

In the new BPEL Module project add two project resource references, the PersdonAbsSvc WSDL and the TriggerCon WSDL. Figure 5.6.1 illustrates project hierarchy after these resources have been added.



```
CommonXML
    Process Files
        Person.xsd
        PersonAbsSvc.wsdl
        TriggerCon.wsdl
    Referenced Resources
PersonCli
    Process Files
        personCli.bpel
    Referenced Resources
        CommonXML/PersonAbsSvc.wsdl
        CommonXML/TriggerCon.wsdl
PersonSvc
```

Figure 5.6.1 *PersonAbsSvc and TriggerCon WSDL References Resources*

Open the personCli business process, if it is not already open. Drag the TriggerCon WSDL reference to the left-hand (provide) swim line and the PersonAbsSvc WSDL reference to the right-hand (invoke) swim line. Name the partner links TriggerRR and PersonWS respectively, as shown in Figure 5.6.2.

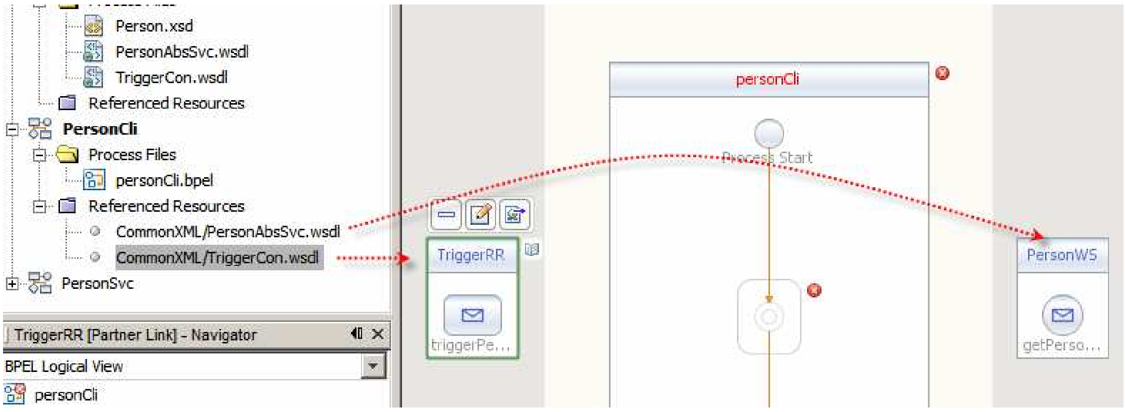


Figure 5.6.2 *Provide and Invoke Partner Links*

Recall that both TriggerCon and PersonAbsSvc use the same request and response structures. BPEL logic we are developing will consist of copying the request message from TriggerCon to PersonAbsSvc and the response message from PersonAbsSvc to TriggerCon. The TriggerCon interface will not be secured in any way so we can conveniently invoke the client service using SoapUI. Security policies, if any, will be applied to the PersonAbsSvc interaction.

Let’s add Receive, Assign, Invoke, Assign and Reply activities to the process canvas, connect Request and Reply to the TriggerRR partner Link and Invoke activity to the PersonWS Partner Link. Figure 5.6.3 illustrates the process at this point in development.

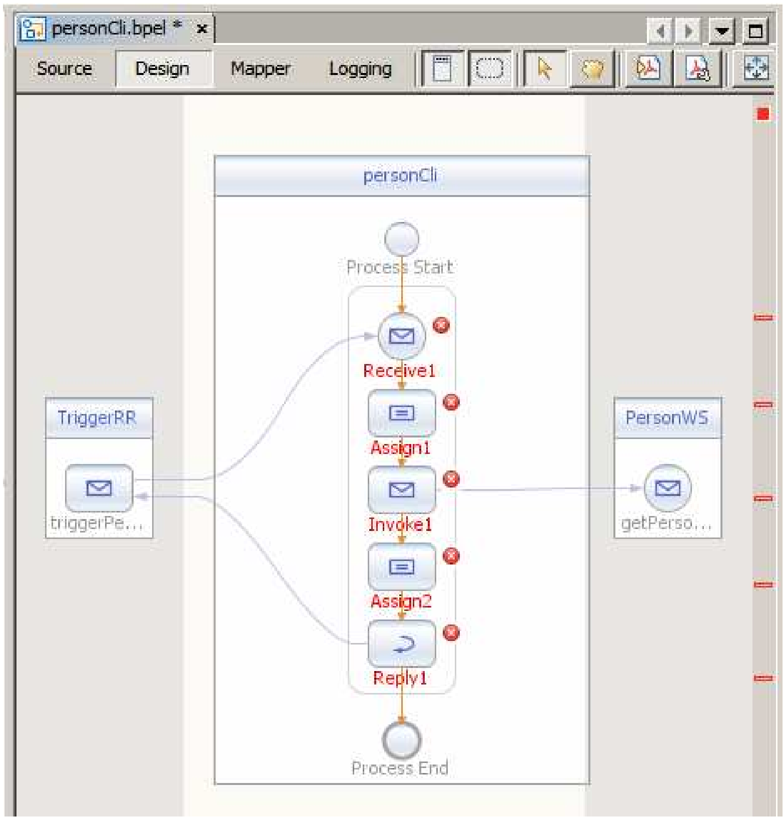
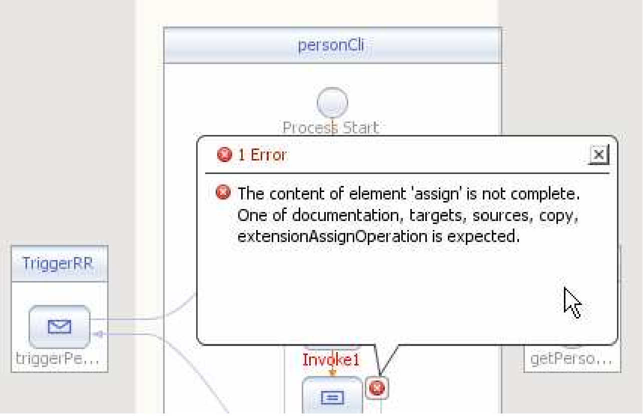


Figure 5.6.3 *Activities added and connected*

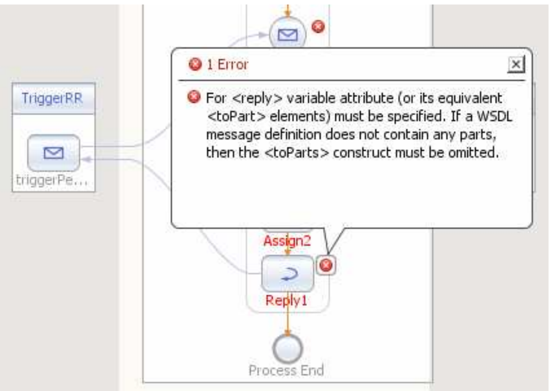
Note the “error indicators”. These tell us that activities are not configured. Figures 5.6.4 and 5.6.5 show error messages for the Assign and the Reply activities.



1 Error

The content of element 'assign' is not complete. One of documentation, targets, sources, copy, extensionAssignOperation is expected.

Figure 5.6.4 *Assign error – no mapping*



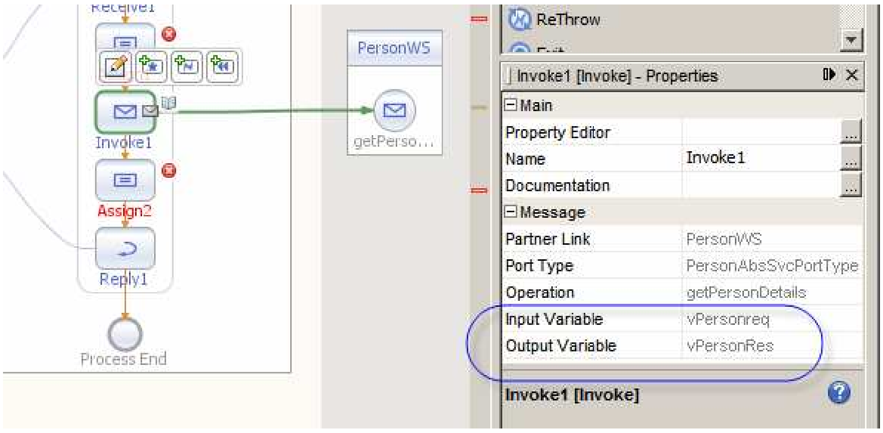
1 Error

For <reply> variable attribute (or its equivalent <toPart> elements) must be specified. If a WSDL message definition does not contain any parts, then the <toParts> construct must be omitted.

Figure 5.6.5 *Error on Reply activity*

BPEL Editor performs continuous background validation so it picks up the fact that we did not finish configuring activities. All these error will be resolved as we continue to work in the process.

Edit Receive, Reply and Invoke activities and add variables that will contain messages – vTriggerReq for Receive, vTriggerRes for the Reply and vPersonReq and vPersonRes for the Invoke. This is done the same way as has been done in the ProcessSvc so no pictures should be necessary. Figure 5.6.6 calls out variable names configured for the Invoke activity.



Invoke1 [Invoke] - Properties

Main	
Property Editor	
Name	Invoke1
Documentation	
Message	
Partner Link	PersonWS
Port Type	PersonAbsSvcPortType
Operation	getPersonDetails
Input Variable	vPersonreq
Output Variable	vPersonRes

Invoke1 [Invoke]

Figure 5.6.6 *Variable names for the Invoke activity*

We can now complete the Assign activities. Mapping in Assign1 are shown in Figure 5.6.7.

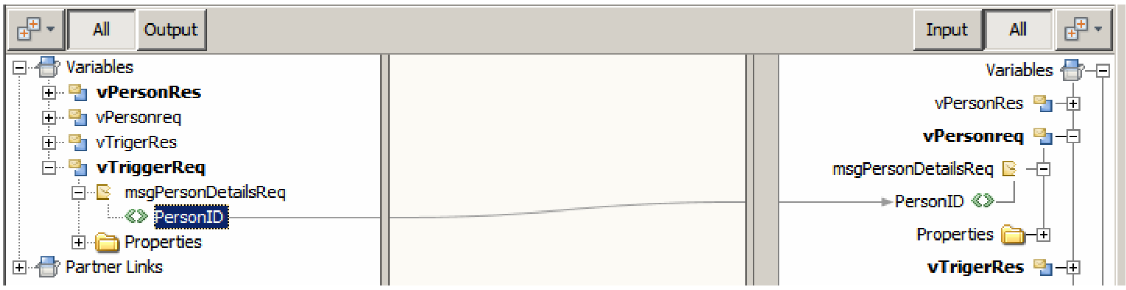


Figure 5.6.7 *Mapping in Assign1*

Mapping in Assign2 are shown in Figure 5.6.8.

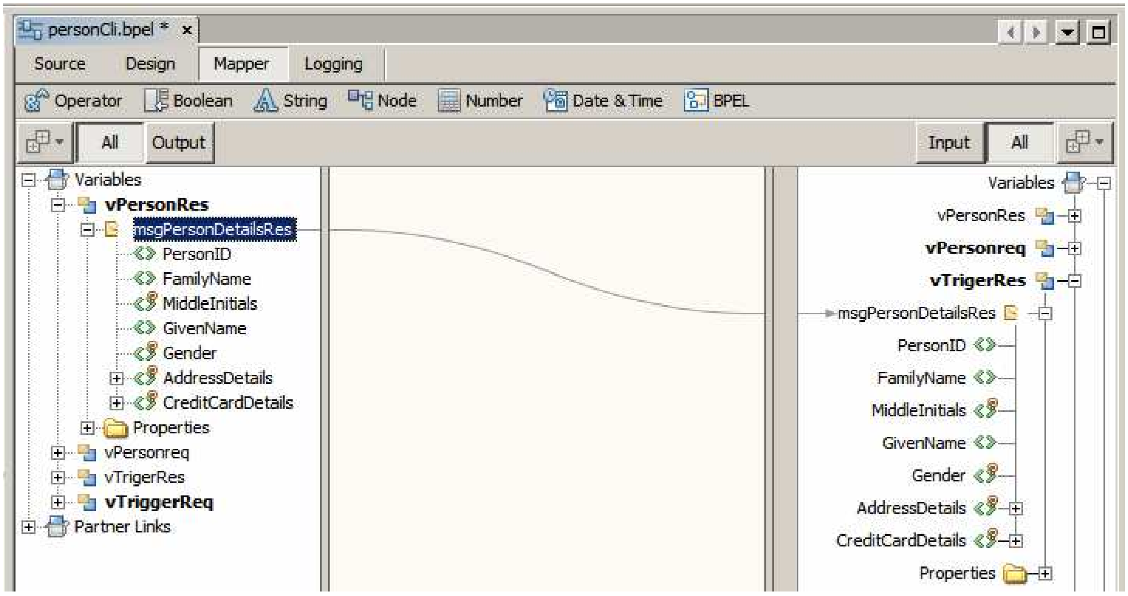
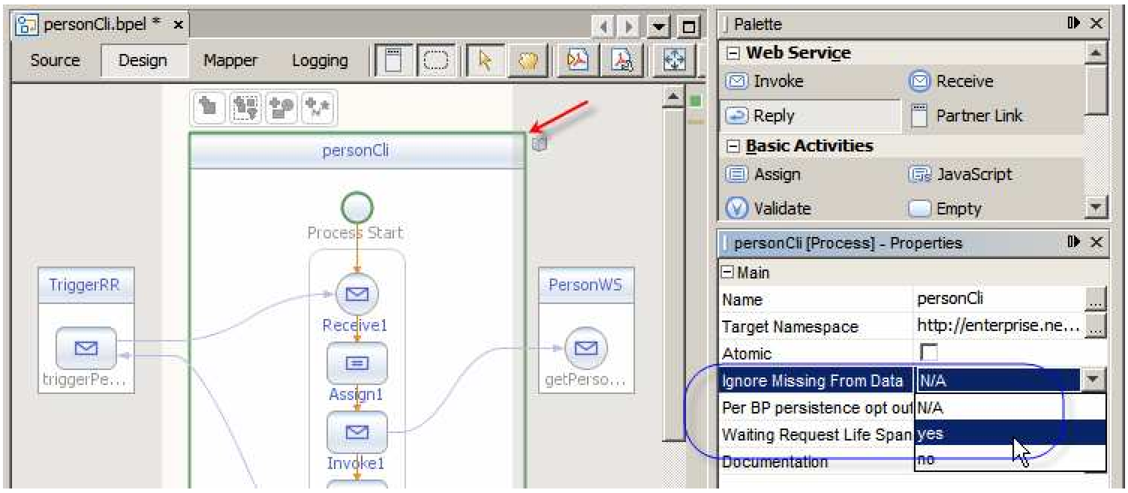


Figure 5.6.8 *Mapping in Assign2*

In Assign2 we map the root nodes, instead of mapping each individual field. We can do this because both the source and the destination messages are of the same structure.

Finally, let's configure the process so that it is lenient with respect to missing data. Switch to Design view, click the personCli process scope and choose "Yes" for the value of process property "Ignore Missing From Data". Figure 5.6.9 illustrates this.



Name personCli
Target Namespace http://enterprise.ne...
Atomic □
Ignore Missing From Data N/A
Per BP persistence opt out
Waiting Request Life Span
Documentation

Figure 5.6.9 *Set "Ignore Missing From Data" to "Yes"*

The PersonCli project, which implements client-side logic, is ready. Build the project.

We will develop the composite application that will encapsulate this logic module and deploy it to runtime in the subsequent sections.

5.7 Person Service – Plain End-to-End

The service provider and service invoker BPEL Module are ready. We are now in a position to create Composite Applications for each and to exercise the solution end-to-end.

Let's start by creating the composite application, PersonSvc_CA_Plain, for the PersonSvc BPEL module, a web service testing project, PersonSvc_WSTP, to exercise this application, then perform the service implementation test.

Create a New Project -> SOA -> Composite Application, named PersonSvc_CA_Plain. Once created, drag the BPEL Module PersonSvc onto the Composite Application Service Assembly canvas and click Build. Figure 5.7.1 illustrates this.

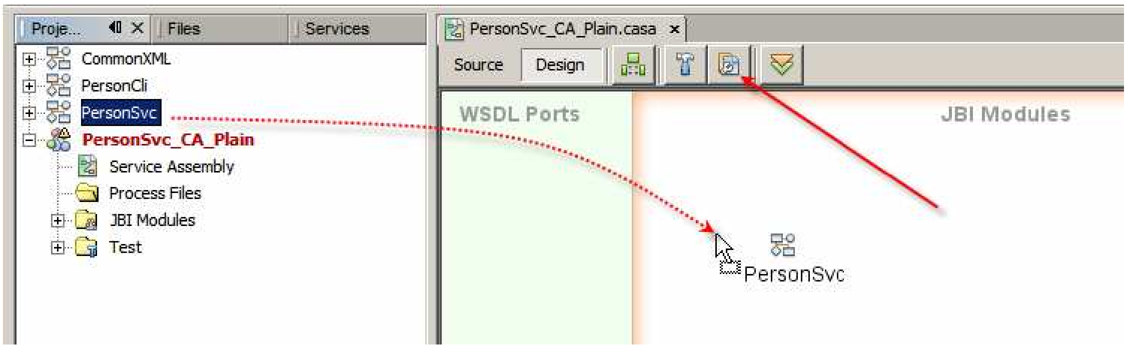


Figure 5.7.1 *Add BPLE Module to the CASA canvas and Build*

Because the service interface WSDL is an Abstract WSDL we don't see a Binding Component on the CASA canvas once the build is finished. We need to provide a concrete binding. AT this point we could use any available binding. Since we are building a web service implementation we will drag the soap binding to the canvas. Figure 5.7.2 illustrates this.

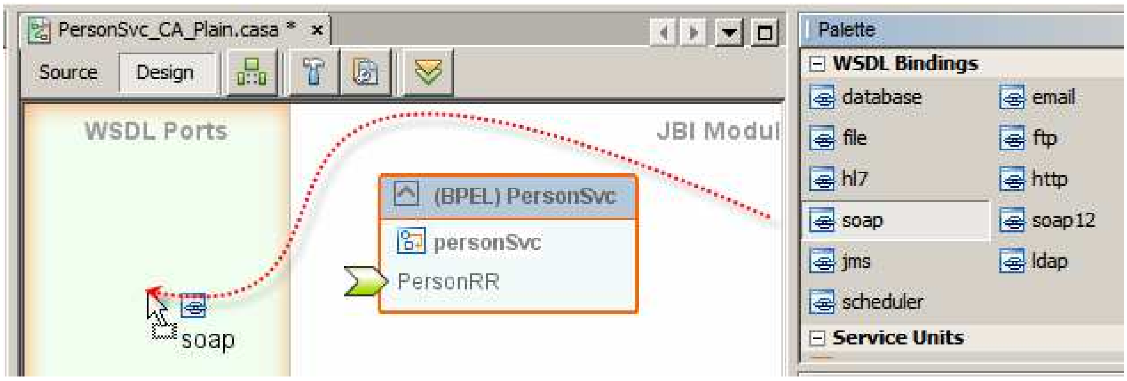


Figure 5.7.2 *Add soap binding to the CASA canvas*

Connect Consume connector of the SOAP BC to the Provide connector of the BPEL Module then build the process again. Figure 5.7.3 illustrates the final CASA map.

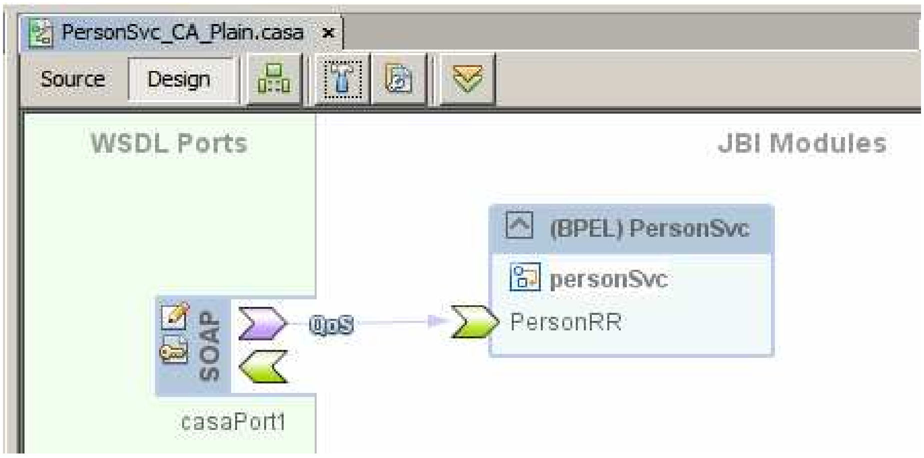
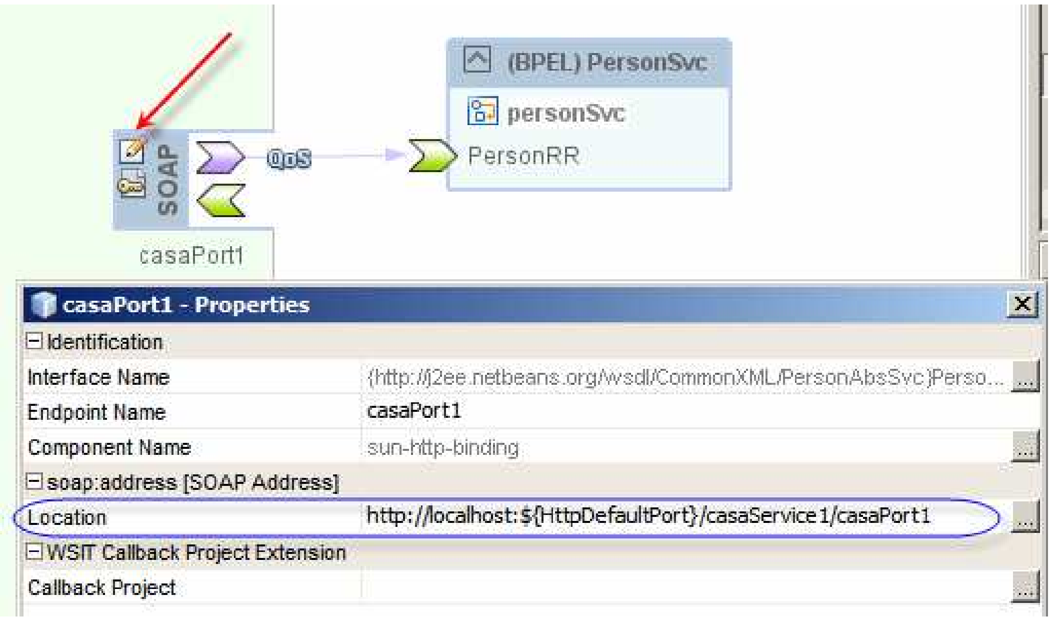


Figure 5.7.3 *Completed CASA map*

Click the “pencil and paper” icon to open SOAP BC properties and note the endpoint address in the Location property, shown in Figure 5.7.4.



casaPort1 - Properties

⊟ Identification	
Interface Name	{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}Perso...
Endpoint Name	casaPort1
Component Name	sun-http-binding
⊟ soap:address [SOAP Address]	
Location	http://localhost:${HttpDefaultPort}/casaService1/casaPort1
⊟ WSIT Callback Project Extension	
Callback Project	

Figure 5.7.4 *Location property*

Note the construction “${HttpDefaultPort}”. The HttpDefaultPort is the name of the environment variable that gets replaced, at build time, with the value configured for the default HTTP port used by the JBI container. By default this will be 9080. For me this will be 29080. You can find out what it is by looking at properties of Services -> Servers -> GlassFish v2 -> JBI -> Binding Components -> sun-http-binding, specifically property named “Default HTTP Port Number”. While at it, also note the value of the “Default HTTPS Port Number”. This is the port for the SSL/TLS protocol. More on that later.

Deploy the project to the local GlassFish instance. For me this will be the “GlassFish v2” running on host mcz02.aus.sun.com.

Let’s now create a New Project … -> Java EE -> Web Service Testing Project and name it PersonSvc_WSTP. We will use this project to submit a SOAP request to the PersonSvc_CA_Plain service, which we just built and deployed, to verify that it works. Enter, or paste, the endpoint URL from the Location property, discussed above, with host and port configured as required, into the property “Initial WSDL (URL)”. For me this will be:

```
http://mcz02.aus.sun.com:29080/casaService1/casaPort1?WSDL
```

For you the FQDN of the host will be different and the port number will be 9080 if you have a default GlassFish installation.

Once the project is created, expand the nodes all the way to getPersonDetails, right-click and choose New Request. Figure 5.7.5 illustrates this.

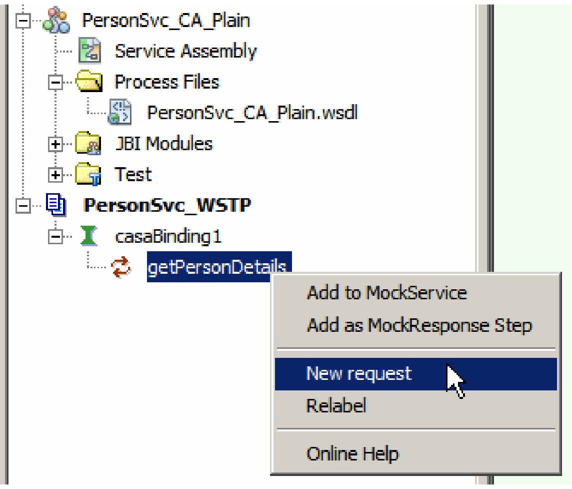
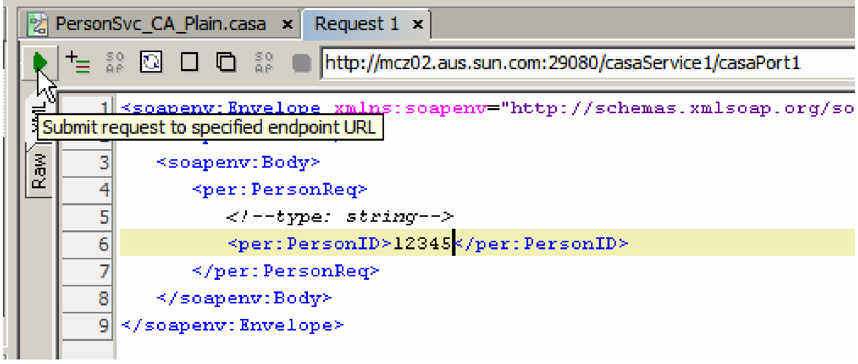


Figure 5.7.5 *Create new Soap Request*

Modify the request by replacing “gero et” with “q2345”, or whatever value you find attractive, and submit the request as shown in Figure 5.7.6.

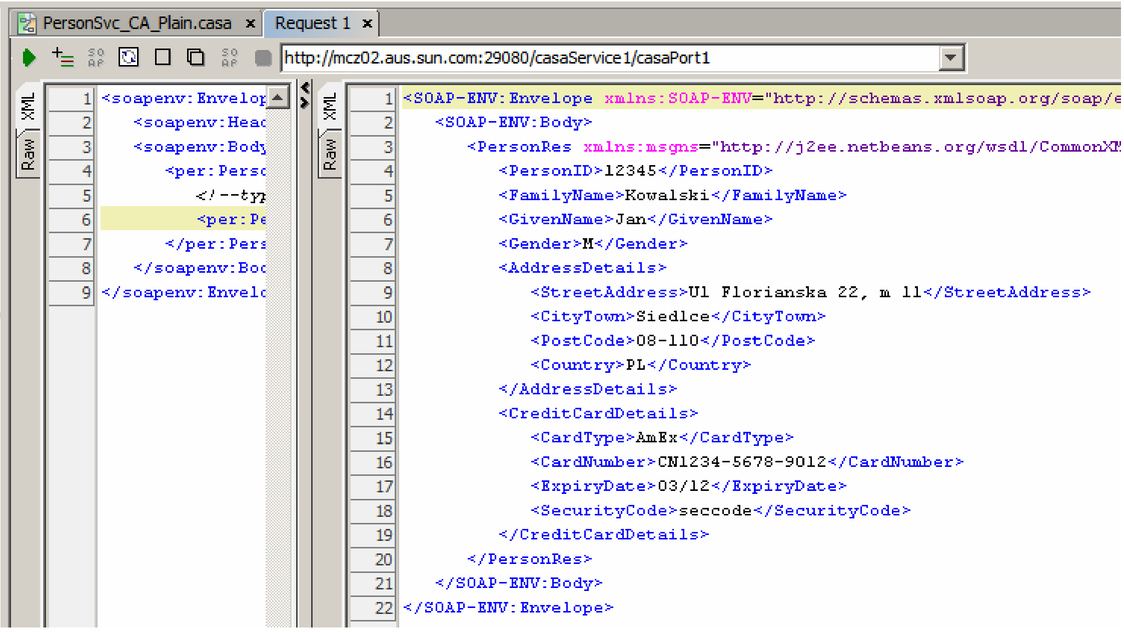


```
http://mcz02.aus.sun.com:29080/casaService1/casaPort1
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/so
   <soapenv:Body>
      <per:PersonReq>
         <!--type: string-->
         <per:PersonID>12345</per:PersonID>
      </per:PersonReq>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 5.7.6 *Submit SOAP Request with the PersonID of 12345*

With mapping in the PersonSvc BPEL Module as shown in Figure 5.5.8 the SOAP Response will look like that in Figure 5.7.7.



http://mcz02.aus.sun.com:29080/casaService1/casaPort1

```
<soapenv:Envelop
    <soapenv:Heac
    <soapenv:Body
        <per:Persc
            <!--typ
            <per:Pe
        </per:Pers
    </soapenv:Boc
</soapenv:Envelc
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/e
    <SOAP-ENV:Body>
        <PersonRes xmlns:msgns="http://j2ee.netbeans.org/wsdl/CommonXM
            <PersonID>12345</PersonID>
            <FamilyName>Kowalski</FamilyName>
            <GivenName>Jan</GivenName>
            <Gender>M</Gender>
            <AddressDetails>
                <StreetAddress>Ul Florianska 22, m 11</StreetAddress>
                <CityTown>Siedlce</CityTown>
                <PostCode>08-110</PostCode>
                <Country>PL</Country>
            </AddressDetails>
            <CreditCardDetails>
                <CardType>AmEx</CardType>
                <CardNumber>CN1234-5678-9012</CardNumber>
                <ExpiryDate>03/12</ExpiryDate>
                <SecurityCode>seccode</SecurityCode>
            </CreditCardDetails>
        </PersonRes>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 5.7.7 *SOAP Response*

The service PersonSvc works as expected. We will not use the testing project PersonSvc_WSTP again. We will create t6he composite application for the PersonCli BPEL Module and will use it to exercise the end-to-end solution.

Create a New Project -> SOA -> Composite Application, named PersonCli_CA_Plain. Drag the PersonCli BPEL Module onto the CASA canvas and Build. Figure 5.7.8 illustrates the key points.

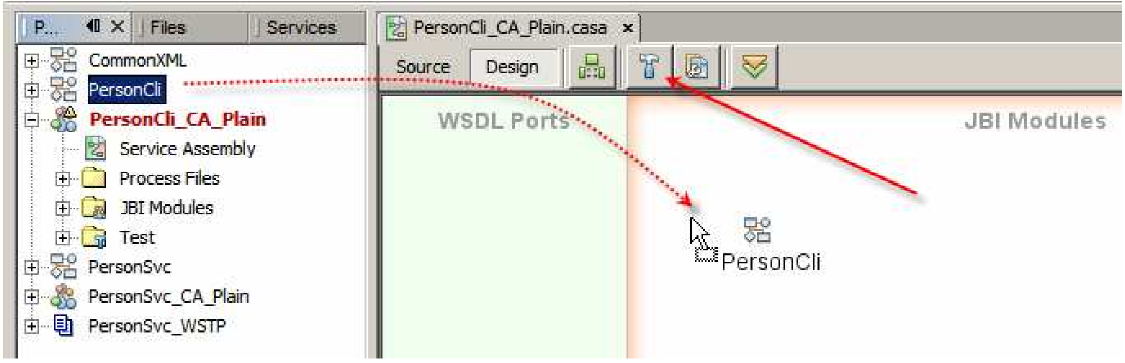


Figure 5.7.8 *Create PersonCli_CA_Plain Composite Application*

Note that the Consume connector of the PersonCli BPEL Module is not connected to a binding component. This is because the PersonAbsSvc WSDL, which we used in the BPEL process, is an Abstract WSDL. Figure 5.7.9 illustrates the CASA canvas at this point.

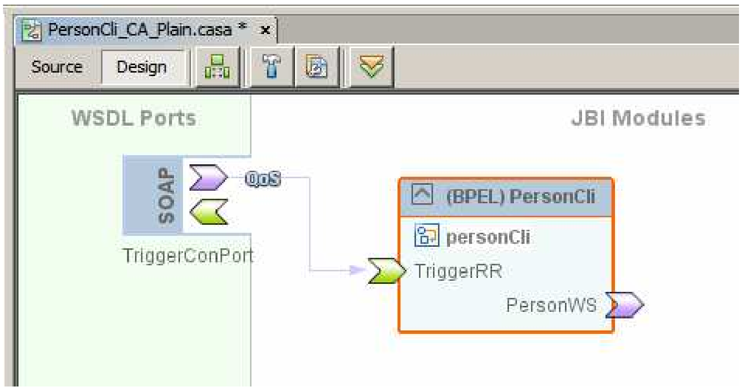


Figure 5.7.9 *CASA canvas with unconnected PersonWS Partner Link*

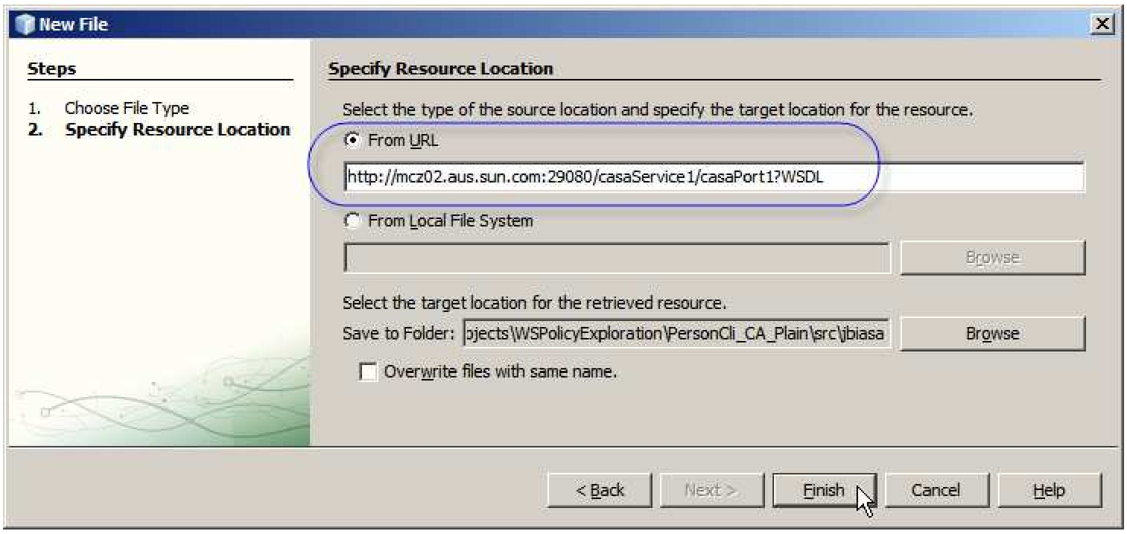
We will add a concrete WSDL to this composite application project to provide concrete binding for the PersonWS partner link.

Copy the WSDL URL of the PersonSvc_CA_Plain service to the clipboard. For me this will be:

```
http://mcz02.aus.sun.com:29080/casaService1/casaPort1?WSDL
```

For you the FQDN of the host will be different and the port number will be 9080 if you have a default GlassFish installation.

Right-click on the name of the project, PersonCli_CA_Pain, choose New -> Other -> XML -> “External WSDL Document(s)” and paste the WSDL URL into the “From URL” text box. Figure 5.7.10 illustrates the dialogue box.



New File

Steps

1. Choose File Type
2. **Specify Resource Location**

Specify Resource Location

Select the type of the source location and specify the target location for the resource.

● From URL

http://mcz02.aus.sun.com:29080/casaService1/casaPort1?WSDL

○ From Local File System

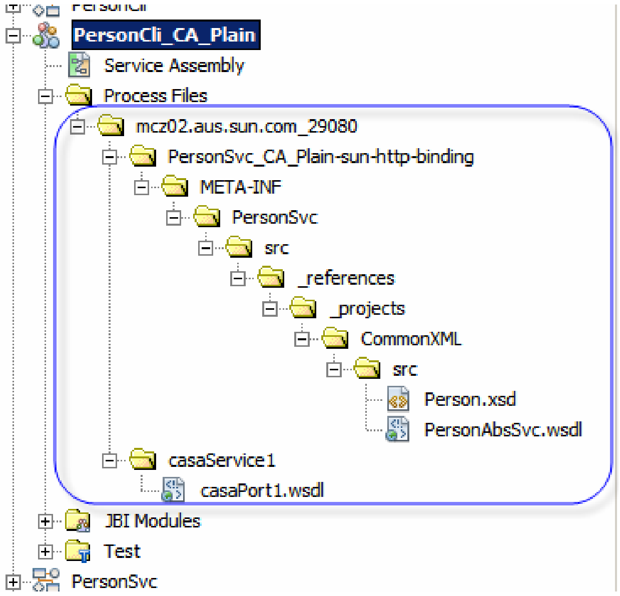
Select the target location for the retrieved resource.

Save to Folder: ojects\WSPolicyExploration\PersonCli_CA_Plain\src\jbiasa

□ Overwrite files with same name.

Figure 5.7.10 *New External WSDL document being created*

New WSDL and XSD objects will be added under the Process File node in the project hierarch. Figure 5.7.11 illustrates this.



```
PersonCli
PersonCli_CA_Plain
    Service Assembly
    Process Files
        mcz02.aus.sun.com_29080
            PersonSvc_CA_Plain-sun-http-binding
                META-INF
                    PersonSvc
                        src
                            _references
                                _projects
                                    CommonXML
                                        src
                                            Person.xsd
                                            PersonAbsSvc.wsdl
            casaService1
                casaPort1.wsdl
    JBI Modules
    Test
PersonSvc
```

Figure 5.7.11 *WSDLs and XSD added to the Composite Application project*

Right-click on the CASA canvas inside the “WSDL Ports” swim line and choose “Load WSDL Port…”, as shown in Figure 5.7.12, to add the SOAP BC, configured to communicate with the PersonSVc service, to the CASA canvas.

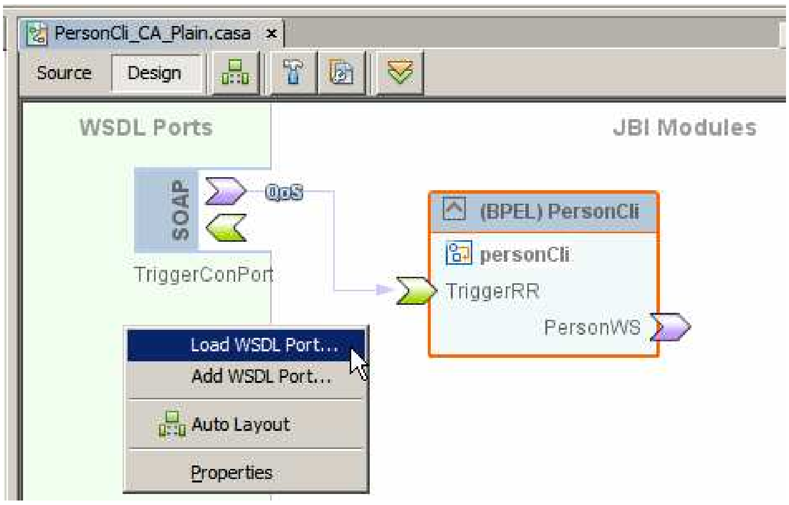
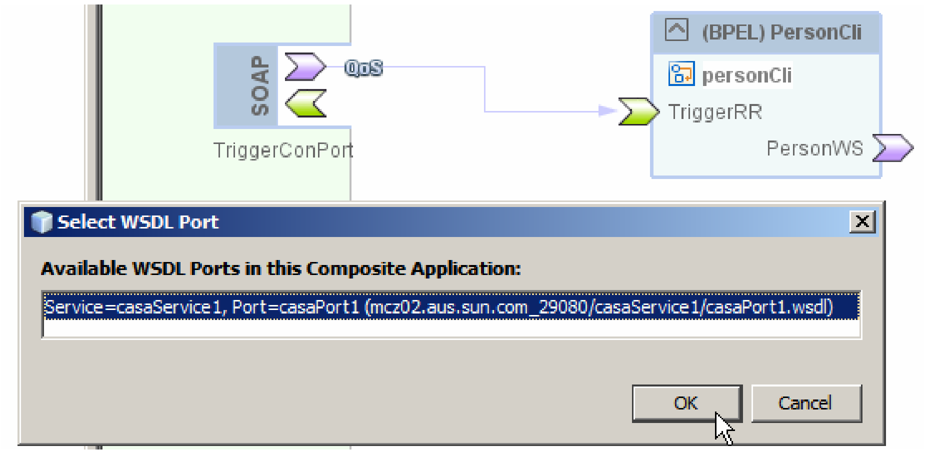


Figure 5.7.12 *Load WSDL Port, part 1*

In the dialogue box that appears select the one and only WSDL Port, as shown in Figure 5.7.13, and click OK. Build the CA project.



Available WSDL Ports in this Composite Application:

Service=casaService1, Port=casaPort1 (mcz02.aus.sun.com_29080/casaService1/casaPort1.wsdl)

Figure 5.7.13 *Select WSDL Port to add*

The CASA canvas should now look like that shown in Figure 5.7.14.

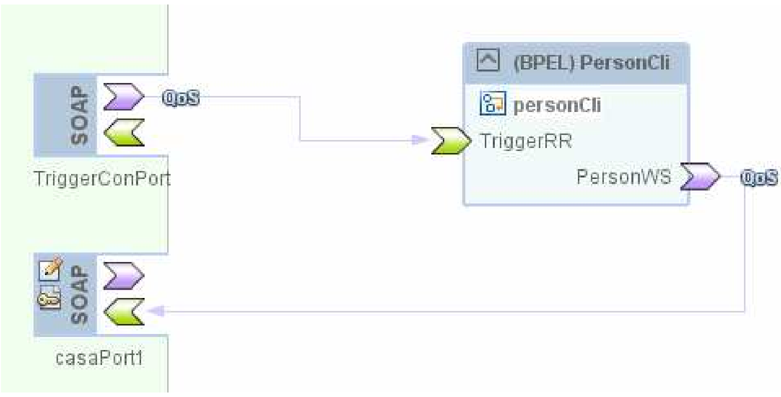


Figure 5.7.14 *CASA canvas with both binding components added and connected*

Deploy the project to the local instance of the GlassFish Applciatin Server. For me this will be “GlassFish v2” running on mcz02.aus.sun.com.

Locate and copy to the clipboard the endpoint URL for the TriggerCon connector. The WSDL associated with that endpoint, TriggerCon.wsdl in project CommonXML, will have that value. For me this is:

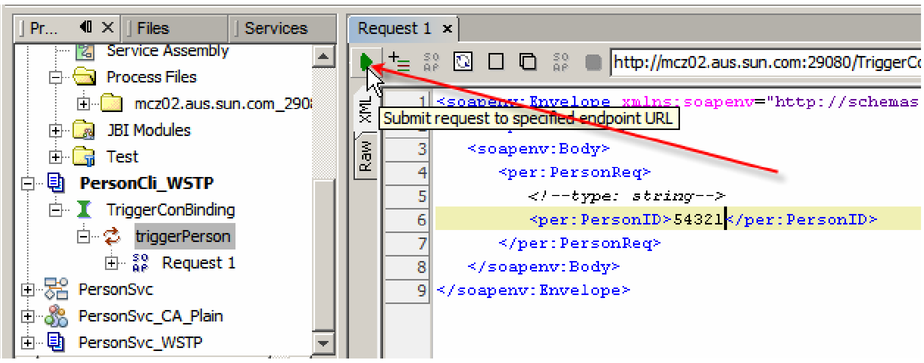
```
http://localhost:${HttpDefaultPort}/TriggerConService/TriggerConPort
```

Replace ${HttpDefaultPort} with the correct value. For me this will be 29080, and append “?WSDL? to the end of the URL. For me the final value will be:

```
http://localhost:29080/TriggerConService/TriggerConPort?WSDL
```

This is the URL to which the web service testing project, created next, will submit SOAP requests. Create a New -> Java EE -> Web Service testing Project, names PersonCli_WSTP. Use the URL shown above as the “Initial WSDL (URL/file)”.

Expand the nodes, right-click on triggerPerson binding, choose “New Request” and create a request. Modify PersonID value to 54321, or whatever value you find attractive, ans submit the request. Figure 5.7.15 illustrates the request.



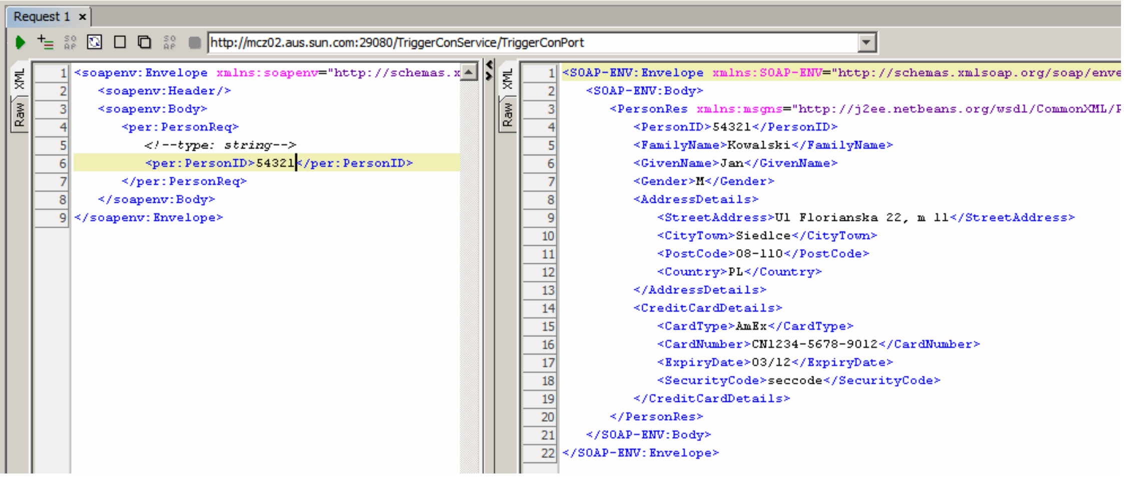
http://mcz02.aus.sun.com:29080/TriggerC

Submit request to specified endpoint URL

```
<soapenv:Envelope xmlns:soapenv="http://schemas
   <soapenv:Body>
      <per:PersonReq>
         <!--type: string-->
         <per:PersonID>54321</per:PersonID>
      </per:PersonReq>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 5.7.15 *SOAP Request to be submitted to the PersonCli service*

Observe the response – it should be like that shown in Figure 5.7.16.



http://mcz02.aus.sun.com:29080/TriggerConService/TriggerConPort

```
<soapenv:Envelope xmlns:soapenv="http://schemas.x
   <soapenv:Header/>
   <soapenv:Body>
      <per:PersonReq>
         <!--type: string-->
         <per:PersonID>54321</per:PersonID>
      </per:PersonReq>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/enve
   <SOAP-ENV:Body>
      <PersonRes xmlns:msgns="http://j2ee.netbeans.org/wsdl/CommonXML/P
         <PersonID>54321</PersonID>
         <FamilyName>Kowalski</FamilyName>
         <GivenName>Jan</GivenName>
         <Gender>M</Gender>
         <AddressDetails>
            <StreetAddress>Ul Florianska 22, m 11</StreetAddress>
            <CityTown>Siedlce</CityTown>
            <PostCode>08-110</PostCode>
            <Country>PL</Country>
         </AddressDetails>
         <CreditCardDetails>
            <CardType>AmEx</CardType>
            <CardNumber>CN1234-5678-9012</CardNumber>
            <ExpiryDate>03/12</ExpiryDate>
            <SecurityCode>seccode</SecurityCode>
         </CreditCardDetails>
      </PersonRes>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 5.7.16 *SOAP Response*

We have the client / invoker (PersonCli) invoking the service / provider, PersonSvc. The end-to-end project works, as does the BPEL logic in the client and the service implementations.

Note that we did not do anything about security policies, nor did we even mention them until now. SOAP requests and SOAP responses are exchanged “in the clear”, in plain text. Anybody eavesdropping on the wire can see the content of the messages.

Before proceeding to the next section undeploy both the invoker and the provider projects.

5.8 Person Service - SSL Server-side Authentication

One way to prevent eavesdropping on messages being exchange between the invoker and the provider is to encrypt the channel between the two. Since web services use the HTTP protocol one can use the Secure Sockets Layer (SSL) / Transport Layer Security (TLS) to encrypt the channel. This is a common mechanism used for securing message exchange with electronic commerce sites in order to prevent intercept of credit card details and other sensitive commercial information submitted by purchasers.

There is a great deal to SSL /TLS. More then I am prepared to discuss in this section. I assume that the reader is either sufficiently familiar with the protocol's operation to not require elaboration, or that the reader does not care for the theory and will be satisfied with the practice as discussed here. All others are referred to the excellent book by Eric Rescorla, "SSL and TLS: Designing and Building Secure Systems", ISBN-10: 0201615983, for elaboration.

SSL with Server-side Authentication adds security to the message exchange in two ways.

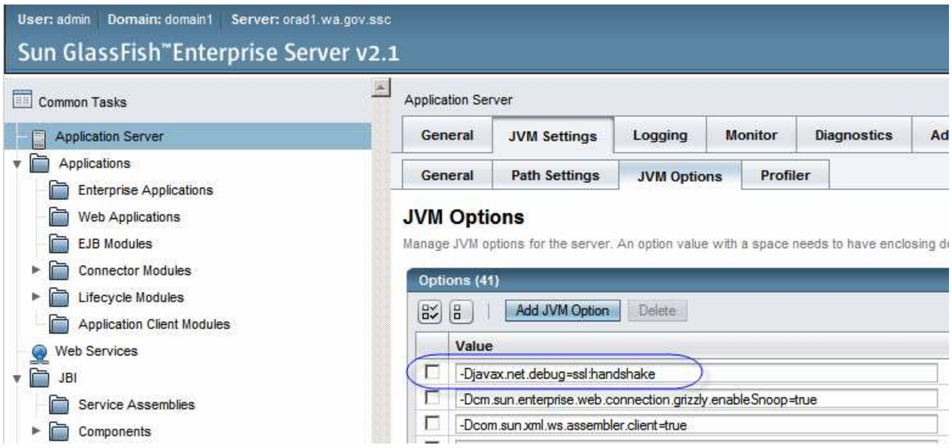
By requiring the server to provide the X.509 Certificate, expected to be issued by a trusted Certification Authority (CA) for a specific Host, the client is able to assure itself that the FQDN of the server is the same as the FQDN of the host in the certificate, therefore no substitution of hosts took place. The client is also able to validate the server certificate by verifying the digital signature of the CA, if the certificate was issued by a well know CA. Signature verification ensures that the certificate was not tampered with and the FQDN of the host was not altered. If the certificate signature is not valid, the certificate FQDN host name is not the same as the server host name, the certificate is a self-signed certificate or the CA is not a well know CA, then client would typically reject the certificate and abort the SSL Handshake. It is possible that the FQDN of the server will not be the same as the FQDN in the certificate. This may be legitimate inside an enterprise. To prevent rejection of the certificate a custom Hostname Verifier class can be provided that resolves this discrepancy. This is beyond the scope of this text. To prevent rejection of a certificate issued by a non-well know CA one can add the CA's certificate to the Client's truststore and mark it as trusted. This will make all certificates issued by the CA trusted by extension. This can also be legitimate inside an enterprise or between enterprises that explicitly trust each other. A self-signed certificate can be made trusted, therefore acceptable, the same way as an unknown CA can be made trusted – by being added to the Client's truststore and marked as trusted.

By encrypting the channel over which messages travel both the Client and the Server ensure that message exchange can not be profitably eavesdropped upon.

These two ways are typically used together. It is possible, though not common, to use channel encryption without certificate exchange. An internal enterprise application concerned with channel security, but not with endpoint authentication, might do that. Endpoint authentication in such an application might be provided in some other way, for example by embedding credentials in messages themselves. Channel encryption will protect these credentials.

Before proceeding with development let's do some groundwork.

SSL Handshake can be logged to the server.log by adding "-Djavax.net.debug=ssl:handshake" to the GlassFish Application Server's JVM Options. Figure 5.8.1 illustrates this in the GlassFish Application Server Admin Console.



JVM Options

Manage JVM options for the server. An option value with a space needs to have enclosing d

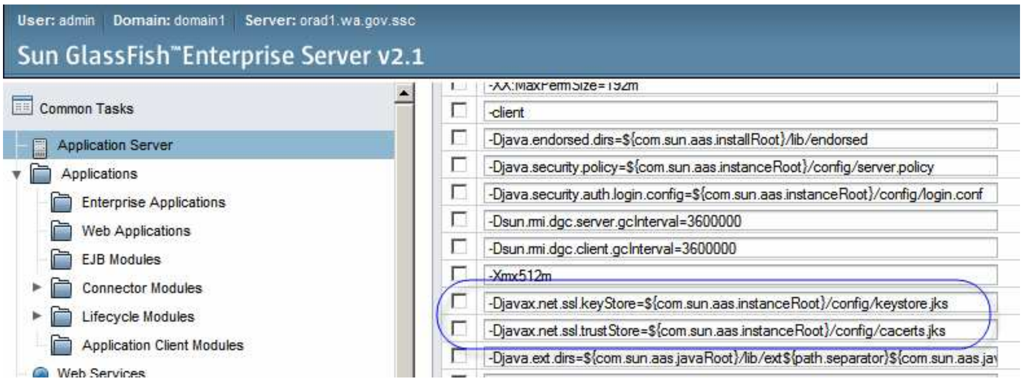
Options (41)

	Value
□	-Djavax.net.debug=ssl:handshake
□	-Dcm.sun.enterprise.web.connection.grizzly.enableSnoop=true
□	-Dcom.sun.xml.ws.assembler.client=true

Figure 5.8.1 ***JVM Options***

If you don’t have this incantation in the JVM Options at both end of the SSL Handshake you will not be able to see the log of the handshake. Please add the JVM Option to both instances of the GlassFish Application Server, if you are using two as I am doing. My local instance runs on mcz02.aus.sun.com and my remote instance runs on orad1.ssc.

When looking at the JVM Options also note the names and locations of the keystore.jks and cacerts.jks, see Figure 5.8.2. These are the cryptographic stores GlassFish uses at runtime. We will work with both in the not too distant future.

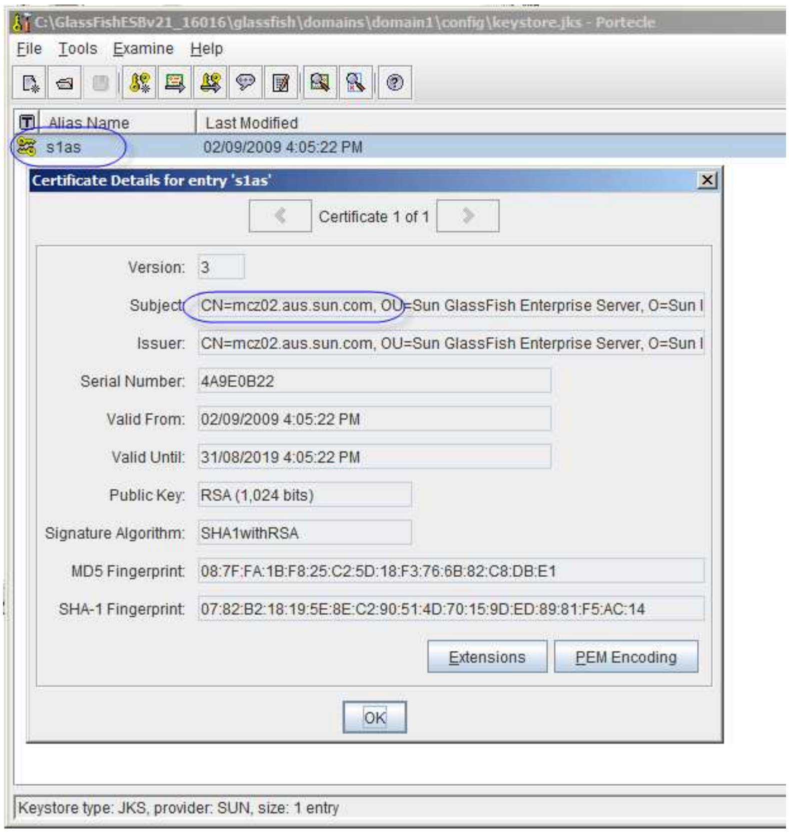


□	-XX:MaxPermSize=192m
□	-client
□	-Djava.endorsed.dirs=${com.sun.aas.installRoot}/lib/endorsed
□	-Djava.security.policy=${com.sun.aas.instanceRoot}/config/server.policy
□	-Djava.security.auth.login.config=${com.sun.aas.instanceRoot}/config/login.conf
□	-Dsun.rmi.dgc.server.gcInterval=3600000
□	-Dsun.rmi.dgc.client.gcInterval=3600000
□	-Xmx512m
□	-Djavax.net.ssl.keyStore=${com.sun.aas.instanceRoot}/config/keystore.jks
□	-Djavax.net.ssl.trustStore=${com.sun.aas.instanceRoot}/config/cacerts.jks
□	-Djava.ext.dirs=${com.sun.aas.javaRoot}/lib/ext${path.separator}${com.sun.aas.ja

Figure 5.8.2 ***Cryptographic objects stores***

Recall, or note, that at installation time the GlassFish installer generates the cryptographic key pair and the server certificate, which embeds the FQDN of the host on which it is being installed. This private key gets added to the keystore.jks, which by default resides in *<glassfishinstallroot >/domains/domain1/config*, under the alias *a1as*. When requested to provide a certificate, as is the case when SSL with Server-side Authentication is configured, the server will return its X.509 certificate as part of the SSL Handshake. The client/invoker is expected to use that certificate to verify whether it “trusts” the server enough to allow the SSL Handshake to succeed. The client verifies that the certificate is “trusted”, that is it is either signed by a trusted certification authority (CA) or it is explicitly trusted, if it is a self-signed certificate, by there being a copy of it in the client’s trust store, typically <glassfishinstallroot>/domains/domain1/config/cacerts.jks.

Using a tool like “Portecle Key Manager”, http://linux.softpedia.com/progDownload/Portecle-Download-3110.html, inspect the keystore.jks. Note the presence of the one and only private key, with the alias of s1as. Note, too, that its corresponding certificate is associated with the host mcz02.aus.sun.com - for you it will be the fully qualified name of the host on which you installed the GlassFish Application Server whose keystore,jks you are inspecting. Figure 5.8.3 illustrates this.


C:\GlassFishESBv21_16016\glassfish\domains\domain1\config\keystore.jks - Portecle
File
Tools
Examine
Help
Certificate Details for entry 's1as'
Certificate 1 of 1
Extensions
PEM Encoding
OK

Alias Name	Last Modified
s1as	02/09/2009 4:05:22 PM

Version: 3

Subject: CN=mcz02.aus.sun.com, OU=Sun GlassFish Enterprise Server, O=Sun I

Issuer: CN=mcz02.aus.sun.com, OU=Sun GlassFish Enterprise Server, O=Sun I

Serial Number: 4A9E0B22

Valid From: 02/09/2009 4:05:22 PM

Valid Until: 31/08/2019 4:05:22 PM

Public Key: RSA (1,024 bits)

Signature Algorithm: SHA1withRSA

MD5 Fingerprint: 08:7F:FA:1B:F8:25:C2:5D:18:F3:76:6B:82:C8:DB:E1

SHA-1 Fingerprint: 07:82:B2:18:19:5E:8E:C2:90:51:4D:70:15:9D:ED:89:81:F5:AC:14

Keystore type: JKS, provider: SUN, size: 1 entry

Figure 5.8.3 ***Private key, s1as, and its certificate***

You can surmise form this that if mcz02.aus.sun.com is the server hosting the PersonSvc web service, and it is required to provide the client with a certificate, the certificate associated with the alias a1as will be provided to the client.

Note, also, that the Issuer of the certificate is the same as the Subject of the certificate in Figure 5.8.3. That makes this certificate a self-signed certificate. There is no separate Certification Authority which issued the certificate so it is unlikely that this certificate will be trusted by any other host unless explicitly told to do so.

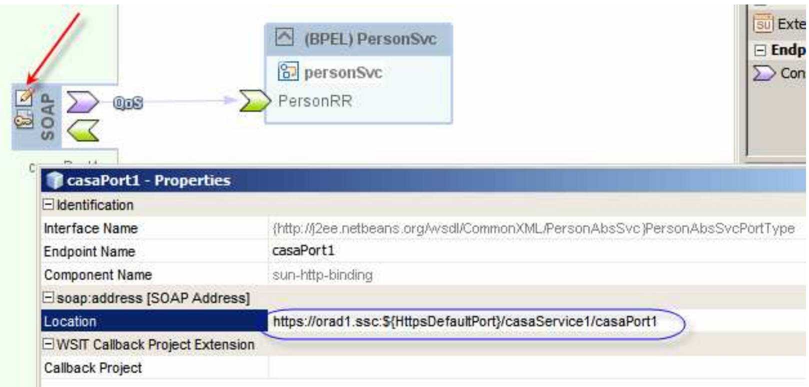
Let’s interrupt the certificate discussion at this point. We will resume it later.

Create the PersonSvc_CA_SSLServerAuth Composite Application, drag the PersonSvc BPEL Module onto the CASA canvas, add and connect a SOAP binding and Build, much as was illustrated in Figures 5.7.1, 5.7.2 and 5.7.3.

Click on the “pencil and paper” icon to open properties of the SOAP BC and modify Location URL to a) use the https scheme instead of the http scheme, b) use the FQDN of the remote host (for me this will be orad1.ssc) and c) modify port number variable name from HttpDefaultPort to HttpsDefaultPort. For me, the modified URL will be:

```
https://orad1.ssc:${HttpsDefaultPort}/casaService1/casaPort1
```

Figure 5.8.4 illustrates the key points.

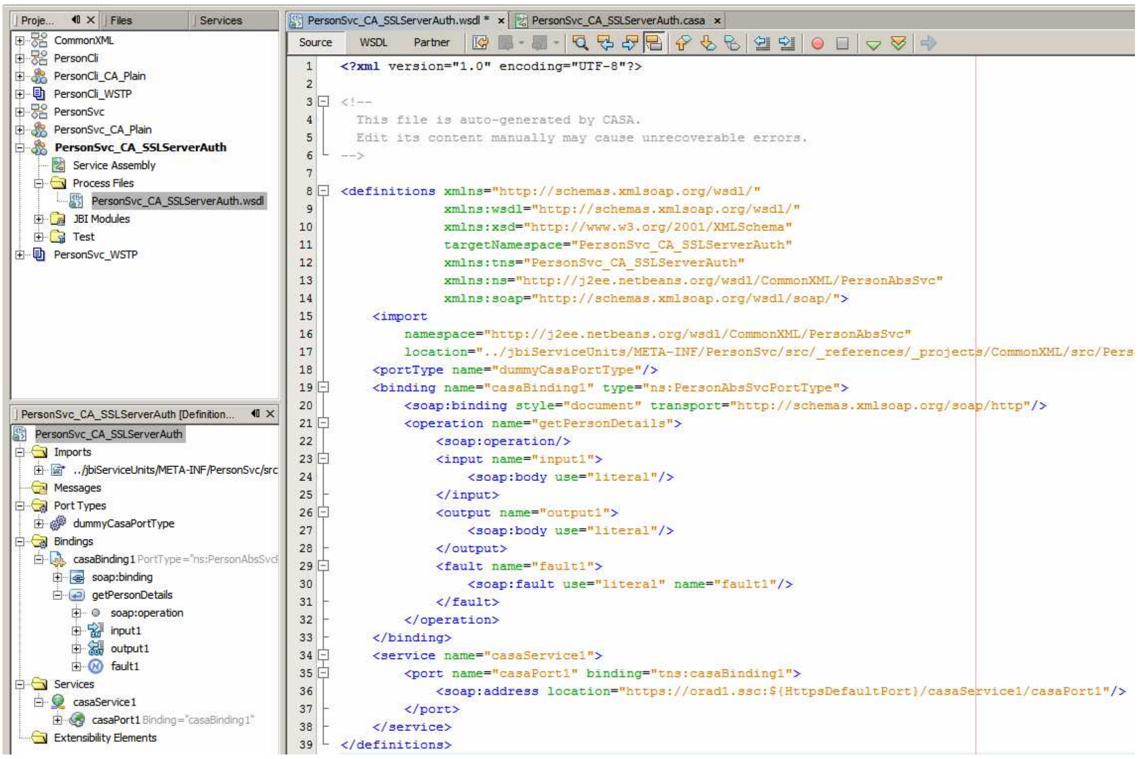


casaPort1 - Properties

Identification	
Interface Name	{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}PersonAbsSvcPortType
Endpoint Name	casaPort1
Component Name	sun-http-binding
soap:address [SOAP Address]	
Location	https://orad1.ssc:${HttpsDefaultPort}/casaService1/casaPort1
WSIT Callback Project Extension	
Callback Project	

Figure 5.8.4 *Modified Location property*

Here comes a twist. The original WSDL does not use any security policies at all. In fact it can not because most security policies are applied to the concrete part of he WSDL and our original WSDL does not have a concrete part. By dragging the SOAP BC onto the CASAS canvas and connecting it to the BPEL Module we created a WSDL which imports our original WSDL and adds the concrete part. Explore the PersonSvc_CA_SSLServerAuth -> Process Files, Figure 5.8.5, and note the WSDL PersonSvc_CA_SSLServerAuth. Open this WSDL and look at the concrete part, Figure 5.8.5.



```
<?xml version="1.0" encoding="UTF-8"?>

<!--
  This file is auto-generated by CASA.
  Edit its content manually may cause unrecoverable errors.
-->

<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             targetNamespace="PersonSvc_CA_SSLServerAuth"
             xmlns:tns="PersonSvc_CA_SSLServerAuth"
             xmlns:ns="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
    <import
        namespace="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc"
        location="../jbiServiceUnits/META-INF/PersonSvc/src/_references/_projects/CommonXML/src/Pers
    <portType name="dummyCasaPortType"/>
    <binding name="casaBinding1" type="ns:PersonAbsSvcPortType">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="getPersonDetails">
            <soap:operation/>
            <input name="input1">
                <soap:body use="literal"/>
            </input>
            <output name="output1">
                <soap:body use="literal"/>
            </output>
            <fault name="fault1">
                <soap:fault use="literal" name="fault1"/>
            </fault>
        </operation>
    </binding>
    <service name="casaService1">
        <port name="casaPort1" binding="tns:casaBinding1">
            <soap:address location="https://orad1.ssc:${HttpsDefaultPort}/casaService1/casaPort1"/>
        </port>
    </service>
</definitions>
```

Figure 5.8.5 *Concrete WSDL*

Note that the imported WSDL location is relative to the location of this WSDL. At build time NetBeans will be able to resolve this but at runtime it will not. To make sure the project can deploy successfully we need to “import” the abstract WSDL from the CommonXML Project to this project’s “Process Files” folder. Right-click on the “process File” folder, choose “New” -> “External WSDL Document(s)”, locate the WSDL in the CommoXML/src folder, select it and import it into the CASA project. This should not be required but … Figure 5.8.9 shows the project structure after this activity.

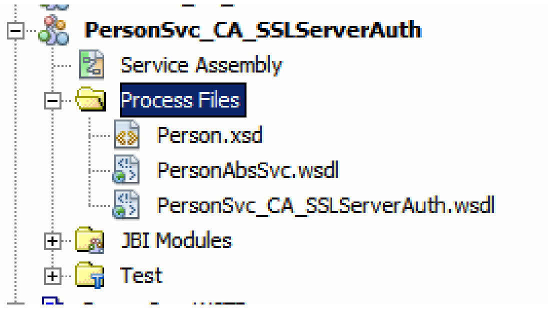


Figure 5.8.9 *Project structure with abstract WSDL and XSD “imported”*

Switch back to the CASA canvas, click the “paper with a key” icon and choose “Server Configuration”. Figure 5.8.10 illustrates this step.

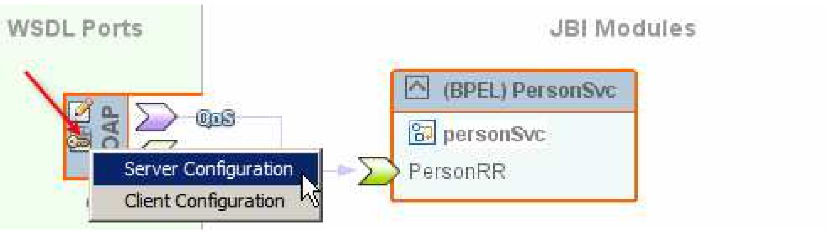
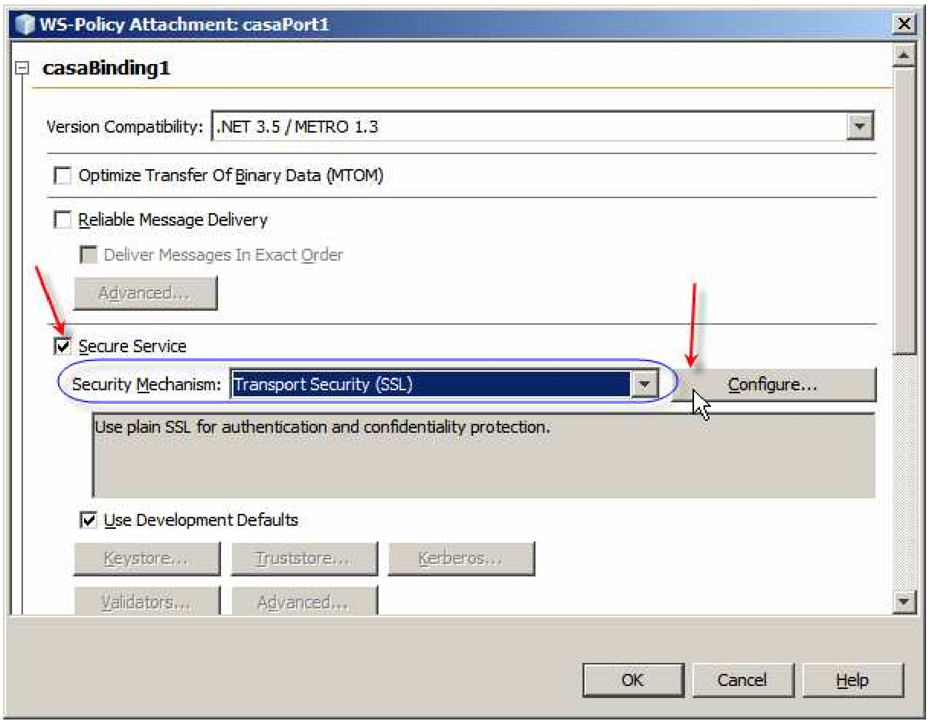


Figure 5.8.10 *Edit Server Configuration*

Check the “Secure Service” checkbox, choose “Transport Security (SSL)” from the Security Mechanism drop-down and click “Configure”, as shown in Figure 5.8.11.



WS-Policy Attachment: casaPort1

casaBinding1

Version Compatibility: .NET 3.5 / METRO 1.3

☐ Optimize Transfer Of Binary Data (MTOM)

☐ Reliable Message Delivery

☐ Deliver Messages In Exact Order

☑ Secure Service

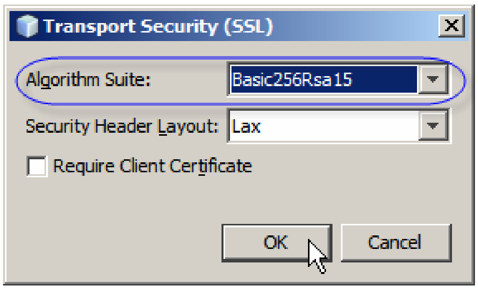
Security Mechanism: Transport Security (SSL)

Use plain SSL for authentication and confidentiality protection.

☑ Use Development Defaults

Figure 5.8.11 *Enable and configure SSL / TLS channel security*

Choose a suitable algorithm suite from the dropdown of supported algorithm suites. This selection does not guarantee that the specified algorithm suite will be used. Final algorithm suite is subject to negotiation between the client and the server and is settled during the SSL Handshake. Leave "Require Client Certificate" checkbox unchecked. We are configuring Server-side Authentication here so we don't need client's certificate. Figure 5.8.12 illustrates the dialogue box.



Transport Security (SSL)

Algorithm Suite: Basic256Rsa15

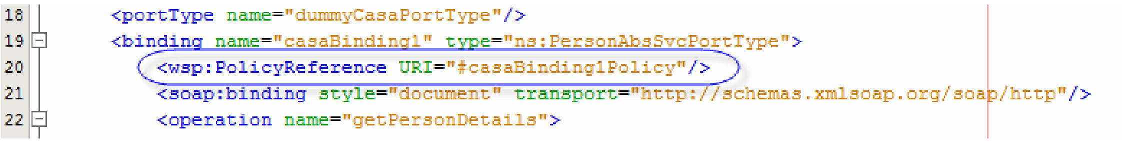
Security Header Layout: Lax

☐ Require Client Certificate

Figure 5.8.12 *Choose algorithm suite*

Dismiss the dialog and look again at the PersonSvc_CA_SSLServerAuth WSDL. Switch to the Source view and inspect the policy formulation.

Note, on line 20, that a policy reference attribute, shown in Figure 5.8.13, was added to the Biding.

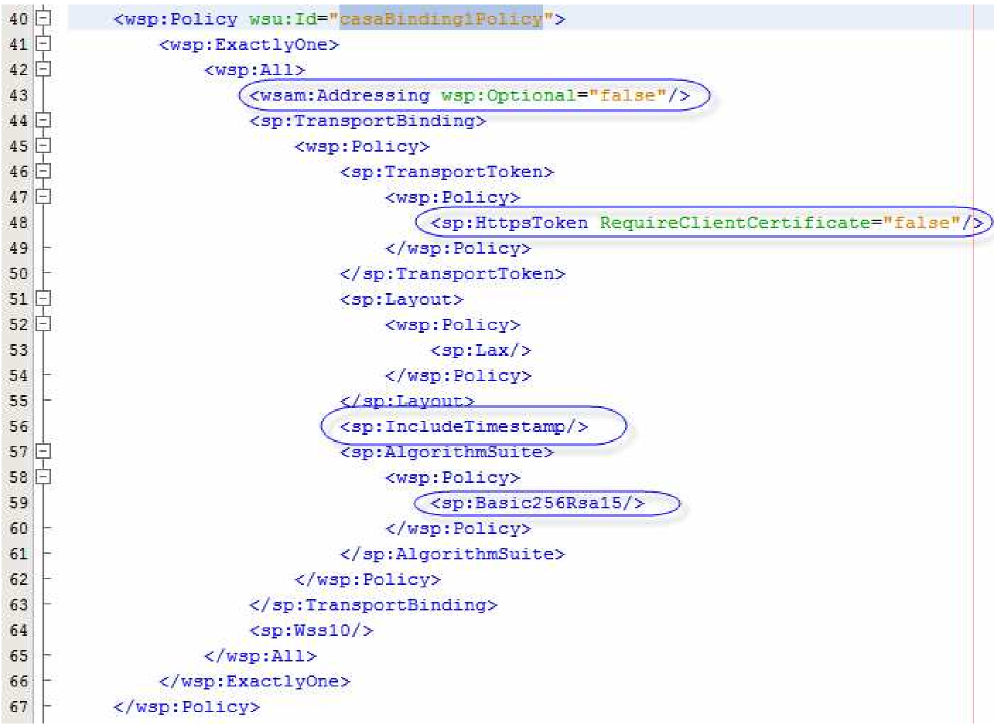


```
    <portType name="dummyCasaPortType"/>
    <binding name="casaBinding1" type="ns:PersonAbsSvcPortType">
        <wsp:PolicyReference URI="#casaBinding1Policy"/>
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="getPersonDetails">
```

Figure 5.8.13 *Policy Reference in the Binding*

Were we to remove this attribute and its value no policy would be applied to the binding, even if one was there in the WSDL.

Scroll down and note the following, called out in Figure 5.8.14:



```
40  <wsp:Policy wsu:Id="casaBinding1Policy">
41      <wsp:ExactlyOne>
42          <wsp:All>
43              <wsam:Addressing wsp:Optional="false"/>
44              <sp:TransportBinding>
45                  <wsp:Policy>
46                      <sp:TransportToken>
47                          <wsp:Policy>
48                              <sp:HttpsToken RequireClientCertificate="false"/>
49                          </wsp:Policy>
50                      </sp:TransportToken>
51                      <sp:Layout>
52                          <wsp:Policy>
53                              <sp:Lax/>
54                          </wsp:Policy>
55                      </sp:Layout>
56                      <sp:IncludeTimestamp/>
57                      <sp:AlgorithmSuite>
58                          <wsp:Policy>
59                              <sp:Basic256Rsa15/>
60                          </wsp:Policy>
61                      </sp:AlgorithmSuite>
62                  </wsp:Policy>
63              </sp:TransportBinding>
64              <sp:Wss10/>
65          </wsp:All>
66      </wsp:ExactlyOne>
67  </wsp:Policy>
```

Figure 5.8.14 *Policy elements of special interest*

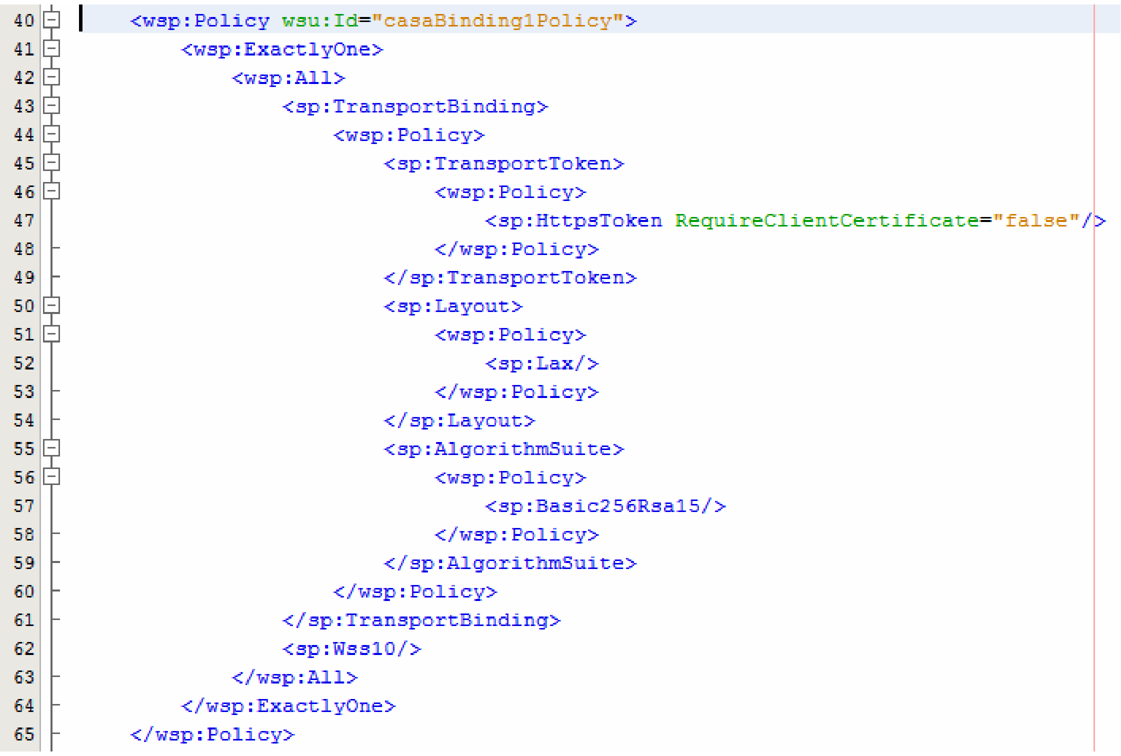
- Line 43: <wsam:Addressing wsp:Optional="false"/>
  One would expect this to mean “WS-Addressing is mandatory”. The presence of the wsp:Optional=”false” attribute make the implementation ignore WS-Addressing altogether. WS-Addressing is not required for SSL and is unrelated to SSL so let’s get rid of this element.

- Line 48: <sp:HttpsToken RequireClientCertificate="false"/>
  This attribute indicates that Server-side authentication is used – no client certificate is required.

- Line 56: <sp:IncludeTimestamp/>
  This element requires addition of a timestamp token. Timestamp token is not related to or required by SSL so delete this element.

- Line 59: <sp:Basic256Rsa15/>
  This specified the preferred algorithm suite

The final policy, after removal of addressing and timestamp elements, is shown in Figure 5.8.15.



```
<wsp:Policy wsu:Id="casaBinding1Policy">
    <wsp:ExactlyOne>
        <wsp:All>
            <sp:TransportBinding>
                <wsp:Policy>
                    <sp:TransportToken>
                        <wsp:Policy>
                            <sp:HttpsToken RequireClientCertificate="false"/>
                        </wsp:Policy>
                    </sp:TransportToken>
                    <sp:Layout>
                        <wsp:Policy>
                            <sp:Lax/>
                        </wsp:Policy>
                    </sp:Layout>
                    <sp:AlgorithmSuite>
                        <wsp:Policy>
                            <sp:Basic256Rsa15/>
                        </wsp:Policy>
                    </sp:AlgorithmSuite>
                </wsp:Policy>
            </sp:TransportBinding>
            <sp:Wss10/>
        </wsp:All>
    </wsp:ExactlyOne>
</wsp:Policy>
```

Figure 5.8.15 *Final policy*

Save and close the modified WSDL and Build, but do not Deploy the project.

This project will be deployed to the remote instance of the GlassFish Application Server – for me orad1.ssc. Before we can deploy the project we need to add the GlassFish instance to the NetBeans IDE so it can address it at deployment time. Switch to the Services Tab in the project explorer, right-click on the Servers node and choose Add Server. Figure 5.8.16 illustrates this. If you already have the remote server in the list, skip this.

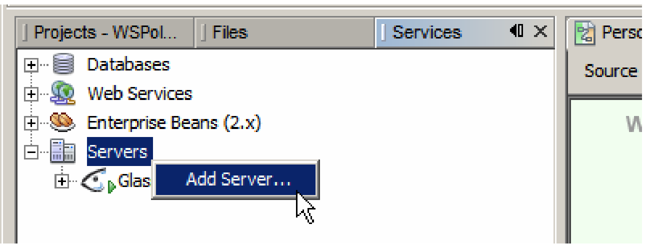
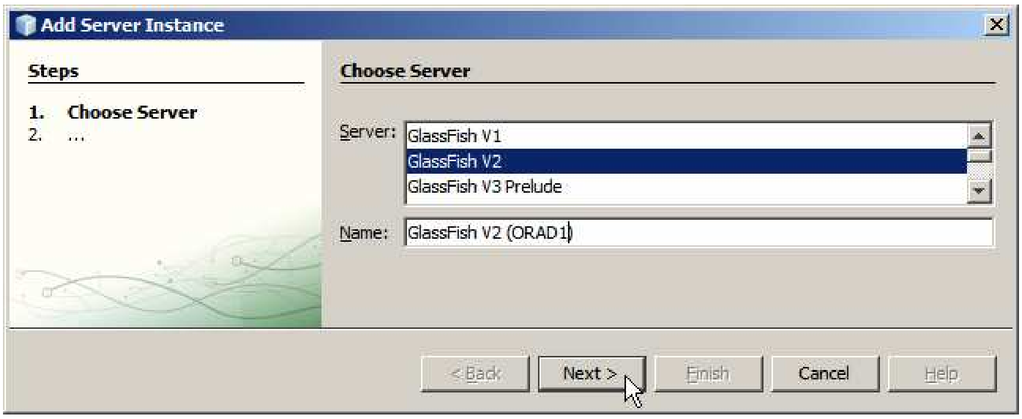


Figure 5.8.16 *Add Server …*

Choose GlassFish v2, modify the name to reflect host name and click Next, as shown in Figure 5.8.17.



Add Server Instance

Steps

1. **Choose Server**
2. ...

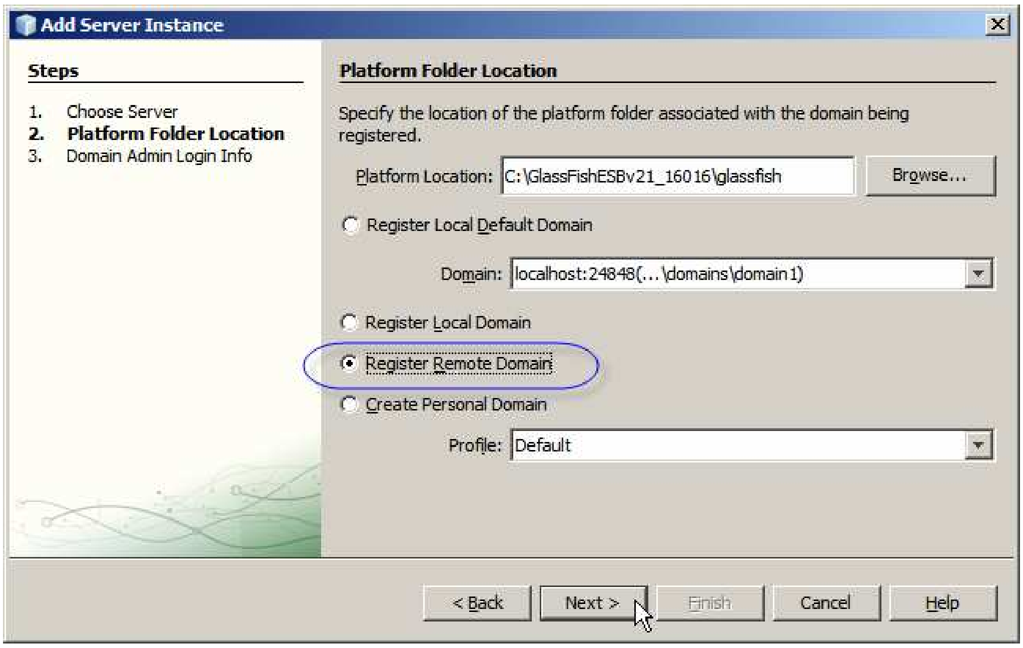
Choose Server

Server: GlassFish V1 / GlassFish V2 / GlassFish V3 Prelude

Name: GlassFish V2 (ORAD1)

Figure 5.8.17 *Choose server type and name it*

Choose "Register Remote Domain" and click Next. Figure 5.8.18 illustrates key points.



Add Server Instance

Steps

1. Choose Server
2. **Platform Folder Location**
3. Domain Admin Login Info

Platform Folder Location

Specify the location of the platform folder associated with the domain being registered.

Platform Location: C:\GlassFishESBv21_16016\glassfish

○ Register Local Default Domain

Domain: localhost:24848(...\domains\domain1)

○ Register Local Domain

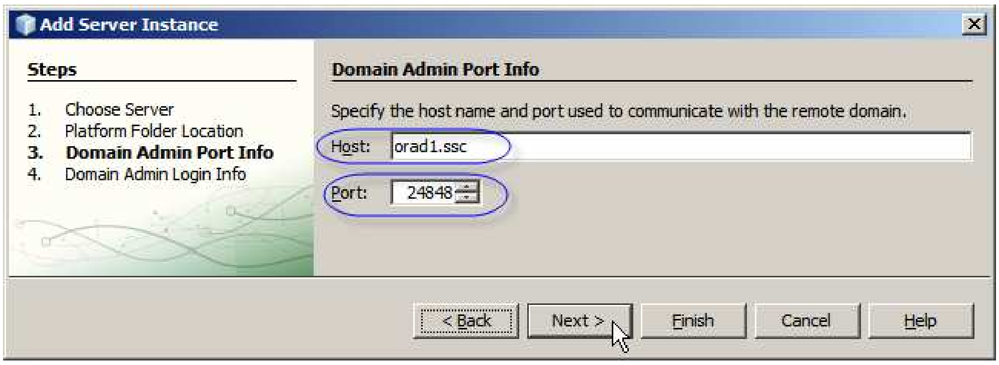
● Register Remote Domain

○ Create Personal Domain

Profile: Default

Figure 5.8.18 *Choose to Register Remote Domain*

Enter FQDN of the remote host, as specified for the server certificate when the remote GlassFish instance was installed, specify the appropriate administrative port number, if different from default, and click Next. Figure 5.8.19 illustrates this.



Add Server Instance

Steps

1. Choose Server
2. Platform Folder Location
3. **Domain Admin Port Info**
4. Domain Admin Login Info

Domain Admin Port Info

Specify the host name and port used to communicate with the remote domain.

Host: orad1.ssc

Port: 24848

Figure 5.8.19 *Specify host and port*

Provide credentials and Finish.

If the remote GlassFish instance is running and correct configuration information was provided to NetBeans, the GlassFish instance will appear in the list of servers, as shown in Figure 5.8.20.

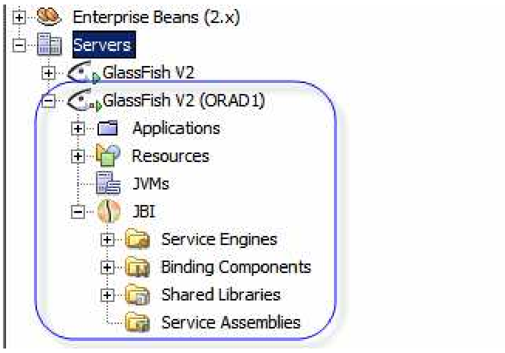
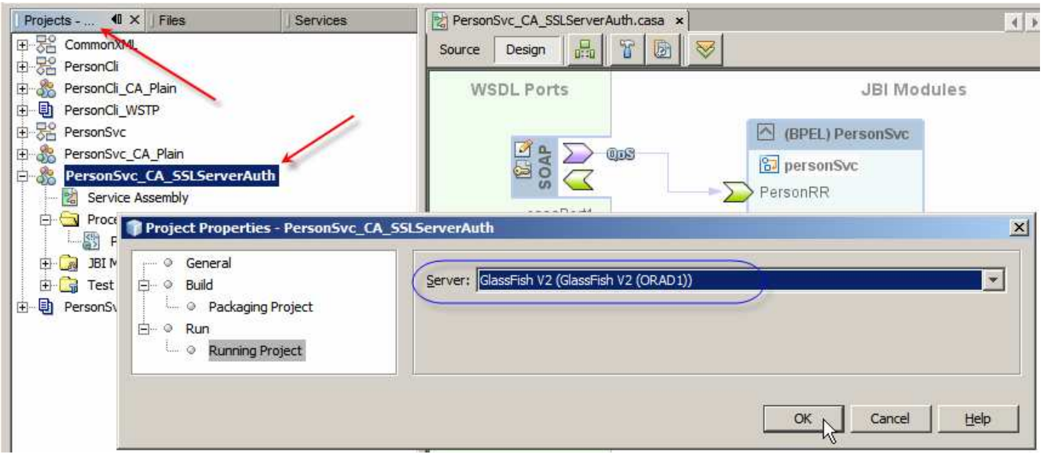


Figure 5.8.20 *Remote GlassFish instance in NetBeans*

Switch back to the Project Explorer's Project tab, right-click on the name PersonSvc_CA_SSLServerAuth, choose Properties, click the "Running Project" property and select the remote GlassFish instance as the deployment target. Figure 5.8.21 illustrates key points.



Project Properties - PersonSvc_CA_SSLServerAuth

Server: GlassFish V2 (GlassFish V2 (ORAD1))

Figure 5.8.21 *Choose the remote GlassFish instance as the deployment target*

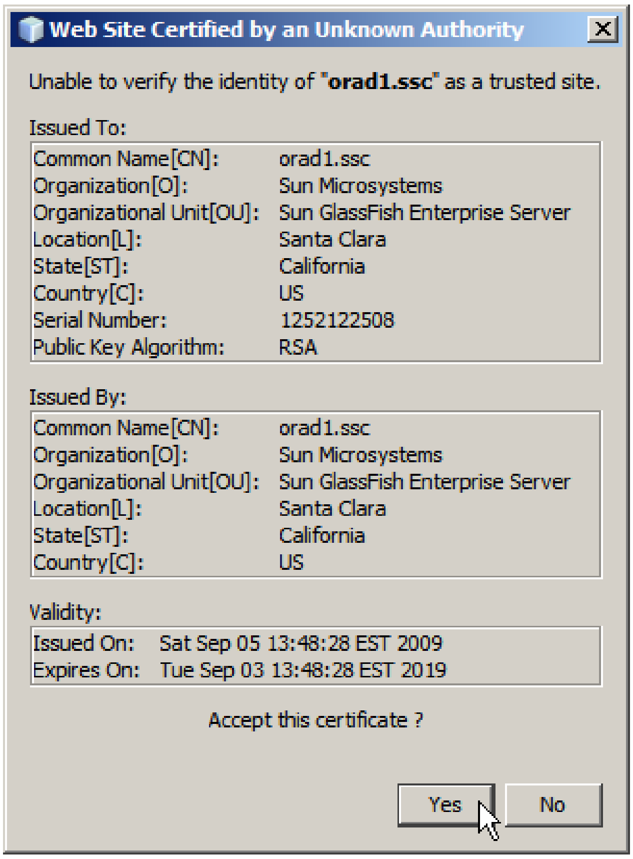
Build the project and Deploy it.

Because this is a project that requires SSL with Server-side Authentication we can use the SoapUI plugin to test the service and observe SSL Handshake at the server side. We will implement and exercise the PersonCli SSL with Server-side Authenticatin project a little later.

Let’s create a “New Project” -> “Java EE” -> “Web Service Testing Project”, named PersonSvc_SSLServerAuth_WSDP, using the WSDL location from the CASA SOAP BC’s Location Property, replacing the ${HttpsDefaultPort} with the appropriate port number, For me this will be:

```
https://orad1.ssc:29181/casaService1/casaPort1?WSDL
```

As the project is created, a dialogue box my pop up asking you to accept remote GlassFish instance’s certificate, similar to what I saw for orad1, Figure 5.8.22.



Web Site Certified by an Unknown Authority

Unable to verify the identity of "**orad1.ssc**" as a trusted site.

Issued To:

Common Name[CN]:	orad1.ssc
Organization[O]:	Sun Microsystems
Organizational Unit[OU]:	Sun GlassFish Enterprise Server
Location[L]:	Santa Clara
State[ST]:	California
Country[C]:	US
Serial Number:	1252122508
Public Key Algorithm:	RSA

Issued By:

Common Name[CN]:	orad1.ssc
Organization[O]:	Sun Microsystems
Organizational Unit[OU]:	Sun GlassFish Enterprise Server
Location[L]:	Santa Clara
State[ST]:	California
Country[C]:	US

Validity:

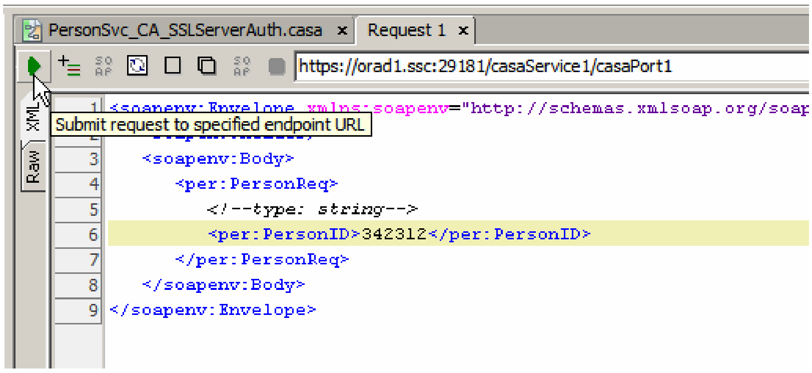
Issued On:	Sat Sep 05 13:48:28 EST 2009
Expires On:	Tue Sep 03 13:48:28 EST 2019

Accept this certificate ?

Figure 5.8.22 *Accept remote GlassFish instance’s certificate*

This will happen once, the first time a reference is made to the remote host. Thereafter NetBeans will trust the certificate and will not ask for confirmation.

Once the project is created add a New Request to the getPersonDetai;ls interface, modify PersonID to 342312 and submit the request. Figure 5.8.23 illustrates the request.

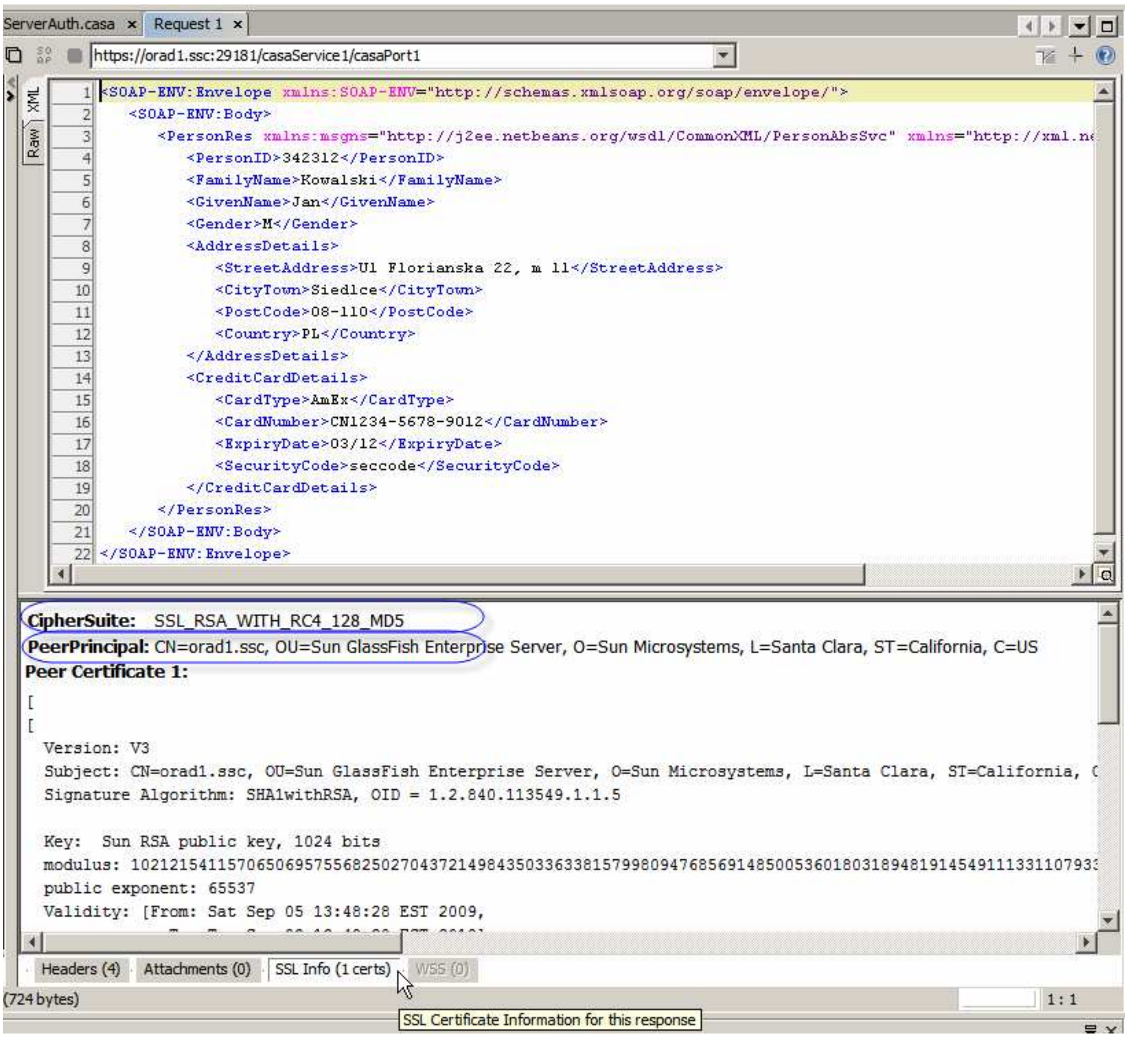

PersonSvc_CA_SSLServerAuth.casa
Request 1
XML
Raw
Submit request to specified endpoint URL

https://orad1.ssc:29181/casaService1/casaPort1

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap
   <soapenv:Body>
      <per:PersonReq>
         <!--type: string-->
         <per:PersonID>342312</per:PersonID>
      </per:PersonReq>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 5.8.23 *SOAP Request*

Observe a SOAP Response response. Click on the "SSL Info" tab and observe the orad1.ssc's certificate, figure 5.8.24.



https://orad1.ssc:29181/casaService1/casaPort1

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <PersonRes xmlns:msgns="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc" xmlns="http://xml.ne
         <PersonID>342312</PersonID>
         <FamilyName>Kowalski</FamilyName>
         <GivenName>Jan</GivenName>
         <Gender>M</Gender>
         <AddressDetails>
            <StreetAddress>Ul Florianska 22, m 11</StreetAddress>
            <CityTown>Siedlce</CityTown>
            <PostCode>08-110</PostCode>
            <Country>PL</Country>
         </AddressDetails>
         <CreditCardDetails>
            <CardType>AmEx</CardType>
            <CardNumber>CN1234-5678-9012</CardNumber>
            <ExpiryDate>03/12</ExpiryDate>
            <SecurityCode>seccode</SecurityCode>
         </CreditCardDetails>
      </PersonRes>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

CipherSuite: SSL_RSA_WITH_RC4_128_MD5

PeerPrincipal: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa Clara, ST=California, C=US

Peer Certificate 1:

```
[
[
  Version: V3
  Subject: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa Clara, ST=California, C
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus: 1021215411570650695755682502704372149843503363381579980947685691485005360180318948191454911133110793
  public exponent: 65537
  Validity: [From: Sat Sep 05 13:48:28 EST 2009,
```

Figure 5.8.24 *SOAP Response and orad1.ssc's certificate*

The SSL Handshake was successful. Let's look at selected lines from the server.log of the remote GlassFish instance to see how the SSL Handshake looked like there. Listing 5.8.1 shows just key lines.

Listing 5.8.1 *Key lines from the SSL Handshake log*

```
[#|2009-09-07T13:51:55.324+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=25;_ThreadName=SelectorThread-
29181;|
Using SSLEngineImpl.|#]
[#|2009-09-07T13:51:55.326+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
httpSSLWorkerThread-29181-1, READ:  SSL v2, contentType = Han
149|#]
[#|2009-09-07T13:51:55.327+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
*** ClientHello, TLSv1|#]
…
[#|2009-09-07T13:51:55.337+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID
ad-29181-1;|
Session ID:  |#]
[#|2009-09-07T13:51:55.337+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|{}|#]
[#|2009-09-07T13:51:55.337+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
Cipher Suites: [SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_ECDH_ECDSA_WITH_RC4_128_SHA,
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDH_RSA_WITH_RC4_128_SHA,
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_ECDSA_WITH_RC4_128_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_RC4_128_SHA,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA,
SSL_RSA_WITH_DES_CBC_SHA, SSL_DHE_RSA_WITH_DES_CBC_SHA, SSL_DHE_DSS_WITH_DES_CBC_SHA,
SSL_RSA_EXPORT_WITH_RC4_40_MD5, SSL_RSA_EXPORT_WITH_DES40_CBC_SHA,
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA, SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA,
SSL_RSA_WITH_NULL_MD5, SSL_RSA_WITH_NULL_SHA, TLS_ECDH_ECDSA_WITH_NULL_SHA,
TLS_ECDH_RSA_WITH_NULL_SHA, TLS_ECDHE_ECDSA_WITH_NULL_SHA, TLS_ECDHE_RSA_WITH_NULL_SHA,
SSL_DH_anon_WITH_RC4_128_MD5, TLS_DH_anon_WITH_AES_128_CBC_SHA,
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA, SSL_DH_anon_WITH_DES_CBC_SHA,
TLS_ECDH_anon_WITH_RC4_128_SHA, TLS_ECDH_anon_WITH_AES_128_CBC_SHA,
TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA, SSL_DH_anon_EXPORT_WITH_RC4_40_MD5,
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA, TLS_ECDH_anon_WITH_NULL_SHA,
TLS_KRB5_WITH_RC4_128_SHA, TLS_KRB5_WITH_RC4_128_MD5, TLS_KRB5_WITH_3DES_EDE_CBC_SHA,
TLS_KRB5_WITH_3DES_EDE_CBC_MD5, TLS_KRB5_WITH_DES_CBC_SHA, TLS_KRB5_WITH_DES_CBC_MD5,
TLS_KRB5_EXPORT_WITH_RC4_40_SHA, TLS_KRB5_EXPORT_WITH_RC4_40_MD5,
TLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA, TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5]|#]
[#|2009-09-07T13:51:55.338+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_T                                     e
ad-29181-1;|
Compression Methods:  { |#]
[#|2009-09-07T13:51:55.338+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|0|#]
…
```

SSL Engine starts processing

Start SSL Handshake

Client Hello message received

No Session ID – need new session

Client is willing to accept any of the cipher suites listed

Client will not use compression

Crteate new session

```
[#|2009-09-07T13:51:55.338+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
%% Created:  [Session-19, SSL_RSA_WITH_RC4_128_MD5]|#]
```

Server sends Hello message

```
[#|2009-09-07T13:51:55.339+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
*** ServerHello, TLSv1|#]
…
```

Session ID generated

```
[#|2009-09-07T13:51:55.349+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
Session ID:  |#]
```

Server chose the cipher suite

```
[#|2009-09-07T13:51:55.349+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|{74, 164, 131, 91, 63, 222, 251, 80, 243, 87, 244, 51, 122, 138, 49, 114, 24,
244, 67, 8, 250, 124, 74, 146, 191, 69, 3, 249, 26, 3, 159, 81}|#]
[#|2009-09-07T13:51:55.350+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
Cipher Suite: SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-07T13:51:55.350+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
Compression Method: 0|#]
[#|2009-09-07T13:51:55.350+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
***|#]
```

Server sends its certificate

```
[#|2009-09-07T13:51:55.350+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
Cipher suite:  SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-07T13:51:55.350+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
*** Certificate chain|#]
[#|2009-09-07T13:51:55.351+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
chain [0] = [
[
  Version: V3
  Subject: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems,
L=Santa Clara, ST=California, C=US
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus:
102121541157065069575568250270437214984350336338157998094768569148500536018031894819145491
113311079336949399472032140208618356816163637202618727298710465300230334324506184949208474
446933178623253726110968175800084405517043740120715529443624373833850530287673442674333181
40807253127294850333975872494855098 2873
  public exponent: 65537
  Validity: [From: Sat Sep 05 13:48:28 EST 2009,
               To: Tue Sep 03 13:48:28 EST 2019]
  Issuer: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  SerialNumber: [    4aa1df8c]

Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 9E 79 9C E9 59 86 34 8F   FD 75 09 F7 82 D0 82 CE  .y..Y.4..u......
0010: BE 9A 44 EE                                        ..D.
]
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 23 A7 FD 51 1F 81 9E 8C   34 3A 58 01 EF 5A 04 CD  #..Q....4:X..Z..
0010: AD 35 2C 67 17 40 3A B8   EA 19 37 DB B2 B3 C8 EA  .5,g.@:...7.....
0020: 5B 4F 0E 30 4E 9D 42 23   52 FE E8 53 44 8B 64 21  [O.0N.B#R..SD.d!
0030: CF 5F EE 07 D5 60 1E F2   1B EA 68 99 E4 BB 6C 89  ._...`....h...l.
0040: 02 21 1D A5 AE 6C 26 14   8C 92 02 92 E3 C1 74 56  .!...l&.......tV
0050: 6A 69 96 8E 2D 1E 7D 6C   52 5E 99 38 20 8B 19 C4  ji..-..lR^.8 ...
0060: 52 11 89 B3 73 D0 6C 61   B2 DB BF CA 58 0A 3A 5D  R...s.la....X.:]
0070: 40 81 97 CC 3F 60 A6 1E   B5 D6 60 8A C6 6B B6 F6  @...?`....`..k..

]|#]
 [#|2009-09-07T13:51:55.352+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
*** ServerHelloDone|#]
```

Hello exchange done, perform secrets exchange

```
[#|2009-09-07T13:51:55.352+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
httpSSLWorkerThread-29181-1, WRITE: TLSv1 Handshake, length = 794|#]
[#|2009-09-07T13:51:55.524+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
httpSSLWorkerThread-29181-1, READ: TLSv1 Handshake, length = 134|#]
[#|2009-09-07T13:51:55.527+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
*** ClientKeyExchange, RSA PreMasterSecret, TLSv1|#]
[#|2009-09-07T13:51:55.528+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
SESSION KEYGEN:|#]
[#|2009-09-07T13:51:55.528+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
PreMaster Secret:|#]
[#|2009-09-07T13:51:55.528+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
0000: |#]
[#|2009-09-07T13:51:55.528+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|03 |#]
…
[#|2009-09-07T13:51:55.543+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
CONNECTION KEYGEN:|#]
[#|2009-09-07T13:51:55.543+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
Client Nonce:|#]
[#|2009-09-07T13:51:55.544+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
0000: |#]
…
```

```
[#|2009-09-07T13:51:55.554+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
Server Nonce:|#]
[#|2009-09-07T13:51:55.554+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
0000: |#]
[#|2009-09-07T13:51:55.554+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|4A |#]
…
[#|2009-09-07T13:51:55.564+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
Master Secret:|#]
[#|2009-09-07T13:51:55.564+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
0000: |#]
[#|2009-09-07T13:51:55.564+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|6F |#]
…
[#|2009-09-07T13:51:55.577+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|Client MAC write Secret:|#]
[#|2009-09-07T13:51:55.577+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
0000: |#]
[#|2009-09-07T13:51:55.577+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|C3 |#]
…
[#|2009-09-07T13:51:55.582+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
Server MAC write Secret:|#]
[#|2009-09-07T13:51:55.582+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
0000: |#]
[#|2009-09-07T13:51:55.583+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|6F |#]
…
[#|2009-09-07T13:51:55.587+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|Client write key:|#]
[#|2009-09-07T13:51:55.587+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
0000: |#]
[#|2009-09-07T13:51:55.587+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|C2 |#]
…
[#|2009-09-07T13:51:55.592+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|Server write key:|#]
[#|2009-09-07T13:51:55.592+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
0000: |#]
[#|2009-09-07T13:51:55.593+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|BF |#]
…
[#|2009-09-07T13:51:55.598+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|... no IV used for this cipher|#]
[#|2009-09-07T13:51:55.598+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
httpSSLWorkerThread-29181-1, READ: TLSv1 Change Cipher Spec, length = 1|#]
[#|2009-09-07T13:51:55.599+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
httpSSLWorkerThread-29181-1, READ: TLSv1 Handshake, length = 32|#]
[#|2009-09-07T13:51:55.599+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
*** Finished|#]
[#|2009-09-07T13:51:55.599+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
verify_data:  { |#]
[#|2009-09-07T13:51:55.599+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|58|#]
…
[#|2009-09-07T13:51:55.603+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
httpSSLWorkerThread-29181-1, WRITE: TLSv1 Change Cipher Spec, length = 1|#]
[#|2009-09-07T13:51:55.604+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
*** Finished|#]
[#|2009-09-07T13:51:55.604+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
verify_data:  { |#]
[#|2009-09-07T13:51:55.604+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|220|#]
…
[#|2009-09-07T13:51:55.609+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
httpSSLWorkerThread-29181-1, WRITE: TLSv1 Handshake, length = 32|#]
[#|2009-09-07T13:51:55.609+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=35;_ThreadName=httpSSLWorkerThre
ad-29181-1;|
%% Cached server session: [Session-19, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-07T13:51:55.662+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=36;_ThreadName=httpSSLWorkerThre
ad-29181-0;|
---[HTTP request]---|#]
…
```

Successful SSL Handshake begin data transfer

The listing shows key points in the SSL Handshake, as seem at the server. The client view will be looked at shortly. The SoapUI plugin does not produce a log I can see, however it received the server certificate as can be seen in Figure 5.8. SoapUI plugin did not object to the certificate as not being trusted. It seems that SoapUI will accept any certificate. It is a testing tool, after all, not an application used to exchange real data.

Let's now create a composite application, PersonCli_CA_SSLServerAuth, for the client side. Drag the PersonCli BPEL module onto the CASA canvas and click Build. Right-click on the name of the project and create a "New" -> "External WSDL Document(s)", providing the endpoint URL exposed by the PersonSvc_CA_SSLServerAuth composite application, with the suffix "?WSDL". Accept the thertificate if the dialog box, like that shown in Figure 5.8.22 pops up.

Right-click on the WSDL Ports swim line and choose "Load WSDL Port". Accept the one and only port and click Build to build the project. These steps are illustrated in Figures 5.7.8 through 5.7.14.

Click on the "paper and key" icon on the SOAP BC and choose Client Configuration, as shown in Figure 5.8.25.

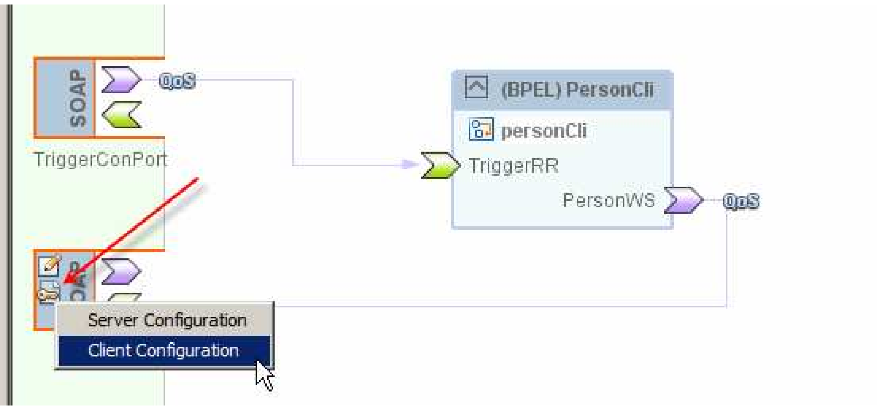
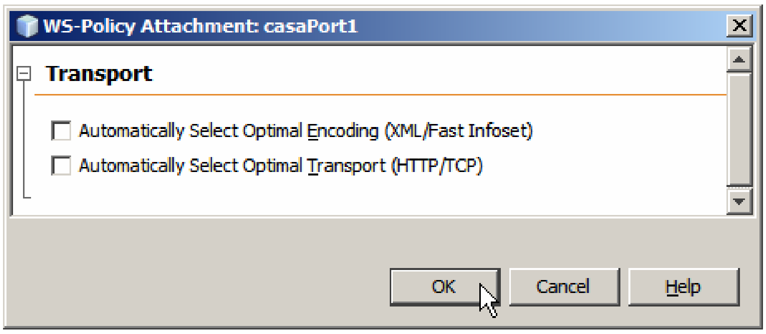


Figure 5.8.25 *Client policy configuration*

Note, as shown in Figure 5.8.26, that there are no specific configuration options for SSL with Server-side Authentication.



□ Automatically Select Optimal Encoding (XML/Fast Infoset)
□ Automatically Select Optimal Transport (HTTP/TCP)

Figure 5.8.26 *No SSL configuration options at the client side*

Build the project.

Attempt to deploy the project to the local GlassFish instance. The expectation is that deployment will fail with an unenlightening message like that shown in Listing 5.8.2.

Listing 5.8.2 Deployment error messages

```
ERROR: Successful execution of Deploy:
G:\GlassFishESBv21Projects\WSPolicyExploration\PersonCli_CA_SSLServerAuth/dist/P
ersonCli_CA_SSLServerAuth.zip
WARNING: (JBIMA0404) Deployment of service assembly PersonCli_CA_SSLServerAuth
succeeded partially; some service units failed to deploy.
    * Component: sun-http-binding
      ERROR: (SOAPBC_DEPLOY_2) HTTPBC-E00201: Deployment failed.
javax.wsdl.WSDLException: WSDLException (at /definitions/import):
faultCode=OTHER_ERROR: Unable to resolve imported document at
''https://orad1.ssc:29181/PersonSvc_CA_SSLServerAuth-sun-http-
binding/PersonAbsSvc.wsdl'', relative to
''file:/C:/GlassFishESBv21_16016/glassfish/domains/domain1/jbi/service-
assemblies/PersonCli_CA_SSLServerAuth.1/PersonCli_CA_SSLServerAuth-sun-http-
binding/sun-http-binding/orad1.ssc_29181/casaService1/casaPort1.wsdl'':
javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target
G:\GlassFishESBv21Projects\WSPolicyExploration\PersonCli_CA_SSLServerAuth\nbproj
ect\build-impl.xml:201: Deployment failure.
BUILD FAILED (total time: 2 seconds)
```

What happened? The deployer attempted to start the composite application and the SOAP Binding Component in the PersonCli_CA_SSLServerAuth on mcz02.aus.sun.com attempted to connect to the SOAP BC in the PersonSvc_CA_SSLServerAuth on orad1.ssc. Since the GalssFish instance on mcz02.aus.sun.com does not know about the certificate returned by the GlassFish instance on orad1.ssc it rejected it with a rude message.

Listing 5.8.3 shows selected messages from the mcz02.aus.sun.com's server.log, relating ot the SSL Handshake failure.

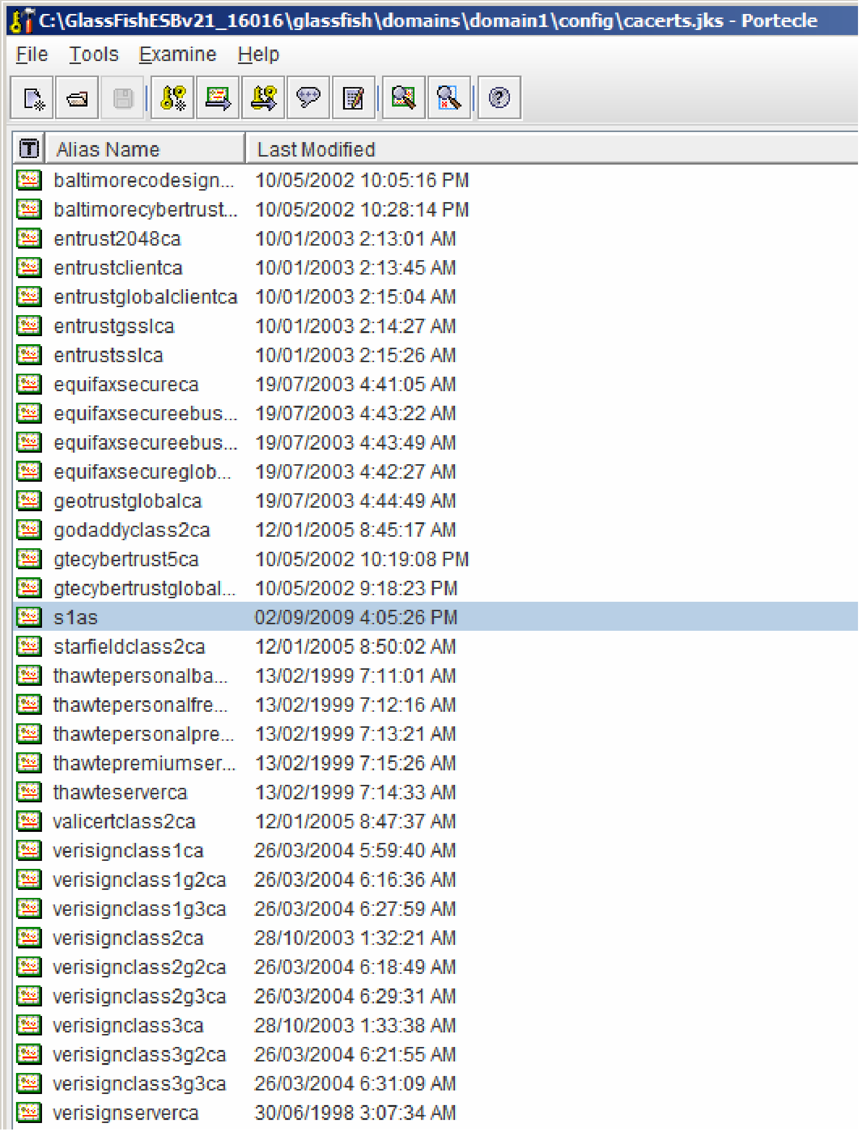
Listing 5.8.3 Client-side SSL Handshake

```
[#|2009-09-07T18:18:18.343+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
%% No cached client session|#]
[#|2009-09-07T18:18:18.343+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
*** ClientHello, TLSv1|#]
...
[#|2009-09-07T18:18:18.343+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
Session ID:  |#]
[#|2009-09-07T18:18:18.343+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|{}|#]
[#|2009-09-07T18:18:18.343+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
Cipher Suites: [SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA,
SSL_RSA_WITH_DES_CBC_SHA, SSL_DHE_RSA_WITH_DES_CBC_SHA, SSL_DHE_DSS_WITH_DES_CBC_SHA,
SSL_RSA_EXPORT_WITH_RC4_40_MD5, SSL_RSA_EXPORT_WITH_DES40_CBC_SHA,
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA, SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA]|#]
[#|2009-09-07T18:18:18.343+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
Compression Methods:  { |#]
...
[#|2009-09-07T18:18:18.343+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
PersonCli_CA_SSLServerAuth-sun-http-binding, WRITE: TLSv1 Handshake, length = 73|#]
[#|2009-09-07T18:18:18.343+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
PersonCli_CA_SSLServerAuth-sun-http-binding, WRITE: SSLv2 client hello message, length =
98|#]
[#|2009-09-07T18:18:18.640+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
PersonCli_CA_SSLServerAuth-sun-http-binding, READ: TLSv1 Handshake, length = 794|#]
[#|2009-09-07T18:18:18.640+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
*** ServerHello, TLSv1|#]
...
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
Session ID:  |#]
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|{74, 164, 193, 203, 245, 191, 123, 38, 91, 15, 20, 255, 117,
203, 207, 130, 17, 102, 76, 59, 54, 207, 0, 9, 12, 125, 143, 33, 189, 59, 111, 26}|#]
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
Cipher Suite: SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
Compression Method: 0|#]
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
***|#]
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
%% Created:  [Session-3, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
** SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
*** Certificate chain|#]
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
chain [0] = [
[
  Version: V3
  Subject: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems,
L=Santa Clara, ST=California, C=US
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus:
10212154115706506957556825027043721498435033633815799809476856914850053601803189481914549l
113311079336949399472032140208618356816163637202618727298710465300230334324506184949208474
44693317862325372611096817580008440551704374012071552944362437383385053028767344267433318l
408072531272948503339758724948550982873
  public exponent: 65537
  Validity: [From: Sat Sep 05 13:48:28 EST 2009,
               To: Tue Sep 03 13:48:28 EST 2019]
  Issuer: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  SerialNumber: [    4aa1df8c]

Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 9E 79 9C E9 59 86 34 8F   FD 75 09 F7 82 D0 82 CE  .y..Y.4..u......
0010: BE 9A 44 EE                                        ..D.
]
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 23 A7 FD 51 1F 81 9E 8C   34 3A 58 01 EF 5A 04 CD  #..Q....4:X..Z..
0010: AD 35 2C 67 17 40 3A B8   EA 19 37 DB B2 B3 C8 EA  .5,g.@:...7.....
0020: 5B 4F 0E 30 4E 9D 42 23   52 FE E8 53 44 8B 64 21  [O.0N.B#R..SD.d!
0030: CF 5F EE 07 D5 60 1E F2   1B EA 68 99 E4 BB 6C 89  ._...`....h...l.
0040: 02 21 1D A5 AE 6C 26 14   8C 92 02 92 E3 C1 74 56  .!...l&.......tV
0050: 6A 69 96 8E 2D 1E 7D 6C   52 5E 99 38 20 8B 19 C4  ji..-..lR^.8 ...
0060: 52 11 89 B3 73 D0 6C 61   B2 DB BF CA 58 0A 3A 5D  R...s.la....X.:]
0070: 40 81 97 CC 3F 60 A6 1E   B5 D6 60 8A C6 6B B6 F6  @...?`....`..k..

]|#]
...
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|, SEND TLSv1 ALERT:  |#]
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|fatal, |#]
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|description = certificate_unknown|#]
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
PersonCli_CA_SSLServerAuth-sun-http-binding, WRITE: TLSv1 Alert, length = 2|#]
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
PersonCli_CA_SSLServerAuth-sun-http-binding, called closeSocket()|#]
[#|2009-09-07T18:18:18.656+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=81;_ThreadName=PersonCli_CA_SSLS
erverAuth-sun-http-binding;|
PersonCli_CA_SSLServerAuth-sun-http-binding, handling exception:
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path
building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target|#]
```

```
[#|2009-09-07T18:18:18.671+1000|SEVERE|sun-
appserver2.1|com.sun.jbi.httpsoapbc.HttpSoapBindingDeployer|_ThreadID=81;_ThreadName=Perso
nCli_CA_SSLServerAuth-sun-http-binding;_RequestID=c5817253-ec00-4d70-9d29-
218bfc560ba3;|HTTPBC-E00201: Deployment failed. javax.wsdl.WSDLException: WSDLException
(at /definitions/import): faultCode=OTHER_ERROR: Unable to resolve imported document at
'https://orad1.ssc:29181/PersonSvc_CA_SSLServerAuth-sun-http-binding/PersonAbsSvc.wsdl',
relative to 'file:/C:/GlassFishESBv21_16016/glassfish/domains/domain1/jbi/service-
assemblies/PersonCli_CA_SSLServerAuth.1/PersonCli_CA_SSLServerAuth-sun-http-binding/sun-
http-binding/orad1.ssc_29181/casaService1/casaPort1.wsdl':
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path
building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target
javax.jbi.JBIException: javax.wsdl.WSDLException: WSDLException (at /definitions/import):
faultCode=OTHER_ERROR: Unable to resolve imported document at
'https://orad1.ssc:29181/PersonSvc_CA_SSLServerAuth-sun-http-binding/PersonAbsSvc.wsdl',
relative to 'file:/C:/GlassFishESBv21_16016/glassfish/domains/domain1/jbi/service-
assemblies/PersonCli_CA_SSLServerAuth.1/PersonCli_CA_SSLServerAuth-sun-http-binding/sun-
http-binding/orad1.ssc_29181/casaService1/casaPort1.wsdl':
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path
building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target
        at
com.sun.jbi.httpsoapbc.ServiceUnitImpl.createEndpoints(ServiceUnitImpl.java:601)
        at com.sun.jbi.httpsoapbc.ServiceUnitImpl.deploy(ServiceUnitImpl.java:201)
        at
com.sun.jbi.httpsoapbc.HttpSoapBindingDeployer.deploy(HttpSoapBindingDeployer.java:146)
        at
com.sun.jbi.framework.ServiceUnitOperation.process(ServiceUnitOperation.java:177)
        at com.sun.jbi.framework.Operation.run(Operation.java:104)
        at java.lang.Thread.run(Thread.java:619)
Caused by: javax.wsdl.WSDLException: WSDLException (at /definitions/import):
faultCode=OTHER_ERROR: Unable to resolve imported document at
'https://orad1.ssc:29181/PersonSvc_CA_SSLServerAuth-sun-http-binding/PersonAbsSvc.wsdl',
relative to 'file:/C:/GlassFishESBv21_16016/glassfish/domains/domain1/jbi/service-
assemblies/PersonCli_CA_SSLServerAuth.1/PersonCli_CA_SSLServerAuth-sun-http-binding/sun-
http-binding/orad1.ssc_29181/casaService1/casaPort1.wsdl':
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path
building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target
        at com.ibm.wsdl.xml.WSDLReaderImpl.parseImport(WSDLReaderImpl.java:561)
        at com.ibm.wsdl.xml.WSDLReaderImpl.parseDefinitions(WSDLReaderImpl.java:331)
        at com.ibm.wsdl.xml.WSDLReaderImpl.readWSDL(WSDLReaderImpl.java:2324)
        at com.ibm.wsdl.xml.WSDLReaderImpl.readWSDL(WSDLReaderImpl.java:2288)
        at com.ibm.wsdl.xml.WSDLReaderImpl.readWSDL(WSDLReaderImpl.java:2341)
        at com.ibm.wsdl.xml.WSDLReaderImpl.readWSDL(WSDLReaderImpl.java:2249)
        at com.ibm.wsdl.xml.WSDLReaderImpl.readWSDL(WSDLReaderImpl.java:2211)
        at
com.sun.jbi.wsdlvalidator.impl.ValidatingWSDLReaderImpl.readWSDL(ValidatingWSDLReaderImpl.
java:88)
        at
com.sun.jbi.wsdlvalidator.impl.ValidatingWSDLReaderImpl.readWSDL(ValidatingWSDLReaderImpl.
java:95)
        at
com.sun.jbi.wsdlvalidator.impl.ValidatingWSDLReaderImpl.readWSDL(ValidatingWSDLReaderImpl.
java:95)
        at
com.sun.jbi.wsdlvalidator.impl.ValidatingWSDLReaderImpl.readWSDL(ValidatingWSDLReaderImpl.
java:95)
        at
com.sun.jbi.httpsoapbc.ServiceUnitImpl.createEndpoints(ServiceUnitImpl.java:486)
        ... 5 more
Caused by: javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException:
PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException:
unable to find valid certification path to requested target
        at com.sun.net.ssl.internal.ssl.Alerts.getSSLException(Alerts.java:174)
        at com.sun.net.ssl.internal.ssl.SSLSocketImpl.fatal(SSLSocketImpl.java:1611)
        at com.sun.net.ssl.internal.ssl.Handshaker.fatalSE(Handshaker.java:187)
        at com.sun.net.ssl.internal.ssl.Handshaker.fatalSE(Handshaker.java:181)
        at
com.sun.net.ssl.internal.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:1035
)
        at
com.sun.net.ssl.internal.ssl.ClientHandshaker.processMessage(ClientHandshaker.java:124)
        at com.sun.net.ssl.internal.ssl.Handshaker.processLoop(Handshaker.java:516)
        at com.sun.net.ssl.internal.ssl.Handshaker.process_record(Handshaker.java:454)
        at com.sun.net.ssl.internal.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:884)
        at
com.sun.net.ssl.internal.ssl.SSLSocketImpl.performInitialHandshake(SSLSocketImpl.java:1112
)
        at
com.sun.net.ssl.internal.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:1139)
        at
com.sun.net.ssl.internal.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:1123)
        at sun.net.www.protocol.https.HttpsClient.afterConnect(HttpsClient.java:434)
        at
sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.connect(AbstractDelegateHttp
sURLConnection.java:166)
        at
sun.net.www.protocol.http.HttpURLConnection.getInputStream(HttpURLConnection.java:1049)
        at
sun.net.www.protocol.https.HttpsURLConnectionImpl.getInputStream(HttpsURLConnectionImpl.ja
va:234)
        at java.net.URL.openStream(URL.java:1010)
        at com.ibm.wsdl.util.StringUtils.getContentAsInputStream(StringUtils.java:184)
        at com.ibm.wsdl.xml.WSDLReaderImpl.parseImport(WSDLReaderImpl.java:442)
        ... 16 more
Caused by: sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid
certification path to requested target
        at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:285)
        at sun.security.validator.PKIXValidator.engineValidate(PKIXValidator.java:191)
        at sun.security.validator.Validator.validate(Validator.java:218)
        at
com.sun.net.ssl.internal.ssl.X509TrustManagerImpl.validate(X509TrustManagerImpl.java:126)
        at
com.sun.net.ssl.internal.ssl.X509TrustManagerImpl.checkServerTrusted(X509TrustManagerImpl.
java:209)
        at
com.sun.net.ssl.internal.ssl.X509TrustManagerImpl.checkServerTrusted(X509TrustManagerImpl.
java:249)
        at
com.sun.net.ssl.internal.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:1014
)
        ... 30 more
Caused by: sun.security.provider.certpath.SunCertPathBuilderException: unable to find
valid certification path to requested target
        at
sun.security.provider.certpath.SunCertPathBuilder.engineBuild(SunCertPathBuilder.java:174)
        at java.security.cert.CertPathBuilder.build(CertPathBuilder.java:238)
        at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:280)
        ... 36 more
|#]
```

The client commenced SSL Handshake by sending the Client Hello message nominating cryptographic algorithms and compression method it is willing to use, the server responded with the Server Hello message, nominating the selected cryptographic algorithm, session id and its server certificate. The client looked at the certificate, did not found the CA that issued it in its truststore and aborted the SSL Handshake.

Recall from earlier discussion that for a self-signed certificate, which the one returned by orad1.ssc is, it must be explicitly imported into the client's GlassFish instance's cecerts.jks truststore. Whe I look at the client's (mcz02) GlassFish instance's cacert.jks I don't see the certificate that corresponds to orad1.ssc. Figure 5.8.27.


C:\GlassFishESBv21_16016\glassfish\domains\domain1\config\cacerts.jks - Portecle
File
Tools
Examine
Help

T	Alias Name	Last Modified
	baltimorecodesign...	10/05/2002 10:05:16 PM
	baltimorecybertrust...	10/05/2002 10:28:14 PM
	entrust2048ca	10/01/2003 2:13:01 AM
	entrustclientca	10/01/2003 2:13:45 AM
	entrustglobalclientca	10/01/2003 2:15:04 AM
	entrustgsslca	10/01/2003 2:14:27 AM
	entrustsslca	10/01/2003 2:15:26 AM
	equifaxsecureca	19/07/2003 4:41:05 AM
	equifaxsecureebus...	19/07/2003 4:43:22 AM
	equifaxsecureebus...	19/07/2003 4:43:49 AM
	equifaxsecureglob...	19/07/2003 4:42:27 AM
	geotrustglobalca	19/07/2003 4:44:49 AM
	godaddyclass2ca	12/01/2005 8:45:17 AM
	gtecybertrust5ca	10/05/2002 10:19:08 PM
	gtecybertrustglobal...	10/05/2002 9:18:23 PM
	s1as	02/09/2009 4:05:26 PM
	starfieldclass2ca	12/01/2005 8:50:02 AM
	thawtepersonalba...	13/02/1999 7:11:01 AM
	thawtepersonalfre...	13/02/1999 7:12:16 AM
	thawtepersonalpre...	13/02/1999 7:13:21 AM
	thawtepremiumser...	13/02/1999 7:15:26 AM
	thawteserverca	13/02/1999 7:14:33 AM
	valicertclass2ca	12/01/2005 8:47:37 AM
	verisignclass1ca	26/03/2004 5:59:40 AM
	verisignclass1g2ca	26/03/2004 6:16:36 AM
	verisignclass1g3ca	26/03/2004 6:27:59 AM
	verisignclass2ca	28/10/2003 1:32:21 AM
	verisignclass2g2ca	26/03/2004 6:18:49 AM
	verisignclass2g3ca	26/03/2004 6:29:31 AM
	verisignclass3ca	28/10/2003 1:33:38 AM
	verisignclass3g2ca	26/03/2004 6:21:55 AM
	verisignclass3g3ca	26/03/2004 6:31:09 AM
	verisignserverca	30/06/1998 3:07:34 AM

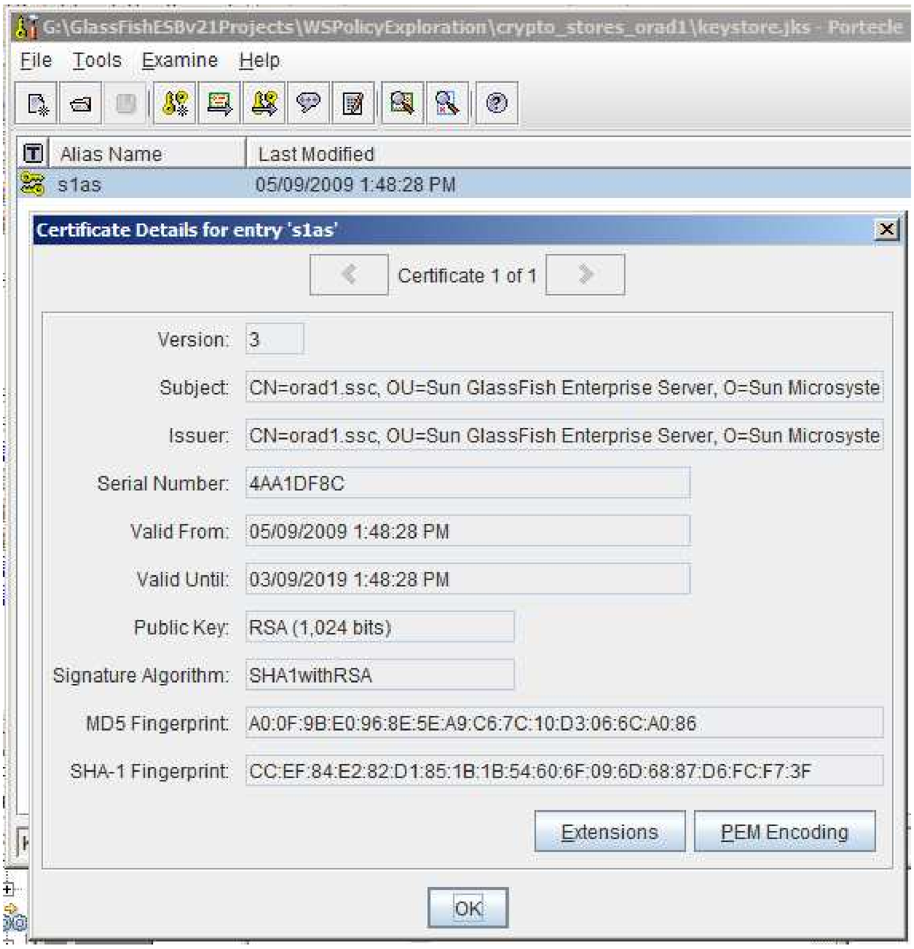
Figure 5.8.27 *Default GlassFish list of trusted certificates*

We need to extract the orad1.ssc's certificate from its keystore.jks and import it into the mcz02's cacerts.jks.

The steps to extract a certificate from a keystore using the Prtacle tool are discussed in the next few paragraphs. If you have the certificate of the remote host, as you might, or use another tool to work with keystores, skip past this section.

I transferred the truststore.jks and the cacerts.jsk from orad1.ssc to the machine on which I have Portacle installed so I can easily manipulate them.

Open the remote host's (orad1 for me) truststore.jsk with Prtacle. Password, by default, is changeit. Figure 5.8.28 shows the keystore content and the content of the s1as certificate.



G:\GlassFishESBv21Projects\WSPolicyExploration\crypto_stores_orad1\keystore.jks - Portecle
File
Tools
Examine
Help
Certificate Details for entry 's1as'
Certificate 1 of 1
Extensions
PEM Encoding
OK

Alias Name	Last Modified
s1as	05/09/2009 1:48:28 PM

Version: 3

Subject: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsyste

Issuer: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsyste

Serial Number: 4AA1DF8C

Valid From: 05/09/2009 1:48:28 PM

Valid Until: 03/09/2019 1:48:28 PM

Public Key: RSA (1,024 bits)

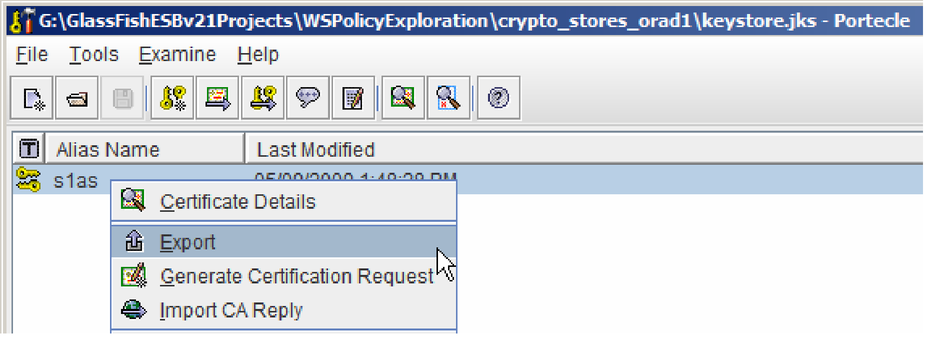
Signature Algorithm: SHA1withRSA

MD5 Fingerprint: A0:0F:9B:E0:96:8E:5E:A9:C6:7C:10:D3:06:6C:A0:86

SHA-1 Fingerprint: CC:EF:84:E2:82:D1:85:1B:1B:54:60:6F:09:6D:68:87:D6:FC:F7:3F

Figure 5.8.28 *orad1's keystore with s1as private key and certificate*

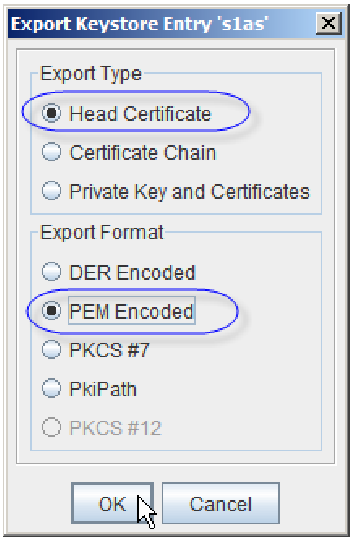
Right-click s1as key and choose "Export", as shown in Figure 5.8.29.



G:\GlassFishESBv21Projects\WSPolicyExploration\crypto_stores_orad1\keystore.jks - Portecle
File
Tools
Examine
Help
Alias Name
Last Modified
s1as
Certificate Details
Export
Generate Certification Request
Import CA Reply

Figure 5.8.29 *Trigger export of s1as certificate*

Choose to export just the “head certificate” and store it in PEM Encode form – Figure 5.8.30. Since the certificate is a self-signed certificate it does not matter whether we export the head certificate (just the certificate itself) or the Certificate Chanin (including all related CA certificates). The PEM Encoded, for Privacy Enhanced Mail (PEM) is basically a Base64 encoded binary certificate.



Export Keystore Entry 's1as'

Export Type

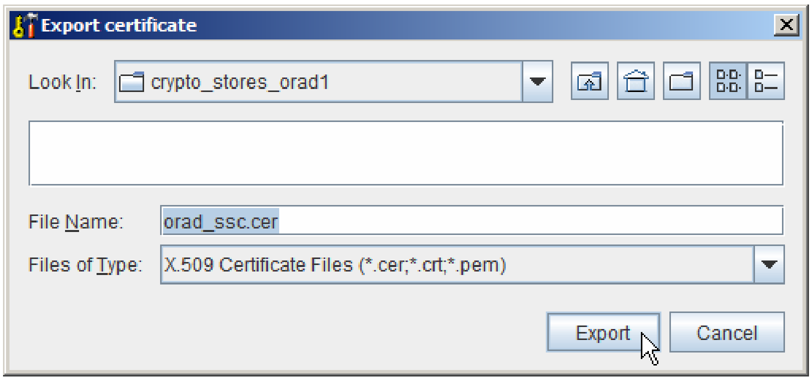
● Head Certificate
○ Certificate Chain
○ Private Key and Certificates

Export Format

○ DER Encoded
● PEM Encoded
○ PKCS #7
○ PkiPath
○ PKCS #12

Figure 5.8.30 *Choose export options*

Complete the wizard by nominating the folder to which to save the certificate. By default the name of the file will be derived from the CN (Common Name) value in the certificate. Figure 5.8.31 illustrates this for my environment.



Export certificate

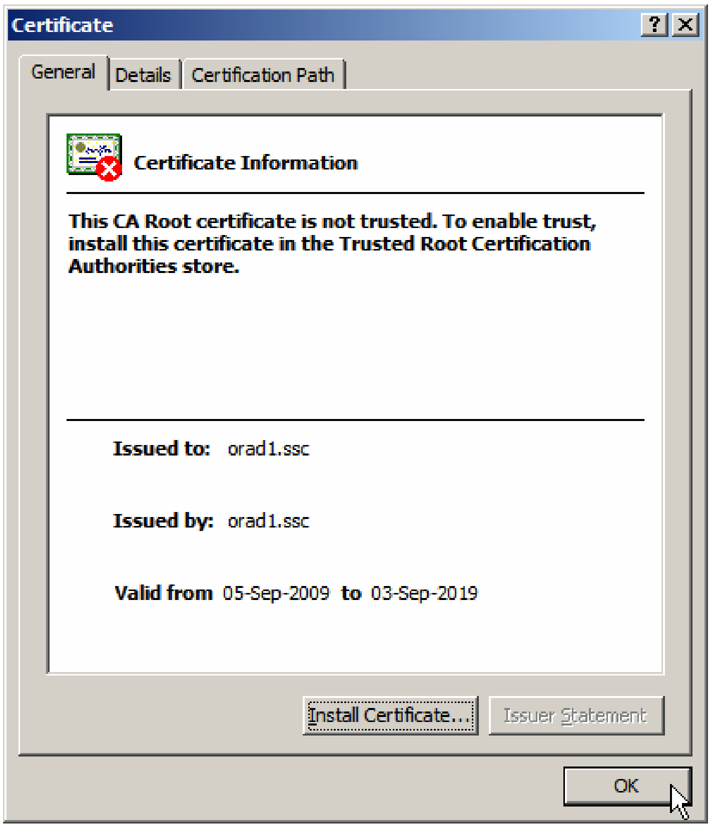
Look In: crypto_stores_orad1

File Name: orad_ssc.cer

Files of Type: X.509 Certificate Files (*.cer;*.crt;*.pem)

Figure 5.8.31 *Save certificate to a file*

If you happen to be on a Windows machine, as I am for the client-side development, you can inspect the certificate with windows tools. Merely double-click the certificate file and see what you see. What I see is shown in Figure 5.8.32.



This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

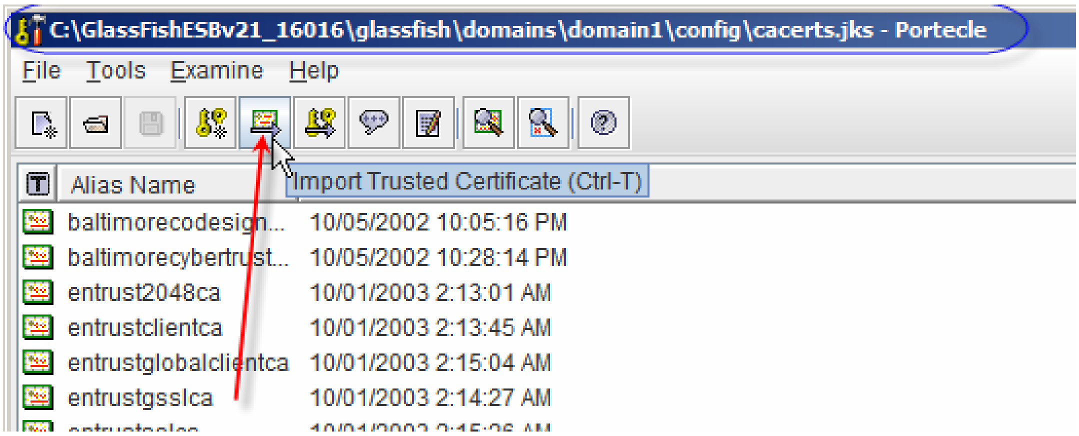
Issued to: orad1.ssc

Issued by: orad1.ssc

Valid from 05-Sep-2009 to 03-Sep-2019

Figure 5.8.32 *orad1_ssc.cer shown in Windows*

Now let's import the certificate to the local GlassFish instance's, mcz02 for me, cacerts.jsk truststore. Figure 5.8.33 illustrates the first step in this process.

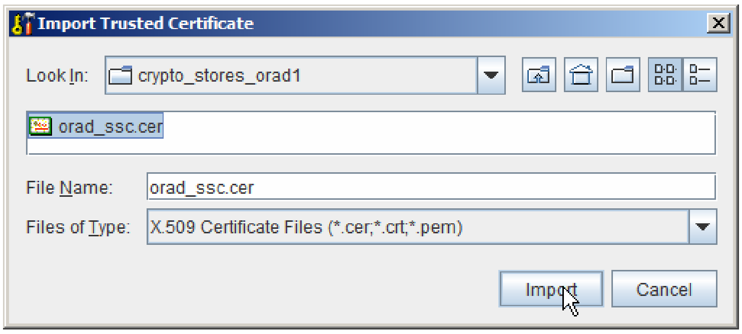


C:\GlassFishESBv21_16016\glassfish\domains\domain1\config\cacerts.jks - Portecle

Alias Name	
baltimorecodesign...	10/05/2002 10:05:16 PM
baltimorecybertrust...	10/05/2002 10:28:14 PM
entrust2048ca	10/01/2003 2:13:01 AM
entrustclientca	10/01/2003 2:13:45 AM
entrustglobalclientca	10/01/2003 2:15:04 AM
entrustgsslca	10/01/2003 2:14:27 AM

Figure 5.8.33 *Start the certificate import process*

Locate the certificate and select it for import, as is shown in Figure 5.8.34.



Import Trusted Certificate

Look In: crypto_stores_orad1

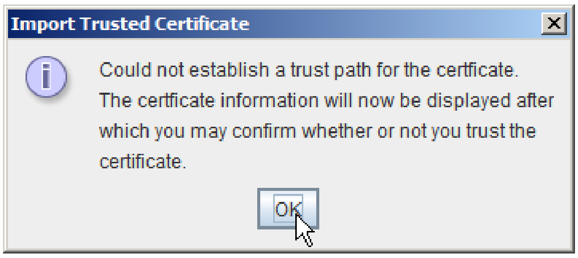
orad_ssc.cer

File Name: orad_ssc.cer

Files of Type: X.509 Certificate Files (*.cer;*.crt;*.pem)

Figure 5.8.34 *Select and import the certificate*

This is a self-signed certificate so the tool will advise that it can not establish trust for the certificate, as shown in Figure 5.8.35. Acknowledge the message.

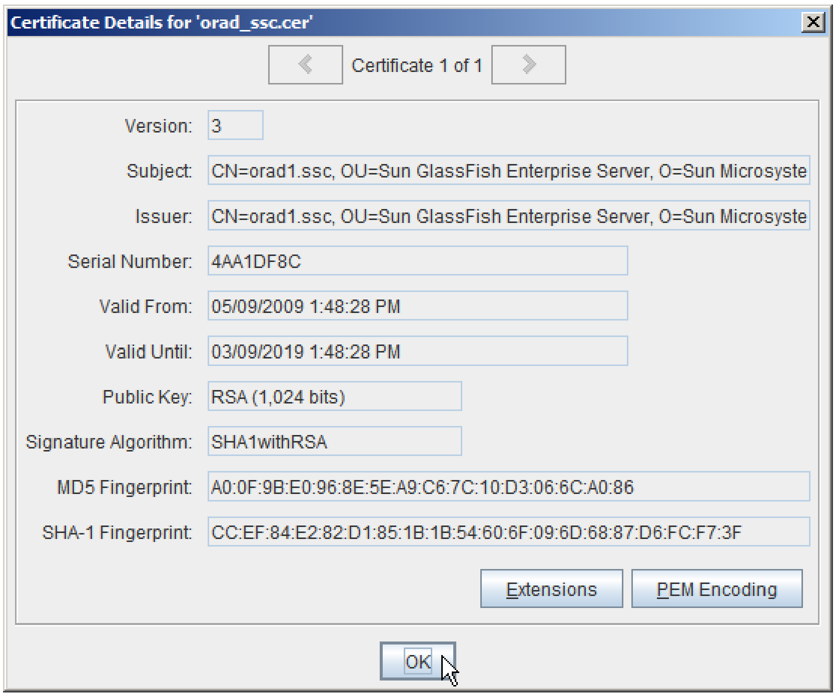


Import Trusted Certificate

Could not establish a trust path for the certficate. The certficate information will now be displayed after which you may confirm whether or not you trust the certificate.

Figure 5.8.45 *Trust path can not be established message*

Then the certificate details will be shown, as can be seen in Figure 5.8.36. Acknowledge this by clicking OK.



Certificate Details for 'orad_ssc.cer'

Certificate 1 of 1

Version: 3

Subject: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsyste

Issuer: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsyste

Serial Number: 4AA1DF8C

Valid From: 05/09/2009 1:48:28 PM

Valid Until: 03/09/2019 1:48:28 PM

Public Key: RSA (1,024 bits)

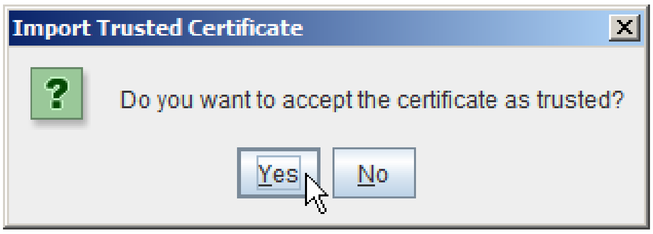
Signature Algorithm: SHA1withRSA

MD5 Fingerprint: A0:0F:9B:E0:96:8E:5E:A9:C6:7C:10:D3:06:6C:A0:86

SHA-1 Fingerprint: CC:EF:84:E2:82:D1:85:1B:1B:54:60:6F:09:6D:68:87:D6:FC:F7:3F

Figure 5.8.36 *Certificate details*

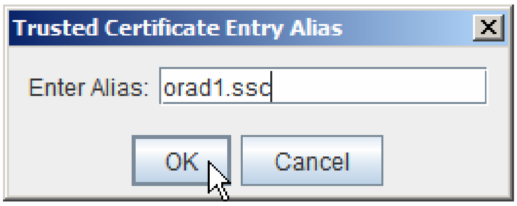
Finally, click the Yes button to accept the certificate as trusted, Figure 5.8.37, and accept the provided or modified certificate alias, Figure 5.8.38.



Import Trusted Certificate

Do you want to accept the certificate as trusted?

Figure 5.8.37 *Accept certificate as trusted*

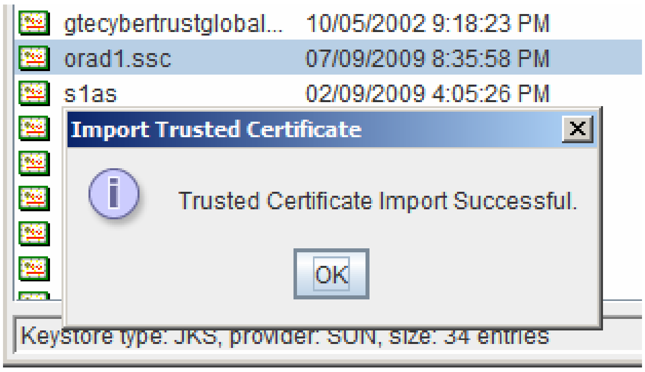


Trusted Certificate Entry Alias

Enter Alias: orad1.ssc

Figure 5.8.38 *Accept certificate alias*

Once this is done the certificate will be imported into the cacerts.jsk truststore, as trusted certificate with alias of orad1.ssc. Figure 5.8.39 shows the final feedback.



```
gtecybertrustglobal...   10/05/2002 9:18:23 PM
orad1.ssc                07/09/2009 8:35:58 PM
s1as                     02/09/2009 4:05:26 PM
```

Import Trusted Certificate

Trusted Certificate Import Successful.

Keystore type: JKS, provider: SUN, size: 34 entries

Figure 5.8.39 *Certificate was imported*

Exit from Portacle Key Manager, saving the modified cacerts.jsk keystore on the way.

The GlassFish Application Server appear to be caching the truststore content. It is necessary to re-start GlassFish after corticated is imported.

We are ready to attempt to deploy the client application again. This time, since the remote GlassFish instance's certificate is in the cacerts.jsk truststore, and is trusted, we should succeed. Listing 5.8.4 shows the feedback fro the NetBeans IDE.

Listing 5.8.4 *Deployment successful*

```
run-jbi-deploy:
[undeploy-service-assembly]
    Undeploying a service assembly...
        host=localhost
        port=24848
        name=PersonCli_CA_SSLServerAuth
[deploy-service-assembly]
    Deploying a service assembly...
        host=localhost
        port=24848

file=G:\GlassFishESBv21Projects\WSPolicyExploration\PersonCli_CA_SSLServerAuth/dist/Person
Cli_CA_SSLServerAuth.zip
[start-service-assembly]
    Starting a service assembly...
        host=localhost
        port=24848
        name=PersonCli_CA_SSLServerAuth
run:
BUILD SUCCESSFUL (total time: 8 seconds)
```

The service implementation composite application, PersonSvc_CA_SSLServerAusth, is deployed to host orad1.wa.gov.ssc. The client implementation composite application, PersonCli_CA_SSLServerAuth, is deployed to mcz02.aus.sun.com.

Let's exercise the solution using the PersonCli_WSTP web service testing project by submitting the SoapUI request, as we did before. Listing 5.8.5 shows abbreviated trace of the SSL Handshake as seen on the client side.

Listing 5.8.5 *Client-side SSL Handshake trace*

```
[#|2009-09-07T20:52:16.500+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
%% Client cached [Session-1, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-07T20:52:16.500+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
%% Try resuming [Session-1, SSL_RSA_WITH_RC4_128_MD5] from port 4707|#]
[#|2009-09-07T20:52:16.500+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
*** ClientHello, TLSv1|#]
...
[#|2009-09-07T20:52:16.515+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
Session ID:  |#]
[#|2009-09-07T20:52:16.515+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|{74, 164, 228,
147, 242, 12, 228, 182, 189, 239, 197, 106, 83, 181, 198, 176, 62, 55, 7, 142, 242, 27,
58, 223, 237, 12, 12, 62, 224, 73, 109, 208}|#]
[#|2009-09-07T20:52:16.515+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
Cipher Suites: [SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA,
SSL_RSA_WITH_DES_CBC_SHA, SSL_DHE_RSA_WITH_DES_CBC_SHA, SSL_DHE_DSS_WITH_DES_CBC_SHA,
SSL_RSA_EXPORT_WITH_RC4_40_MD5, SSL_RSA_EXPORT_WITH_DES40_CBC_SHA,
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA, SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA]|#]
[#|2009-09-07T20:52:16.515+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
Compression Methods:  { |#]
...
[#|2009-09-07T20:52:16.515+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
HTTPBC-OutboundReceiver-2, WRITE: TLSv1 Handshake, length = 105|#]
[#|2009-09-07T20:52:16.843+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
HTTPBC-OutboundReceiver-2, READ: TLSv1 Handshake, length = 74|#]
[#|2009-09-07T20:52:16.843+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
*** ServerHello, TLSv1|#]
...
[#|2009-09-07T20:52:16.890+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
Session ID:  |#]
[#|2009-09-07T20:52:16.890+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|{74, 164, 228,
147, 242, 12, 228, 182, 189, 239, 197, 106, 83, 181, 198, 176, 62, 55, 7, 142, 242, 27,
58, 223, 237, 12, 12, 62, 224, 73, 109, 208}|#]
[#|2009-09-07T20:52:16.890+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
Cipher Suite: SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-07T20:52:16.890+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
Compression Method: 0|#]
[#|2009-09-07T20:52:16.890+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
***|#]
[#|2009-09-07T20:52:16.890+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
CONNECTION KEYGEN:|#]
[#|2009-09-07T20:52:16.890+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
Client Nonce:|#]
...
Server Nonce:|#]
[#|2009-09-07T20:52:16.906+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
```

```
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
0000: |#]
...
[#|2009-09-07T20:52:16.906+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
Master Secret:|#]
[#|2009-09-07T20:52:16.906+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
0000: |#]
...
[#|2009-09-07T20:52:16.906+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
Client MAC write Secret:|#]
[#|2009-09-07T20:52:16.906+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
0000: |#]
...
[#|2009-09-07T20:52:16.906+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|Server MAC write
Secret:|#]
[#|2009-09-07T20:52:16.906+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
0000: |#]
...
[#|2009-09-07T20:52:16.906+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
Client write key:|#]
[#|2009-09-07T20:52:16.906+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
0000: |#]
...
[#|2009-09-07T20:52:16.906+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|Server write
key:|#]
[#|2009-09-07T20:52:16.906+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
0000: |#]
...
...
[#|2009-09-07T20:52:16.921+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
... no IV used for this cipher|#]
```

```
[#|2009-09-07T20:52:16.921+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
%% Server resumed [Session-1, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-07T20:52:16.921+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
HTTPBC-OutboundReceiver-2, READ: TLSv1 Change Cipher Spec, length = 1|#]
[#|2009-09-07T20:52:16.921+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
HTTPBC-OutboundReceiver-2, READ: TLSv1 Handshake, length = 32|#]
[#|2009-09-07T20:52:16.921+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
*** Finished|#]
[#|2009-09-07T20:52:16.921+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
verify_data:  { |#]
...
[#|2009-09-07T20:52:16.921+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
HTTPBC-OutboundReceiver-2, WRITE: TLSv1 Change Cipher Spec, length = 1|#]
[#|2009-09-07T20:52:16.921+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
*** Finished|#]
[#|2009-09-07T20:52:16.921+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
verify_data:  { |#]
...
[#|2009-09-07T20:52:16.921+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
HTTPBC-OutboundReceiver-2, WRITE: TLSv1 Handshake, length = 32|#]
[#|2009-09-07T20:52:16.921+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
HTTPBC-OutboundReceiver-2, WRITE: TLSv1 Application Data, length = 323|#]
[#|2009-09-07T20:52:16.921+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
HTTPBC-OutboundReceiver-2, WRITE: TLSv1 Application Data, length = 549|#]
[#|2009-09-07T20:52:17.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
HTTPBC-OutboundReceiver-2, READ: TLSv1 Application Data, length = 856|#]
[#|2009-09-07T20:52:17.187+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=58;_ThreadName=HTTPBC-
OutboundReceiver-2;Context=PersonCli_CA_SSLServerAuth-sun-http-binding-
{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
---[HTTP response 200]---|#]
...
*** a lot of stuff here - messages exchanged and so on ****
...
[#|2009-09-07T20:52:27.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=61;_ThreadName=Keep-Alive-
Timer;|
Keep-Alive-Timer, called close()|#]
[#|2009-09-07T20:52:27.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=61;_ThreadName=Keep-Alive-
Timer;|
Keep-Alive-Timer, called closeInternal(true)|#]
[#|2009-09-07T20:52:27.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=61;_ThreadName=Keep-Alive-
Timer;|
Keep-Alive-Timer|#]
[#|2009-09-07T20:52:27.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=61;_ThreadName=Keep-Alive-
Timer;|, SEND TLSv1 ALERT:  |#]
[#|2009-09-07T20:52:27.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=61;_ThreadName=Keep-Alive-
Timer;|warning, |#]
[#|2009-09-07T20:52:27.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=61;_ThreadName=Keep-Alive-
Timer;|description = close_notify|#]
[#|2009-09-07T20:52:27.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=61;_ThreadName=Keep-Alive-
Timer;|
Keep-Alive-Timer, WRITE: TLSv1 Alert, length = 18|#]
```

Note, in Listing 5.8.5, that rather than establishing a new session, the two parties resumed an earlier SSL session. The client requested session resumption and the server agreed to resume the session. No certificate was sent from the server to the client.

When the session expires a complete SSL Handshake will be performed again, including supply of the server certificate to the client. Host validation and certificate trust path validation.

The end-to-end solution using SSL with Server-side Authentication works. Let’s undeploy both composite applications in preparation for the next section – channel security using SSL with Mutual Authentication.

5.9 *Person Service - SSL Mutual Authentication*

What is described in this section ought to work.
Note that at this point in time this configuration does not work so this discussion is theoretical.

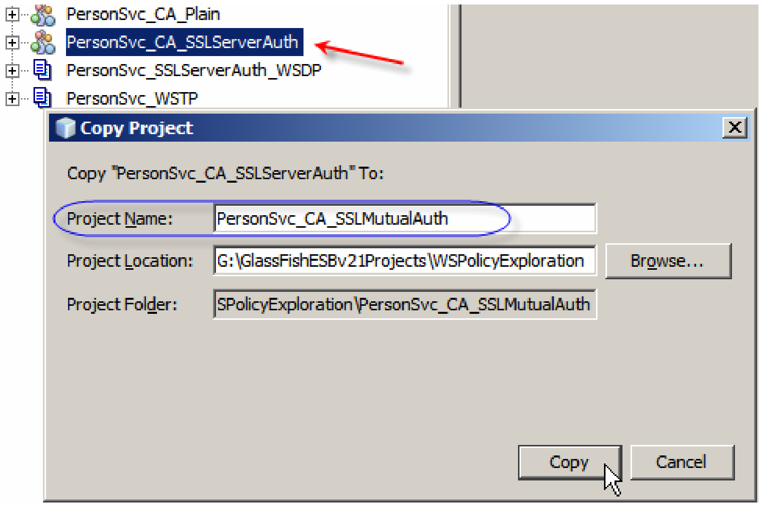
SSL with Server-side Authentication is a good choice for enforcing message privacy as it travels between two end points. It is also a good choice if the client cares about authenticity of the server but the server does not use SSL to establish authenticity of the client. This is common in electronic commerce application where the client needs to make sure it is communicating with the expected server before providing credit card and similar information to it. Channel security takes care of privacy for the credit card details and server certificate allows the client to validate the server. The commerce site uses credit card information to obtain the payment, activity which is completely unrelated to the message exchange between two endpoints. All that the commerce site cares about is that credit card information is valid and the payment can be exacted. It does not need to authenticate the client machine because it is not relevant to the transaction.

Still, there are situations where both the server and the client, in addition to maintaining message privacy through channel encryption, need to authenticate one another. SSL with Mutual Authentication can be used for this purpose. The SSL Handshake is modified in such a way that as well as server sending its certificate to the client for verification the client sends its certificate to the server for validation. Either side can abort the handshake if it is not happy with the other's certificate. Once the handshake completes successfully the channel is encrypted as normal and the same protocol operating applies (periodic change of keys, etc.).

In this section the end-to-end solution which uses secures the channel using SSL with Mutual Authentication will be developed and exercised.

As, hopefully, is clear by now, security policies, if any, can be applied to a service endpoint in a composite application without having to disturb or redevelop application logic. PersonSvc BPEL Module and PersonCli BPEL Module implement business logic. PersonSvc_CA_xxx and PersonCli_CA_xxx are used to configure varying policies.

To save ourselves the trouble of creating a composite application from scratch, not that is is a big deal, let's copy the PersonSvc_CA_SSLServerAuth composite application as PersonSvc_CA_SSLMutualAuth. Figure 5.9.1 provides an illustration of this.



Copy "PersonSvc_CA_SSLServerAuth" To:

Project Name: PersonSvc_CA_SSLMutualAuth

Project Location: G:\GlassFishESBv21Projects\WSPolicyExploration

Project Folder: SPolicyExploration\PersonSvc_CA_SSLMutualAuth

Figure 5.9.1 *Clone PersonSvc_CA_SSLServerAuth as PersonSvc_CA_SSLMutualAuth*

Open "Server Configuration" properties pane as illustrated in Figure 5.9 2.

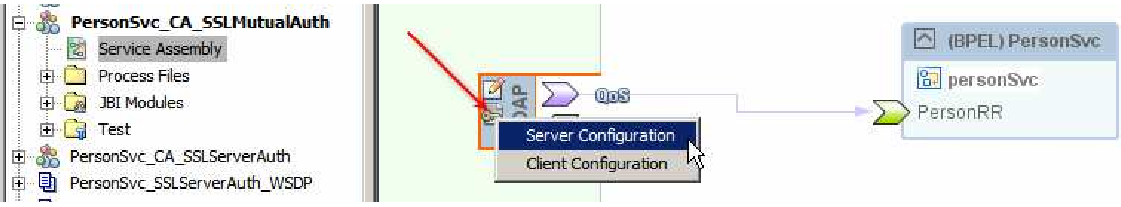
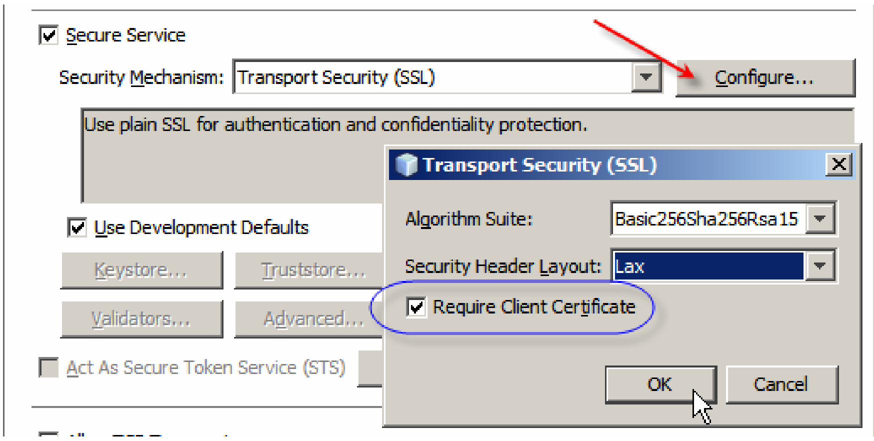


Figure 5.9.2 *Open Server Configuration Properties Pane*

Click the Configure button, check the "Require Client Certificate" checkbox and dismiss both dialogue boxes by clicking OK. Figure 5.9.3 illustrates key points.



☑ Secure Service

Security Mechanism: Transport Security (SSL)

Use plain SSL for authentication and confidentiality protection.

☑ Use Development Defaults

□ Act As Secure Token Service (STS)

Transport Security (SSL)

Algorithm Suite: Basic256Sha256Rsa15

Security Header Layout: Lax

☑ Require Client Certificate

Figure 5.9.3 *Require Client Certificate*

Build and Deploy the application to the remote GlassFish instance.

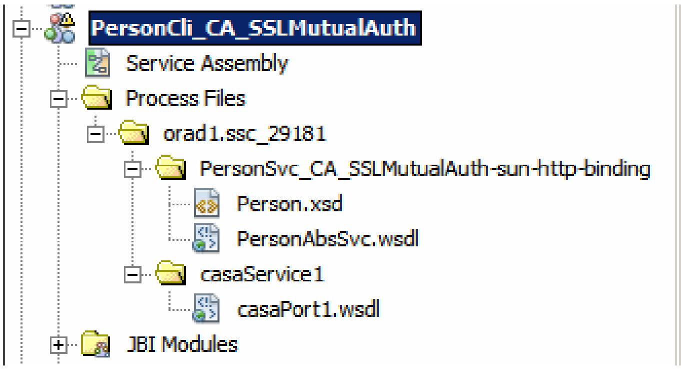
Since the service WSDL has changed we can not re-use the PersonCli_CA_SSLServerAuth and trivially modify configuraitn before re-deploying, as we have done with the PersonSvc_CA_SSLServerAuth. This is because the composite application has a copy of the server WSDL, provided when we create a "New -> External WSDL Document(s)". This copy of the WSDL is a copy of the old WSDL and there is no way to refresh it short of rdeleting and re-creating it from scratch. By the time we do that we might as well create a new PersonCli_CA_SSLMutualAuth project form scratch. This is what we will do.

Create a "New Project" -> SOA -> "Composite Application", named PersonCli_CA_SSLMutualAuth.

Determine the WSDL URL for the PersonSvc_CA_SSLMutualAuth service. For me this will be:

```
https://orad1.ssc:29181/casaService1/casaPort1?WSDL
```

Using this WSDL URL create a "New" -> "External WSDL Document(s)" in the new composite application project. Figure 5.9.4 illustrates the outcome of this process.



```
PersonCli_CA_SSLMutualAuth
  Service Assembly
  Process Files
    orad1.ssc_29181
      PersonSvc_CA_SSLMutualAuth-sun-http-binding
        Person.xsd
        PersonAbsSvc.wsdl
      casaService1
        casaPort1.wsdl
  JBI Modules
```

Figure 5.9.3 *External WSDL in the composite application project*

As we have done twice before, drag the PersonCli BPEL module onto the CASA and build the project. Notice two SOAP BCs on the canvas, connected to the BPEL Module. The Provider (TriggerCon) and the consumer (PersonSvc).

Open and inspect "Client Configuration" properties of the Consume SOAP BC, illustrated in figure 5.9.4. Notice that, as before, there are no SSL-related properties to configure.

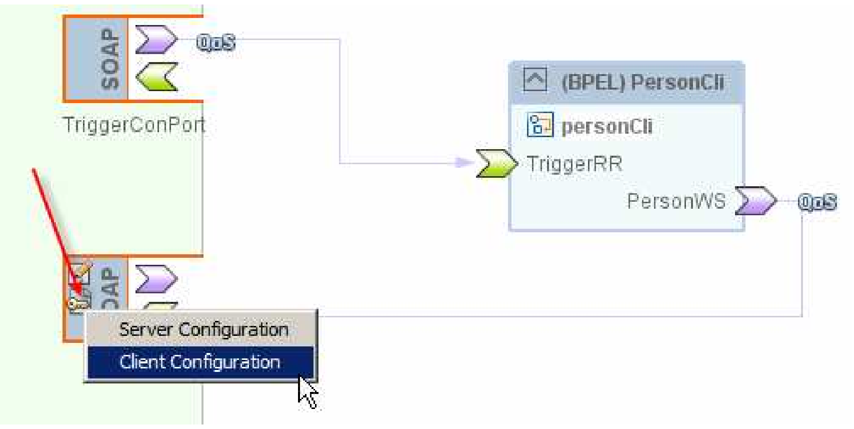


Figure 5.9.4 *Open Client Configuration of the service consumer endpoint*

We are expecting the SSL Handshake to call for client's certificate. The only certificate the client's (local) GlassFish instance know about is the one with the alias of s1as in the GlassFish instance's keystore.jsk (<glassfish_root>/domains//domain1/config/keystore.jks).

Build and deploy the project to the local GlassFish instance. Inspecting the local server.log will show that the deployment process results in a SSL Handshake and that the SSL session established when we created a new external WSDL is resumed. SSL session gets resumed if it is between the same two hosts and it is still "fresh" enough.

Use the SOAP request in PersonCli_WSTP web service testing project to trigger this solution.

5.10 EJB-based Person Svc with No Channel Security

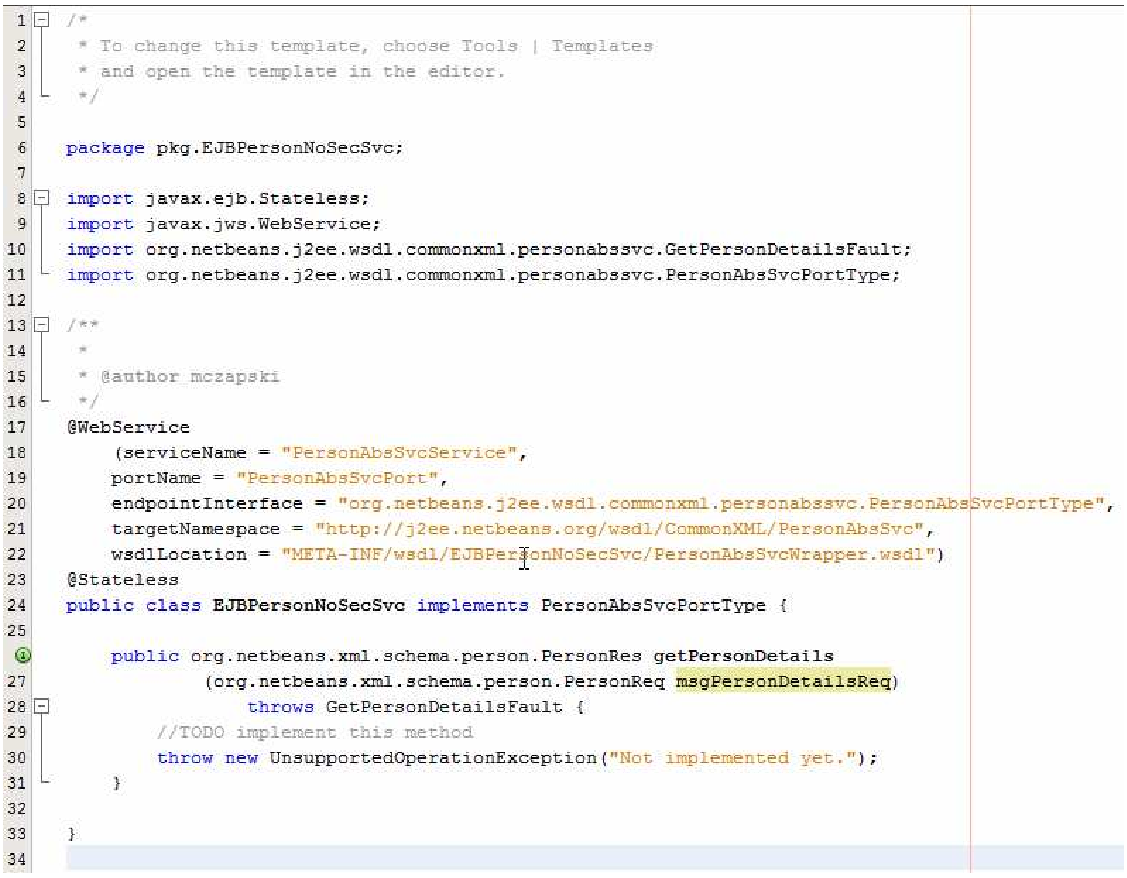
In this section the EJB-based web service provider and consumer will be built and exercised, in preparation for configuration of security policies in subsequent sections.

Listing 5.3.2 shows the Abstract WSDL that describes the interface to our service. In this section we will implement this service in Java as an EJB-based web service.

Create "New Project" -> "Java EE" -> "EJB Module", choosing the remote GlassFish instance as the deployment target. Name this module EJBPersonNoSecSvc.

In the new project create "New" -> "Web Service From WSDL", naming the service EJBPersonNoSecSvc, placing it in the package pkg.EJBPersonNoSecSvc and using the CommonXML/PersonAbsSvc "Local WSDL File".

Figure 5.10.1 shows the source of the service implementation, at this point in time, reformatted for better readability.



```
 1  /*
 2   * To change this template, choose Tools | Templates
 3   * and open the template in the editor.
 4   */
 5
 6  package pkg.EJBPersonNoSecSvc;
 7
 8  import javax.ejb.Stateless;
 9  import javax.jws.WebService;
10  import org.netbeans.j2ee.wsdl.commonxml.personabssvc.GetPersonDetailsFault;
11  import org.netbeans.j2ee.wsdl.commonxml.personabssvc.PersonAbsSvcPortType;
12
13  /**
14   *
15   * @author mczapski
16   */
17  @WebService
18      (serviceName = "PersonAbsSvcService",
19      portName = "PersonAbsSvcPort",
20      endpointInterface = "org.netbeans.j2ee.wsdl.commonxml.personabssvc.PersonAbsSvcPortType",
21      targetNamespace = "http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc",
22      wsdlLocation = "META-INF/wsdl/EJBPersonNoSecSvc/PersonAbsSvcWrapper.wsdl")
23  @Stateless
24  public class EJBPersonNoSecSvc implements PersonAbsSvcPortType {
25
        public org.netbeans.xml.schema.person.PersonRes getPersonDetails
27              (org.netbeans.xml.schema.person.PersonReq msgPersonDetailsReq)
28                  throws GetPersonDetailsFault {
29          //TODO implement this method
30          throw new UnsupportedOperationException("Not implemented yet.");
31      }
32
33  }
34
```

Figure 5.10.1 *Skeleton service implementation*

Replace lines 29-30 (in my source) with the java code shown in Listing 5.10.1.

Listing 5.10.1 Service implementation

```
        System.out.println("\n%%%%%%% PersonID: " + msgPersonDetailsReq.getPersonID());

        org.netbeans.xml.schema.person.PersonRes sRes
                = new org.netbeans.xml.schema.person.PersonRes();
        sRes.setPersonID(msgPersonDetailsReq.getPersonID());
        sRes.setFamilyName("Doe");
        sRes.setGivenName("John");
        sRes.setGender("M");

        org.netbeans.xml.schema.person.PersonRes.AddressDetails addr
                = new org.netbeans.xml.schema.person.PersonRes.AddressDetails();

        addr.setStreetAddress("33 Berry Street");
        addr.setCityTown("North Sydney");
        addr.setStateProvince("NSW");
        addr.setPostCode("2160");
        addr.setCountry("Australia");
        sRes.setAddressDetails(addr);

        org.netbeans.xml.schema.person.PersonRes.CreditCardDetails card
                = new org.netbeans.xml.schema.person.PersonRes.CreditCardDetails();
        card.setCardNumber("123-456-7689-0123");
        card.setCardType("Passport");
        card.setExpiryDate("01/21");
        card.setSecurityCode((new java.util.Date()).toString());
        sRes.setCreditCardDetails(card);

        if (msgPersonDetailsReq.getPersonID().equalsIgnoreCase("FAULT")) {
            PersonFlt pFlt = new PersonFlt();
            pFlt.setPersonID(msgPersonDetailsReq.getPersonID());
            pFlt.setFaultDetail("Induced GetPersonDetailsFault");
            GetPersonDetailsFault sFlt
                    = new GetPersonDetailsFault
                        ("Induced GetpersonDetailsFault"
                        ,pFlt);
            throw sFlt;
        }
        return sRes;
```

Right-click inside the source window and choose “Fix Imports” to resolve import-related issues.

Build and Deploy the project.

Create “New Project” -> “Java EE” -> “EJB Module”, named EJBPersonNoSecCli, to be deployed to the local GlassFish instance.

Create “New” -> “Web Service From WSDL”, named EJBPersonNoSecCli, in package pkg.EJBPersonNoSecCli, using WSDL from the CommonXML/TriggerCon.wsdl.

Create “New” -> “Web Service Client”. Click “Project”, browse to the EJBPersonNoSecSvc project and choose the web service implementation. Figure 5.10.2 illustrates this step.

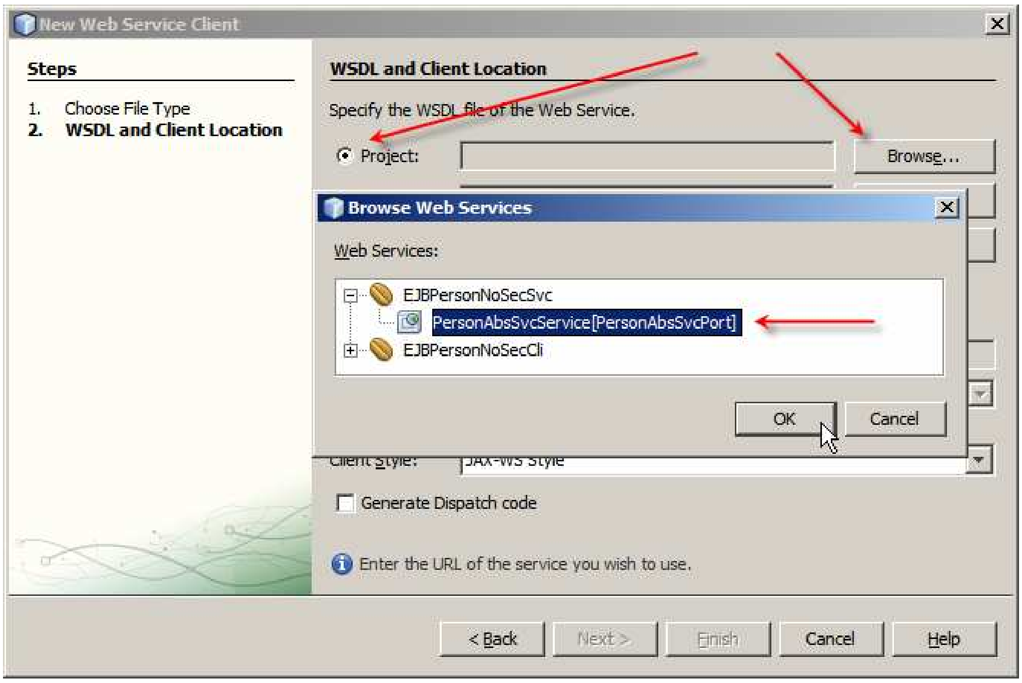


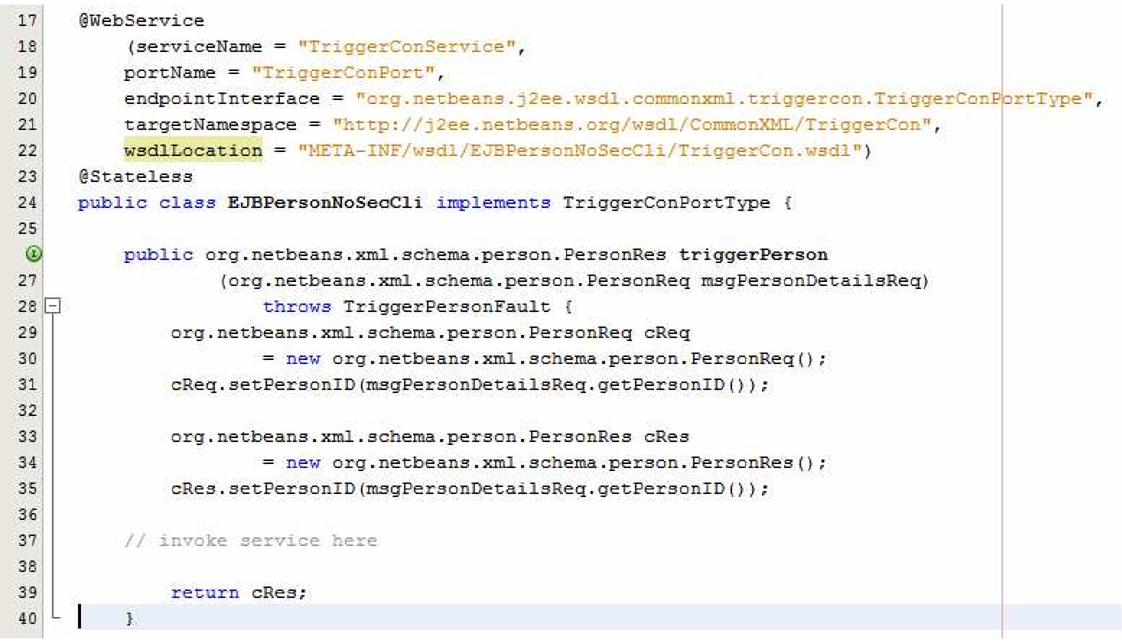
Figure 5.10.2 *Choose web service implementation for the client to invoke*

Replace the two lines of code, “//TODO …” and “throw new Unsupported…” with the code shown in Listing 5.10.2.

Listing 5.10.2 Replacement code

```
        org.netbeans.xml.schema.person.PersonReq cReq
                = new org.netbeans.xml.schema.person.PersonReq();
        cReq.setPersonID(msgPersonDetailsReq.getPersonID());

        org.netbeans.xml.schema.person.PersonRes cRes
                = new org.netbeans.xml.schema.person.PersonRes();
        cRes.setPersonID(msgPersonDetailsReq.getPersonID());

      // invoke service here

        return cRes;
```

The source, reformatted for better readability, is shown in Figure 5.10.3.

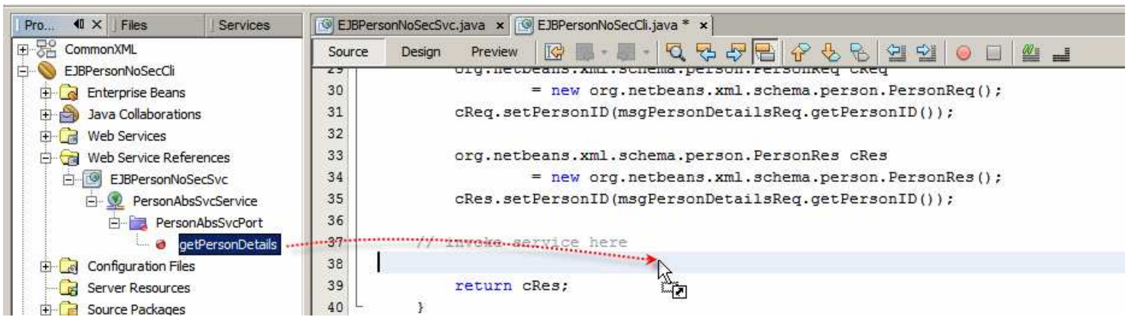


```
@WebService
    (serviceName = "TriggerConService",
    portName = "TriggerConPort",
    endpointInterface = "org.netbeans.j2ee.wsdl.commonxml.triggercon.TriggerConPortType",
    targetNamespace = "http://j2ee.netbeans.org/wsdl/CommonXML/TriggerCon",
    wsdlLocation = "META-INF/wsdl/EJBPersonNoSecCli/TriggerCon.wsdl")
@Stateless
public class EJBPersonNoSecCli implements TriggerConPortType {

    public org.netbeans.xml.schema.person.PersonRes triggerPerson
            (org.netbeans.xml.schema.person.PersonReq msgPersonDetailsReq)
                throws TriggerPersonFault {
        org.netbeans.xml.schema.person.PersonReq cReq
                = new org.netbeans.xml.schema.person.PersonReq();
        cReq.setPersonID(msgPersonDetailsReq.getPersonID());

        org.netbeans.xml.schema.person.PersonRes cRes
                = new org.netbeans.xml.schema.person.PersonRes();
        cRes.setPersonID(msgPersonDetailsReq.getPersonID());

    // invoke service here

        return cRes;
    }
```

Figure 5.10.3 *Client implementation code before service invocation is added*

Variables cReq and cRes will contain request to the service and response from the service respectively.

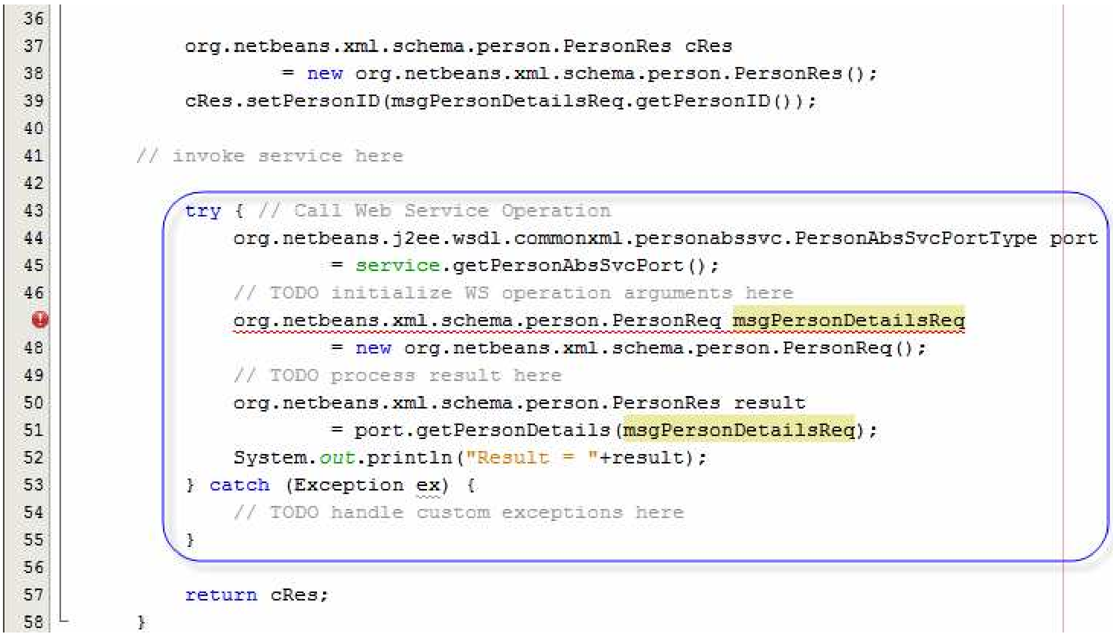
Expand Web Service References node all the way to the service operation and drag the service operation onto the source window following the comment “// invoke service here”, as shown in Figure 5.10.4.



```
        org.netbeans.xml.schema.person.PersonReq cReq
                = new org.netbeans.xml.schema.person.PersonReq();
        cReq.setPersonID(msgPersonDetailsReq.getPersonID());

        org.netbeans.xml.schema.person.PersonRes cRes
                = new org.netbeans.xml.schema.person.PersonRes();
        cRes.setPersonID(msgPersonDetailsReq.getPersonID());

    // invoke service here

        return cRes;
    }
```

Figure 5.10.4 *Drag the service operation onto the source window*

The code fragment, which was generated by the NetBeans tooling, reformatted for better readability, is shown in Figure 5.10.5.



```

        org.netbeans.xml.schema.person.PersonRes cRes
                = new org.netbeans.xml.schema.person.PersonRes();
        cRes.setPersonID(msgPersonDetailsReq.getPersonID());

    // invoke service here

        try { // Call Web Service Operation
            org.netbeans.j2ee.wsdl.commonxml.personabssvc.PersonAbsSvcPortType port
                    = service.getPersonAbsSvcPort();
            // TODO initialize WS operation arguments here
            org.netbeans.xml.schema.person.PersonReq msgPersonDetailsReq
                    = new org.netbeans.xml.schema.person.PersonReq();
            // TODO process result here
            org.netbeans.xml.schema.person.PersonRes result
                    = port.getPersonDetails(msgPersonDetailsReq);
            System.out.println("Result = "+result);
        } catch (Exception ex) {
            // TODO handle custom exceptions here
        }

        return cRes;
    }
```

Figure 5.10.5 *Service invocation skeleton code*

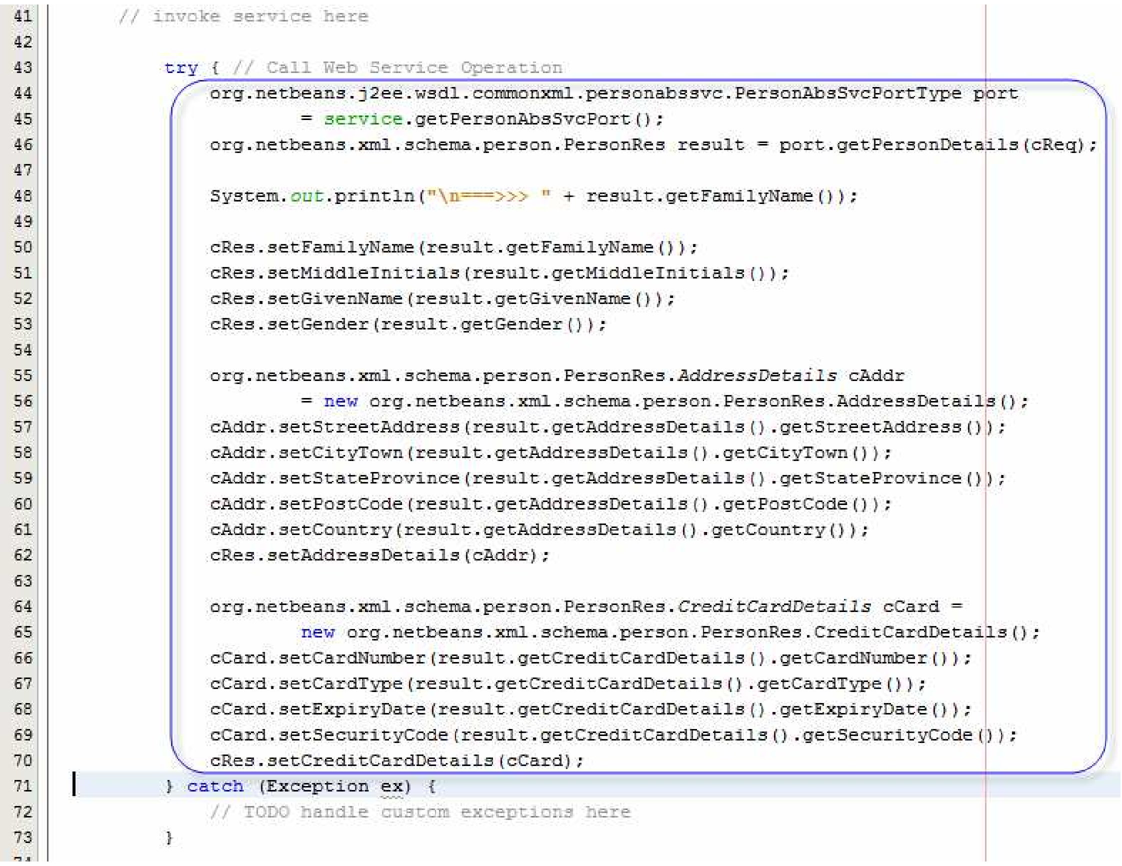
Replace the lines inside the try { … } catch block with the code in Listing 5.10.3.

Listing 5.10.3 Code to invoke the service and process the reply

```
org.netbeans.j2ee.wsdl.commonxml.personabssvc.PersonAbsSvcPortType port
        = service.getPersonAbsSvcPort();
org.netbeans.xml.schema.person.PersonRes result = port.getPersonDetails(cReq);

System.out.println("\n===>>> " + result.getFamilyName());

cRes.setFamilyName(result.getFamilyName());
cRes.setMiddleInitials(result.getMiddleInitials());
cRes.setGivenName(result.getGivenName());
cRes.setGender(result.getGender());

org.netbeans.xml.schema.person.PersonRes.AddressDetails cAddr
        = new org.netbeans.xml.schema.person.PersonRes.AddressDetails();
cAddr.setStreetAddress(result.getAddressDetails().getStreetAddress());
cAddr.setCityTown(result.getAddressDetails().getCityTown());
cAddr.setStateProvince(result.getAddressDetails().getStateProvince());
cAddr.setPostCode(result.getAddressDetails().getPostCode());
cAddr.setCountry(result.getAddressDetails().getCountry());
cRes.setAddressDetails(cAddr);

org.netbeans.xml.schema.person.PersonRes.CreditCardDetails cCard =
        new org.netbeans.xml.schema.person.PersonRes.CreditCardDetails();
cCard.setCardNumber(result.getCreditCardDetails().getCardNumber());
cCard.setCardType(result.getCreditCardDetails().getCardType());
cCard.setExpiryDate(result.getCreditCardDetails().getExpiryDate());
cCard.setSecurityCode(result.getCreditCardDetails().getSecurityCode());
cRes.setCreditCardDetails(cCard);
```

I will leave analysis of what this slab of code does to the reader. Suffices it to say that the new request, cReq, populated with the PersonID from the client, is submitted to the service and the response, cRes, is sued to populate the response of the client.

The resulting source is illustrated in Figure 5.10.6.



```
        // invoke service here

            try { // Call Web Service Operation
                org.netbeans.j2ee.wsdl.commonxml.personabssvc.PersonAbsSvcPortType port
                        = service.getPersonAbsSvcPort();
                org.netbeans.xml.schema.person.PersonRes result = port.getPersonDetails(cReq);

                System.out.println("\n===>>> " + result.getFamilyName());

                cRes.setFamilyName(result.getFamilyName());
                cRes.setMiddleInitials(result.getMiddleInitials());
                cRes.setGivenName(result.getGivenName());
                cRes.setGender(result.getGender());

                org.netbeans.xml.schema.person.PersonRes.AddressDetails cAddr
                        = new org.netbeans.xml.schema.person.PersonRes.AddressDetails();
                cAddr.setStreetAddress(result.getAddressDetails().getStreetAddress());
                cAddr.setCityTown(result.getAddressDetails().getCityTown());
                cAddr.setStateProvince(result.getAddressDetails().getStateProvince());
                cAddr.setPostCode(result.getAddressDetails().getPostCode());
                cAddr.setCountry(result.getAddressDetails().getCountry());
                cRes.setAddressDetails(cAddr);

                org.netbeans.xml.schema.person.PersonRes.CreditCardDetails cCard =
                        new org.netbeans.xml.schema.person.PersonRes.CreditCardDetails();
                cCard.setCardNumber(result.getCreditCardDetails().getCardNumber());
                cCard.setCardType(result.getCreditCardDetails().getCardType());
                cCard.setExpiryDate(result.getCreditCardDetails().getExpiryDate());
                cCard.setSecurityCode(result.getCreditCardDetails().getSecurityCode());
                cRes.setCreditCardDetails(cCard);
            } catch (Exception ex) {
                // TODO handle custom exceptions here
            }
```

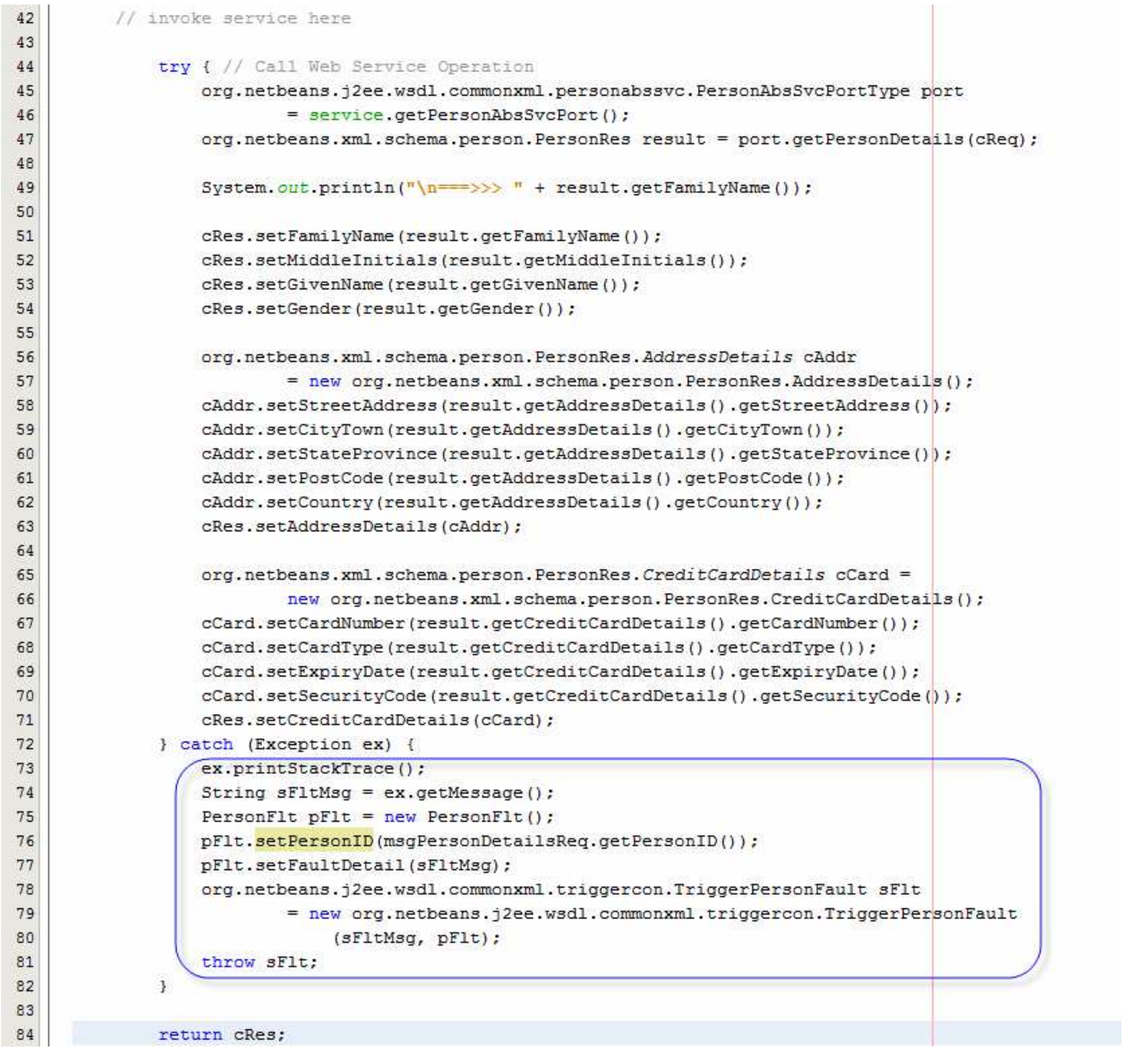
Figure 5.10.6 *Submit request and process response*

To complete client implementation let’s replace the one line comment “// TODO handle …” with the slab of code shown in Listing 5.10.4, then right-click inside the source window and choose “Fix Imports”.

Listing 5.10.4 Convert exception into a Fault

```
ex.printStackTrace();
String sFltMsg = ex.getMessage();
PersonFlt pFlt = new PersonFlt();
pFlt.setPersonID(msgPersonDetailsReq.getPersonID());
pFlt.setFaultDetail(sFltMsg);
org.netbeans.j2ee.wsdl.commonxml.triggercon.TriggerPersonFault sFlt
        = new org.netbeans.j2ee.wsdl.commonxml.triggercon.TriggerPersonFault
            (sFltMsg, pFlt);
throw sFlt;
```

Figure 5.10.7 shows the completed implementation code.



```
    // invoke service here

        try { // Call Web Service Operation
            org.netbeans.j2ee.wsdl.commonxml.personabssvc.PersonAbsSvcPortType port
                    = service.getPersonAbsSvcPort();
            org.netbeans.xml.schema.person.PersonRes result = port.getPersonDetails(cReq);

            System.out.println("\n===>>> " + result.getFamilyName());

            cRes.setFamilyName(result.getFamilyName());
            cRes.setMiddleInitials(result.getMiddleInitials());
            cRes.setGivenName(result.getGivenName());
            cRes.setGender(result.getGender());

            org.netbeans.xml.schema.person.PersonRes.AddressDetails cAddr
                    = new org.netbeans.xml.schema.person.PersonRes.AddressDetails();
            cAddr.setStreetAddress(result.getAddressDetails().getStreetAddress());
            cAddr.setCityTown(result.getAddressDetails().getCityTown());
            cAddr.setStateProvince(result.getAddressDetails().getStateProvince());
            cAddr.setPostCode(result.getAddressDetails().getPostCode());
            cAddr.setCountry(result.getAddressDetails().getCountry());
            cRes.setAddressDetails(cAddr);

            org.netbeans.xml.schema.person.PersonRes.CreditCardDetails cCard =
                    new org.netbeans.xml.schema.person.PersonRes.CreditCardDetails();
            cCard.setCardNumber(result.getCreditCardDetails().getCardNumber());
            cCard.setCardType(result.getCreditCardDetails().getCardType());
            cCard.setExpiryDate(result.getCreditCardDetails().getExpiryDate());
            cCard.setSecurityCode(result.getCreditCardDetails().getSecurityCode());
            cRes.setCreditCardDetails(cCard);
        } catch (Exception ex) {
            ex.printStackTrace();
            String sFltMsg = ex.getMessage();
            PersonFlt pFlt = new PersonFlt();
            pFlt.setPersonID(msgPersonDetailsReq.getPersonID());
            pFlt.setFaultDetail(sFltMsg);
            org.netbeans.j2ee.wsdl.commonxml.triggercon.TriggerPersonFault sFlt
                    = new org.netbeans.j2ee.wsdl.commonxml.triggercon.TriggerPersonFault
                        (sFltMsg, pFlt);
            throw sFlt;
        }

        return cRes;
```

Figure 5.10.7 *Client implementation*

Build and Deploy the project.

The web service client we just built is itself a web service. We designed this client in this way so that we can use the SoapUI plugin to invoke the client, rather then go to the trouble of working out how else to invoke it and to build the appropriate implementation.

To create a SopUI project we need a WSDL URL. Expand the Web Service node in the client project, right-click the web service name and choose Test Web Service, as illustrated in Figure 5.10.8.

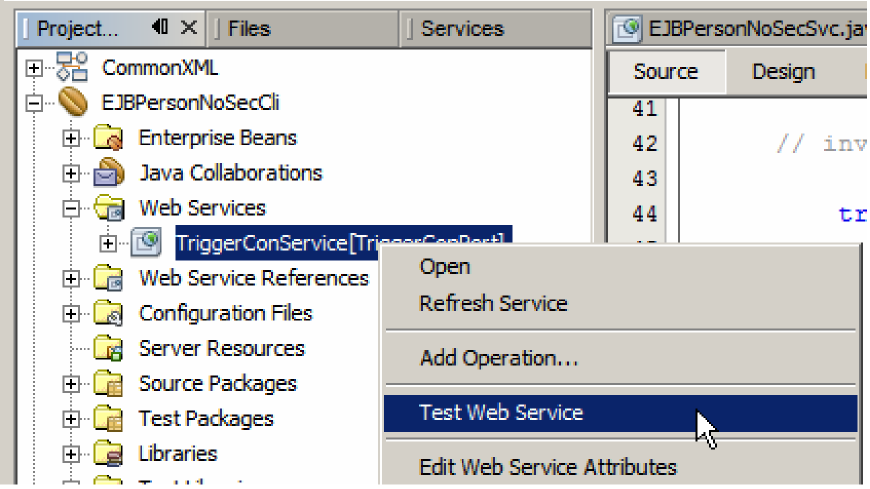
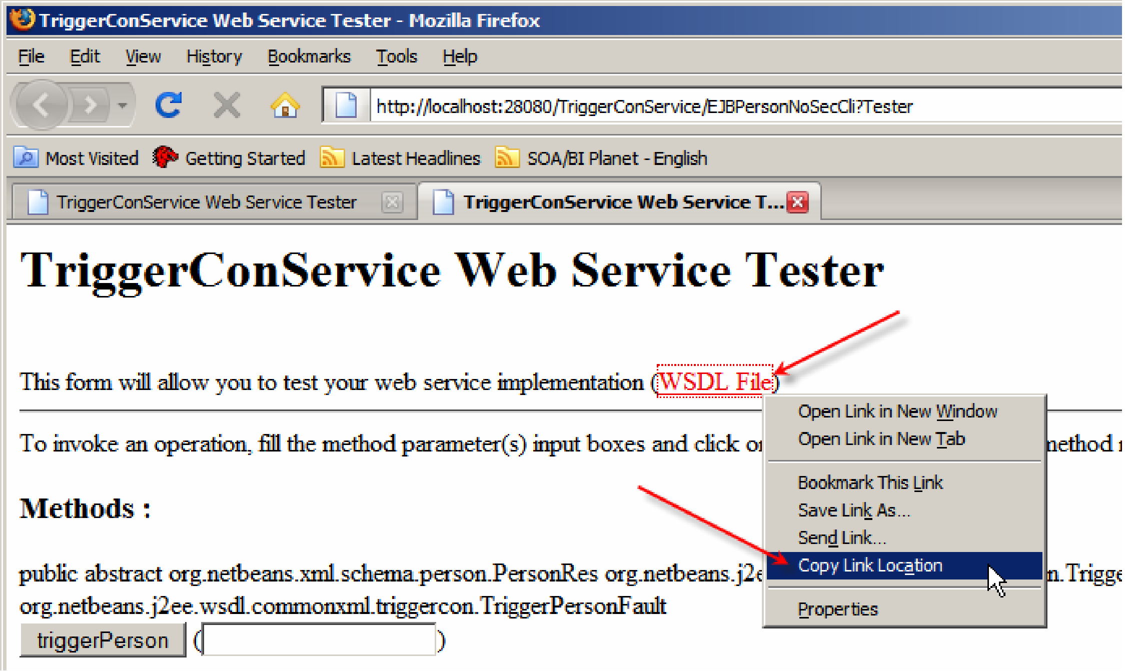


Figure 5.10.8 *Choose to test Web Service*

Once the web browser window opens with the test page, right-click the WSDL link and choose Copy Link Location, or equivalent. Figure 5.10.9 shows this in the Firefox web browser.



TriggerConService Web Service Tester

This form will allow you to test your web service implementation (WSDL File)

To invoke an operation, fill the method parameter(s) input boxes and click o

Methods :

public abstract org.netbeans.xml.schema.person.PersonRes org.netbeans.j2e
org.netbeans.j2ee.wsdl.commonxml.triggercon.TriggerPersonFault

Figure 5.10.9 *Copy WSDL link location*

For me the URL will be:

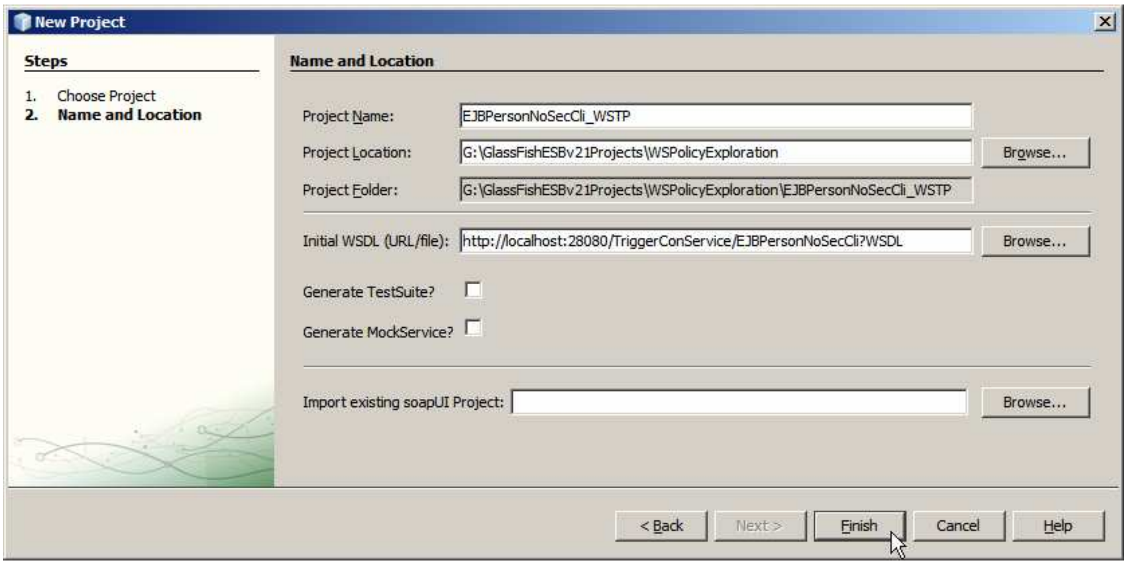
```
http://localhost:28080/TriggerConService/EJBPersonNoSecCli?WSDL
```

Default port number is 8080. I changed mine to 28080.

We can not actually test this service through the web browser. The service expects a structured message, conforming to Person XML Schema. We would have to construct a XML instance document and paste it into the text box. Even if we did go to that trouble the XML text would have gotten “escaped” by the form processor and the resulting message would have been garbled anyway.

As mentioned, we will use the SoapUI to trigger the client.

Create “New Project” -> “Java EE” -> “Web Service Testing Project”, named EJBPersonNoSecCli_WSTP (this assumes that the SoapUI plugin has been installed – if it has not then now is the time to obtain and install it). Use the WSDL URL just copied to the clipboard as the service URL. Figure 5.10.10 illustrates this step.



New Project

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: EJBPersonNoSecCli_WSTP

Project Location: G:\GlassFishESBv21Projects\WSPolicyExploration

Project Folder: G:\GlassFishESBv21Projects\WSPolicyExploration\EJBPersonNoSecCli_WSTP

Initial WSDL (URL/file): http://localhost:28080/TriggerConService/EJBPersonNoSecCli?WSDL

Generate TestSuite? □

Generate MockService? □

Import existing soapUI Project:

Figure 5.10.10 *Create new web service testing project*

Once the project is created expand the nodes, right-click the service operation and choose “New request”, as illustrated in Figure 5.10.11.

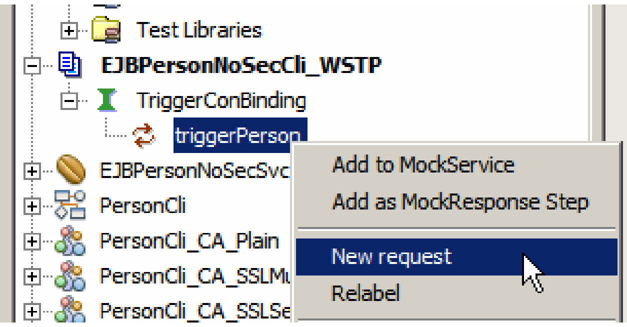
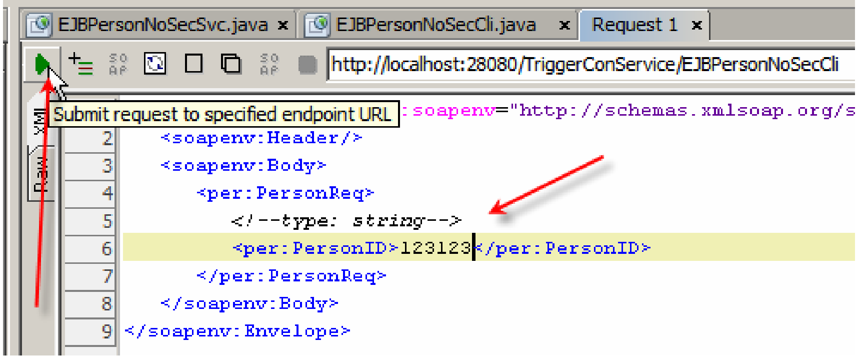


Figure 5.10.11 *Add new request*

Set PersonID to some favourite value and click the Submitt request “button”. This is illustrated in Figure 5.10.12.



EJBPersonNoSecSvc.java × | EJBPersonNoSecCli.java × | Request 1 ×

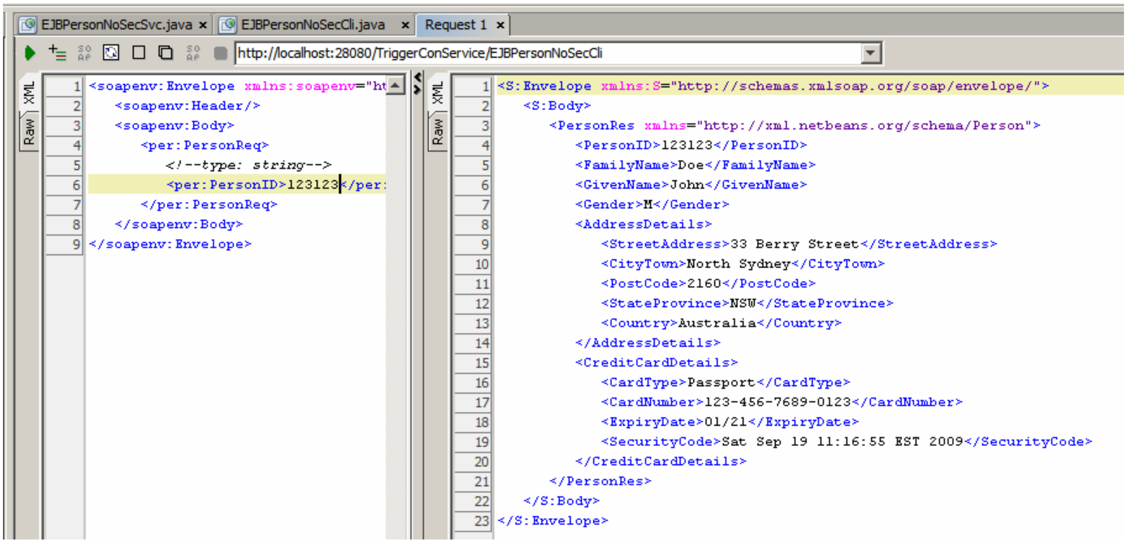
http://localhost:28080/TriggerConService/EJBPersonNoSecCli

Submit request to specified endpoint URL

```
: soapenv="http://schemas.xmlsoap.org/s
    <soapenv:Header/>
    <soapenv:Body>
       <per:PersonReq>
          <!--type: string-->
          <per:PersonID>123123</per:PersonID>
       </per:PersonReq>
    </soapenv:Body>
</soapenv:Envelope>
```

Figure 5.10.12 *Submit SOAP Request*

If all goes well, the service response will look similar to that shown in Figure 5.10.13.



EJBPersonNoSecSvc.java × | EJBPersonNoSecCli.java × | Request 1 ×

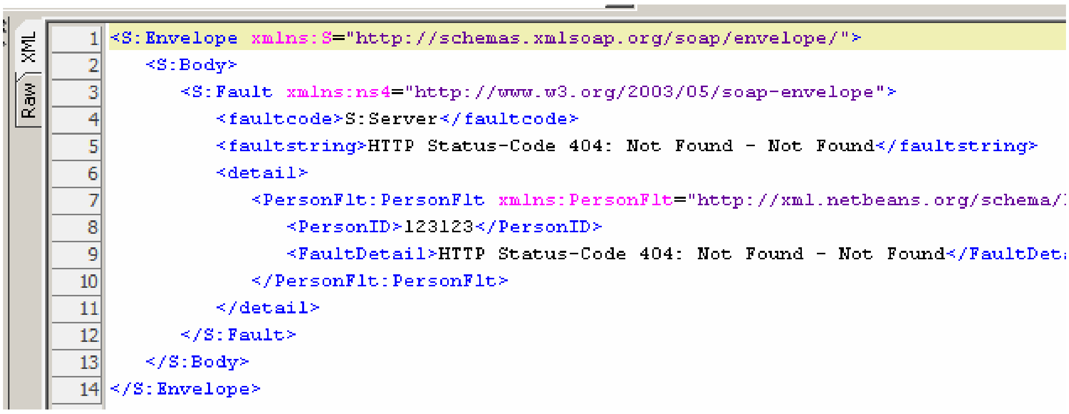
http://localhost:28080/TriggerConService/EJBPersonNoSecCli

```
<soapenv:Envelope xmlns:soapenv="ht
   <soapenv:Header/>
   <soapenv:Body>
      <per:PersonReq>
         <!--type: string-->
         <per:PersonID>123123</per:
      </per:PersonReq>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <PersonRes xmlns="http://xml.netbeans.org/schema/Person">
         <PersonID>123123</PersonID>
         <FamilyName>Doe</FamilyName>
         <GivenName>John</GivenName>
         <Gender>M</Gender>
         <AddressDetails>
            <StreetAddress>33 Berry Street</StreetAddress>
            <CityTown>North Sydney</CityTown>
            <PostCode>2160</PostCode>
            <StateProvince>NSW</StateProvince>
            <Country>Australia</Country>
         </AddressDetails>
         <CreditCardDetails>
            <CardType>Passport</CardType>
            <CardNumber>123-456-7689-0123</CardNumber>
            <ExpiryDate>01/21</ExpiryDate>
            <SecurityCode>Sat Sep 19 11:16:55 EST 2009</SecurityCode>
         </CreditCardDetails>
      </PersonRes>
   </S:Body>
</S:Envelope>
```

Figure 5.10.13 *Service Response*

To test exception processing in the client, undeploy the service and submit the request again. Figure 5.10.14 shows the Fault response.



```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <S:Fault xmlns:ns4="http://www.w3.org/2003/05/soap-envelope">
         <faultcode>S:Server</faultcode>
         <faultstring>HTTP Status-Code 404: Not Found - Not Found</faultstring>
         <detail>
            <PersonFlt:PersonFlt xmlns:PersonFlt="http://xml.netbeans.org/schema/
               <PersonID>123123</PersonID>
               <FaultDetail>HTTP Status-Code 404: Not Found - Not Found</FaultDet
            </PersonFlt:PersonFlt>
         </detail>
      </S:Fault>
   </S:Body>
</S:Envelope>
```

Figure 5.10.14 *Fault response*

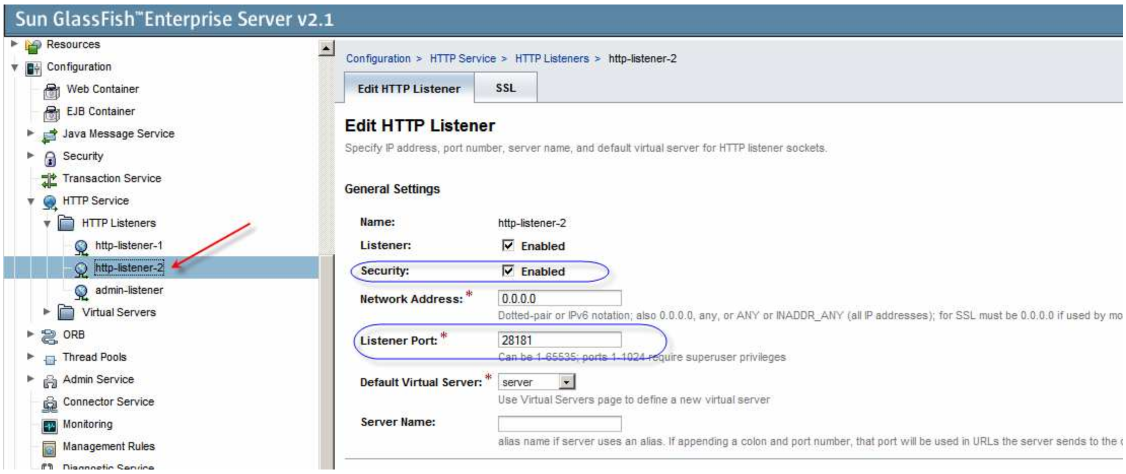
The service provider and the service consumer were implemented and exercised. This will help with the next section. Before going on let’s undeploy the client as well so that neuither the client nor the service are deployed.

5.11 EJB-based Person Svc with Server-side Authentication

In this section I will discuss how to configure SSL with Server-side Authentication for an EJB-based Web Service, using the “Interface First” programming model, though this equally applies to “Implementation First” programming model for EJB Web Services.

WS-Security Policy and Metro / WSIT are not used for this. A far as I can tell the EJB-based web service implementation, and the infrastructure that supports it, completely ignores the WS-Security Policy and Metro / WSIT policies relating to channel security – SSL / TLS, whether with Server-side or Mutual Authentication. This section is not, then, an exploration of security policy but an exploration of a practical implementation of SSL- / TLS-based channel security.

The security parameters, port, certificate alias, cryptographic suites, are configured in the GlassFish Application Server. Figures 5.11.1 and 5.11.2 show the areas of particular interest.



Edit HTTP Listener

Specify IP address, port number, server name, and default virtual server for HTTP listener sockets.

General Settings

Name: http-listener-2

Listener: ☑ Enabled

Security: ☑ Enabled

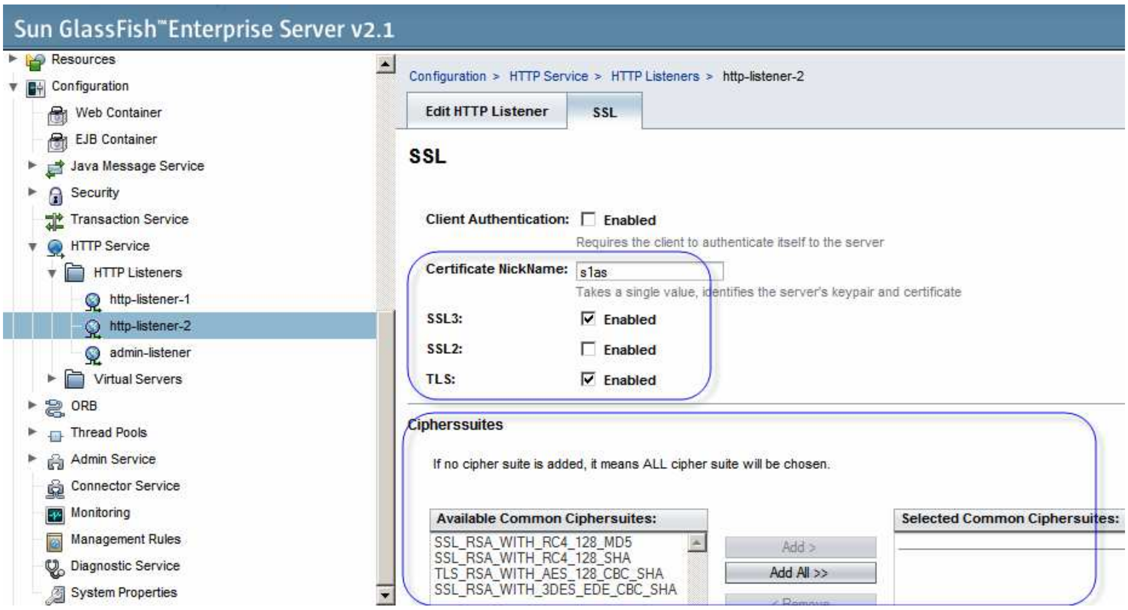
Network Address: * 0.0.0.0
Dotted-pair or IPv6 notation; also 0.0.0.0, any, or ANY or INADDR_ANY (all IP addresses); for SSL must be 0.0.0.0 if used by mo

Listener Port: * 28181
Can be 1-65535; ports 1-1024 require superuser privileges

Default Virtual Server: * server
Use Virtual Servers page to define a new virtual server

Server Name:
alias name if server uses an alias. If appending a colon and port number, that port will be used in URLs the server sends to the c

Figure 5.11.1 *http-listener-2 port configuration*



SSL

Client Authentication: □ Enabled
Requires the client to authenticate itself to the server

Certificate NickName: s1as
Takes a single value, identifies the server's keypair and certificate

SSL3: ☑ Enabled

SSL2: □ Enabled

TLS: ☑ Enabled

Cipherssuites

If no cipher suite is added, it means ALL cipher suite will be chosen.

Available Common Ciphersuites:

SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA

Selected Common Ciphersuites:

Figure 5.11.2 *SSL / TLS Configuration*

Unlike in the JBI world, where logic and policy can be separated into a BPLE Module and a Composite Application projects, with EJB-based Web Services a single project will implement both.

5.11.1 EJB-based Web Service Provider with Server-side Authentication

Click anywhere in the empty area of the Project Explorer and create "New Project" -> "Java EE" -> "EJB Module". Name this module EJBPersonSSLServerAuthSvc and have it deploy to the remote GlassFish instance.

Right-click on the name of the new project. Choose "New" -> "Other" -> "Web Services" -> "Web Service From WSDL". Name the service EJBPersonSSLServerAuthSvc, name the package pkg.EJBPersonSSLServerAuthSvc, browse to the CommonXML/PersonAbsSvc WSDL, select it and click Finish.

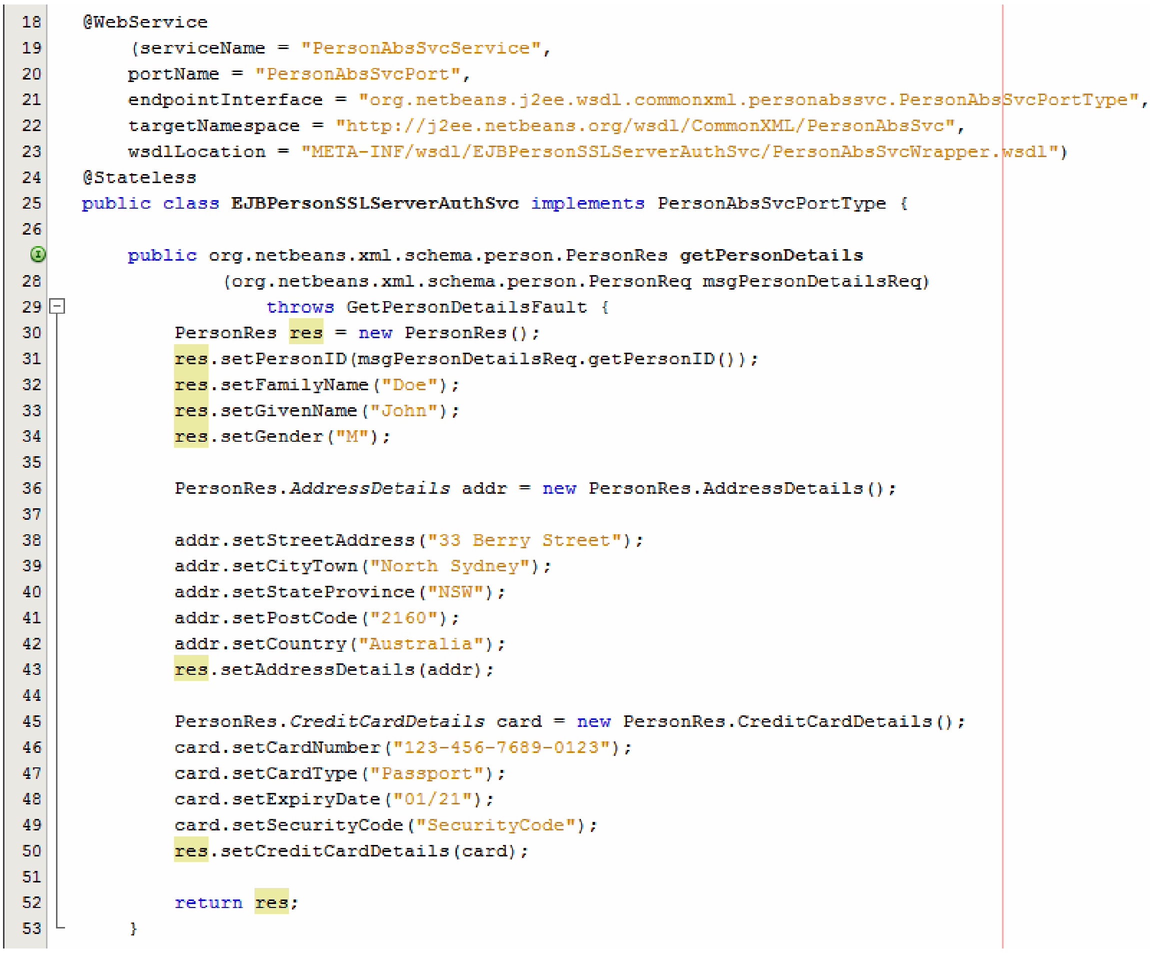
As the wizard completes a skeleton Java Source of the implementation class will appear in a window.

Replace the line containing the comment "// TO DO …" and the following line with the java statements shown in Listing 5.11.1.

Listing 5.10.1 *Method body*

```
PersonRes res = new PersonRes();
res.setPersonID(msgPersonDetailsReq.getPersonID());
res.setFamilyName("Doe");
res.setGivenName("John");
res.setGender("M");

PersonRes.AddressDetails addr = new PersonRes.AddressDetails();

addr.setStreetAddress("33 Berry Street");
addr.setCityTown("North Sydney");
addr.setStateProvince("NSW");
addr.setPostCode("2160");
addr.setCountry("Australia");
res.setAddressDetails(addr);

PersonRes.CreditCardDetails card = new PersonRes.CreditCardDetails();
card.setCardNumber("123-456-7689-0123");
card.setCardType("Passport");
card.setExpiryDate("01/21");
card.setSecurityCode("SecurityCode");
res.setCreditCardDetails(card);

return res;
```

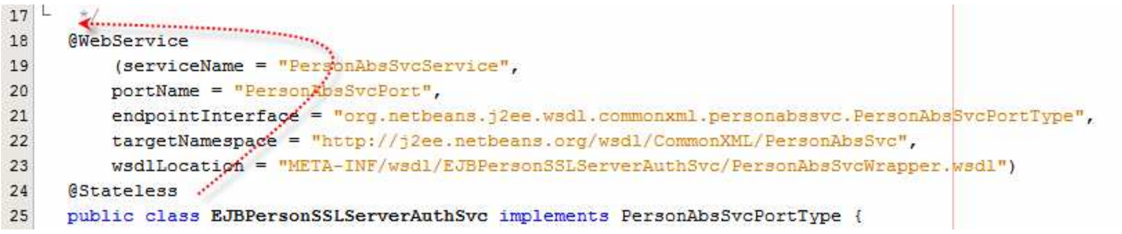
The class implementation should look like that shown in Figure 5.11.3.



```
18  @WebService
19      (serviceName = "PersonAbsSvcService",
20      portName = "PersonAbsSvcPort",
21      endpointInterface = "org.netbeans.j2ee.wsdl.commonxml.personabssvc.PersonAbsSvcPortType",
22      targetNamespace = "http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc",
23      wsdlLocation = "META-INF/wsdl/EJBPersonSSLServerAuthSvc/PersonAbsSvcWrapper.wsdl")
24  @Stateless
25  public class EJBPersonSSLServerAuthSvc implements PersonAbsSvcPortType {
26
        public org.netbeans.xml.schema.person.PersonRes getPersonDetails
28              (org.netbeans.xml.schema.person.PersonReq msgPersonDetailsReq)
29                  throws GetPersonDetailsFault {
30          PersonRes res = new PersonRes();
31          res.setPersonID(msgPersonDetailsReq.getPersonID());
32          res.setFamilyName("Doe");
33          res.setGivenName("John");
34          res.setGender("M");
35
36          PersonRes.AddressDetails addr = new PersonRes.AddressDetails();
37
38          addr.setStreetAddress("33 Berry Street");
39          addr.setCityTown("North Sydney");
40          addr.setStateProvince("NSW");
41          addr.setPostCode("2160");
42          addr.setCountry("Australia");
43          res.setAddressDetails(addr);
44
45          PersonRes.CreditCardDetails card = new PersonRes.CreditCardDetails();
46          card.setCardNumber("123-456-7689-0123");
47          card.setCardType("Passport");
48          card.setExpiryDate("01/21");
49          card.setSecurityCode("SecurityCode");
50          res.setCreditCardDetails(card);
51
52          return res;
53      }
```

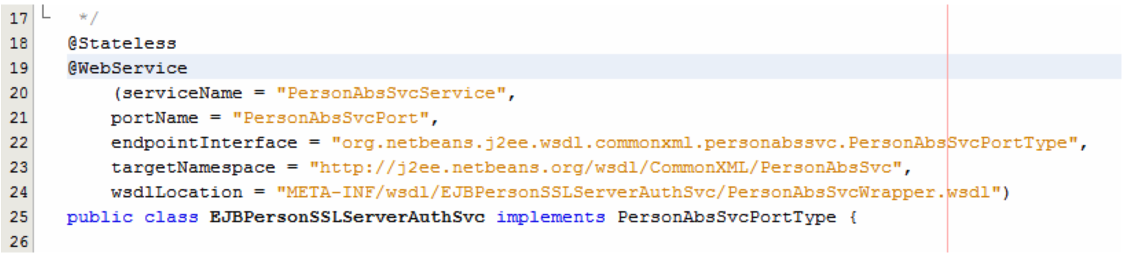
Figure 5.11.3 *Implementation code*

Move the “@Stateless” annotation from its location on line 23 to just before the “@WebService annotation”. Figures 5.11.4 and 5.11.5 illustrate the “before” and “after” state.



```
  */
@WebService
    (serviceName = "PersonAbsSvcService",
    portName = "PersonAbsSvcPort",
    endpointInterface = "org.netbeans.j2ee.wsdl.commonxml.personabssvc.PersonAbsSvcPortType",
    targetNamespace = "http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc",
    wsdlLocation = "META-INF/wsdl/EJBPersonSSLServerAuthSvc/PersonAbsSvcWrapper.wsdl")
@Stateless
public class EJBPersonSSLServerAuthSvc implements PersonAbsSvcPortType {
```

Figure 5.11.4 *"Before" state*



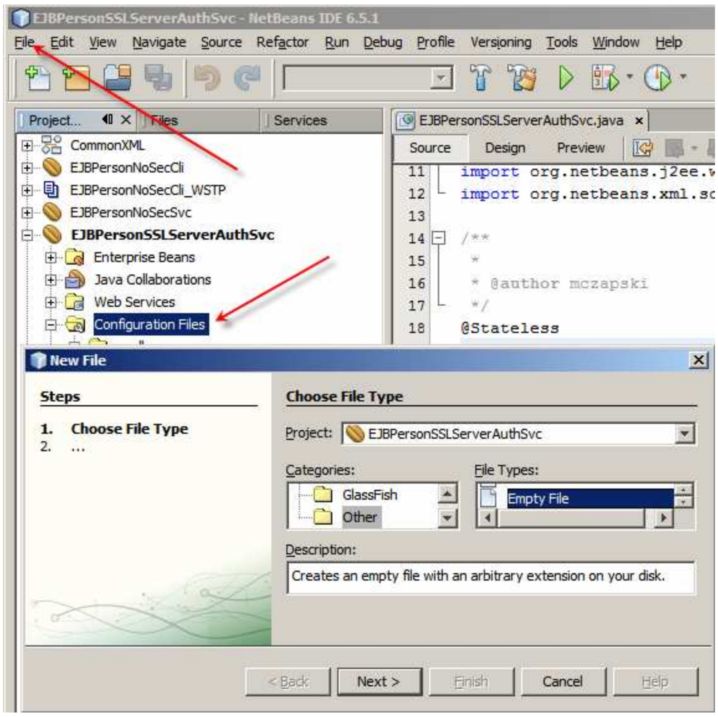
```
  */
@Stateless
@WebService
    (serviceName = "PersonAbsSvcService",
    portName = "PersonAbsSvcPort",
    endpointInterface = "org.netbeans.j2ee.wsdl.commonxml.personabssvc.PersonAbsSvcPortType",
    targetNamespace = "http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc",
    wsdlLocation = "META-INF/wsdl/EJBPersonSSLServerAuthSvc/PersonAbsSvcWrapper.wsdl")
public class EJBPersonSSLServerAuthSvc implements PersonAbsSvcPortType {
```

Figure 5.11.5 *"After" state*

The empirical reason for this is that processing of the @WebService annotation before the @Stateless annotation generates code that ignores channel security.

Build the project.

Click on the "Configuration Files" folder to select it. Click on the drop down "File" NetBeans menu. Choose "New File" -> "Other" -> "Empty File". Figure 5.11.6 illustrates a step in this process.



```
import org.netbeans.j2ee.w
import org.netbeans.xml.sc

/**
 *
 * @author mczapski
 */
@Stateless
```

New File

Steps

1. Choose File Type
2. ...

Choose File Type

Project: EJBPersonSSLServerAuthSvc

Categories: GlassFish, Other

File Types: Empty File

Description: Creates an empty file with an arbitrary extension on your disk.

Figure 5.11.6 *Create Empty File in the Configuration Files folder*

Name the new file “sun-ejb-jar.xml”.

Paste the XML content shown in Listing 5.10.2 into the sun-ejb-jar.xml.

Listing 5.11.2 *Content of the sun-ejb-sar.xml*

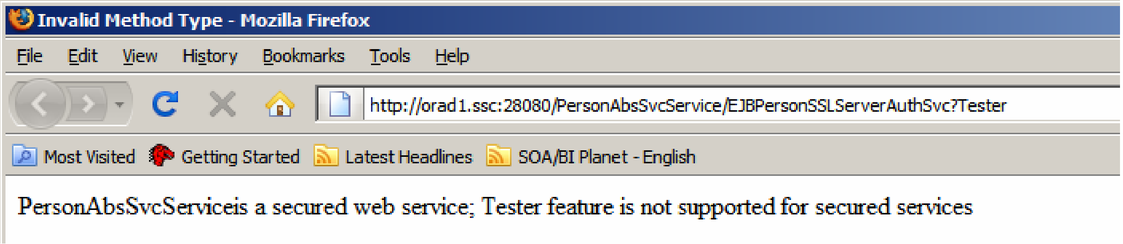
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD
Application Server 9.0 EJB 3.0//EN'
'http://www.sun.com/software/appserver/dtds/sun-ejb-jar_3_0-0.dtd'>
<sun-ejb-jar>
    <enterprise-beans>
        <ejb>
            <ejb-name>EJBPersonSSLServerAuthSvc</ejb-name>
            <webservice-endpoint>
                <port-component-name>EJBPersonSSLServerAuthSvc</port-component-name>
                <transport-guarantee>CONFIDENTIAL</transport-guarantee>
            </webservice-endpoint>
        </ejb>
    </enterprise-beans>
</sun-ejb-jar>
```

This seems to be the standard way of requiring SSL-based channel security for EJB-based web endpoints deployed to the GlassFish Application Server. See http://java.sun.com/developer/EJTechTips/2006/tt0527.html for discussion. The ejb-name and port-component-name values are the name of the implementation class, EJBPersonSSLServerAuthSvc.

Save the project, Build and Deploy.

Expand the “Web Service” node, right click on the port name and choose “Test Web Service”.

When the Web Browser opens, with the web service tester URL, there will be an error message in the content windows. Figure 5.17.7 shows the error message in the Firefox browser.



http://orad1.ssc:28080/PersonAbsSvcService/EJBPersonSSLServerAuthSvc?Tester

PersonAbsSvcServiceis a secured web service; Tester feature is not supported for secured services

Figure 5.11.7 *Error invoking web service test functionality*

Clearly, there is no way to access the service over non-secure channel.

For me, because I don’t use default port numbers, the secure channel URL will be:

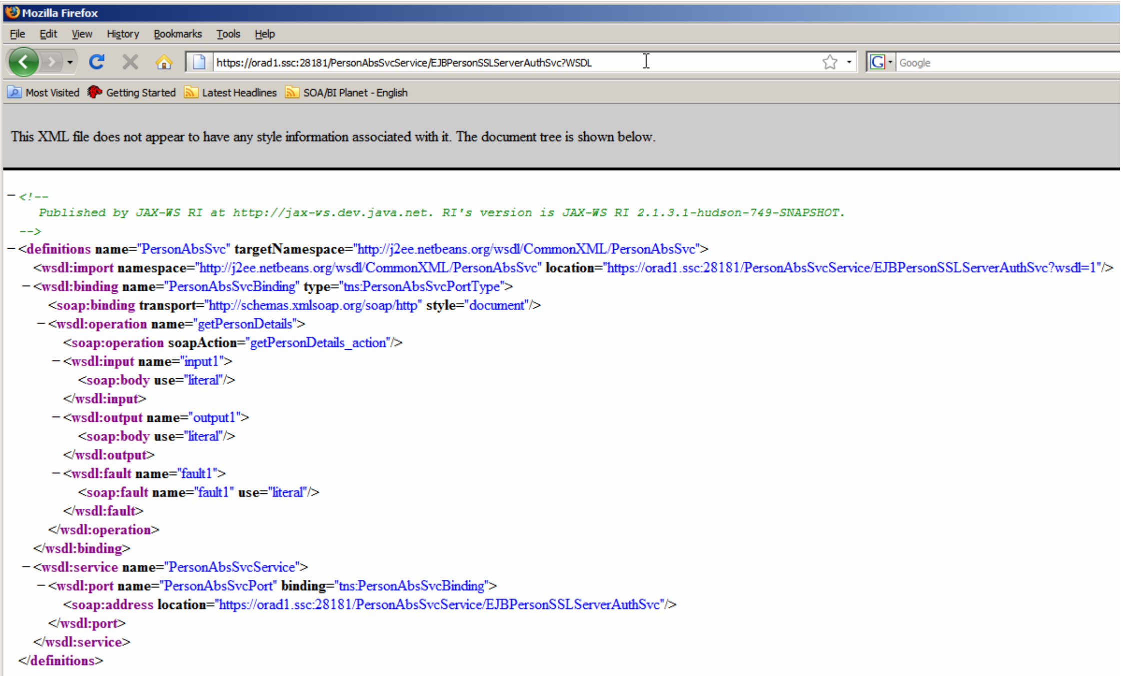
`https://orad1.ssc:28181/PersonAbsSvcService/EJBPersonSSLServerAuthSvc`

The default port number of the SSL-enabled listener will be 8181.

Let’s attempt to access the WSDL for this service using the following URL:

`https://orad1.ssc:28181/PersonAbsSvcService/EJBPersonSSLServerAuthSvc?WSDL`

Access succeeds. Figure 5.11.8 shows the WSDL in Firefox.



Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://orad1.ssc:28181/PersonAbsSvcService/EJBPersonSSLServerAuthSvc?WSDL
Google
Most Visited
Getting Started
Latest Headlines
SOA/BI Planet - English

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <!--
    Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.3.1-hudson-749-SNAPSHOT.
  -->
- <definitions name="PersonAbsSvc" targetNamespace="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc">
    <wsdl:import namespace="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc" location="https://orad1.ssc:28181/PersonAbsSvcService/EJBPersonSSLServerAuthSvc?wsdl=1"/>
  - <wsdl:binding name="PersonAbsSvcBinding" type="tns:PersonAbsSvcPortType">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    - <wsdl:operation name="getPersonDetails">
        <soap:operation soapAction="getPersonDetails_action"/>
      - <wsdl:input name="input1">
          <soap:body use="literal"/>
        </wsdl:input>
      - <wsdl:output name="output1">
          <soap:body use="literal"/>
        </wsdl:output>
      - <wsdl:fault name="fault1">
          <soap:fault name="fault1" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
    </wsdl:binding>
  - <wsdl:service name="PersonAbsSvcService">
    - <wsdl:port name="PersonAbsSvcPort" binding="tns:PersonAbsSvcBinding">
        <soap:address location="https://orad1.ssc:28181/PersonAbsSvcService/EJBPersonSSLServerAuthSvc"/>
      </wsdl:port>
    </wsdl:service>
</definitions>
```

Figure 5.11.8 *WSDL obtained over the secure channel*

Let’s now try to use the non-secure channel to get the WSDL. For me, the URL will be:

```
http://orad1.ssc:28080/PersonAbsSvcService/EJBPersonSSLServerAuthSvc?WSDL
```

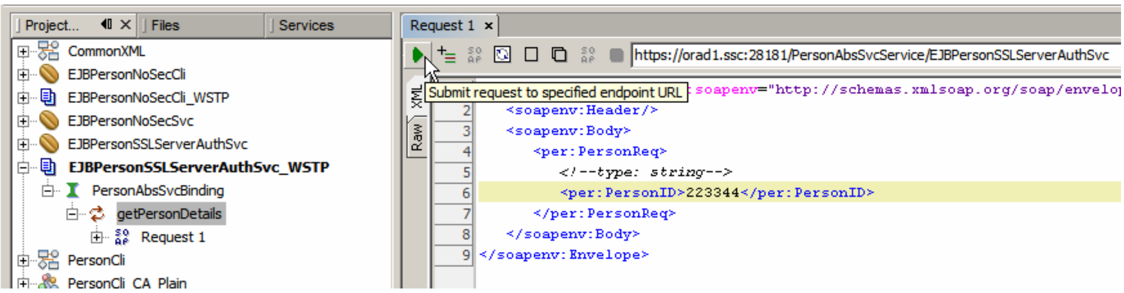
Access will fail. The browser will show a blank page and the server.log will have a WARNING-level message:

```
[#|2009-09-19T11:46:16.556+1000|WARNING|sun-
appserver2.1|javax.enterprise.system.container.ejb|_ThreadID=30;_ThreadName=http
SSLWorkerThread-28080-2;_RequestID=b4cbae17-fdd9-4051-991b-71482e166246;|Invalid
request scheme for Endpoint EJBPersonSSLServerAuthSvc. Expected https . Received
http|#]
```

Let’s now create a “New Project” -> “Java EE” -> “Web Service Testing Project”, named EJBPersonSSLServerAuthSvc_WSTP. Use the secure WSDL URL, which for me will be:

```
https://orad1.ssc:28181/PersonAbsSvcService/EJBPersonSSLServerAuthSvc?WSDL
```

Create, populate and submit a SOAP request. Figure 5.11.9 illustrates the project and the request.

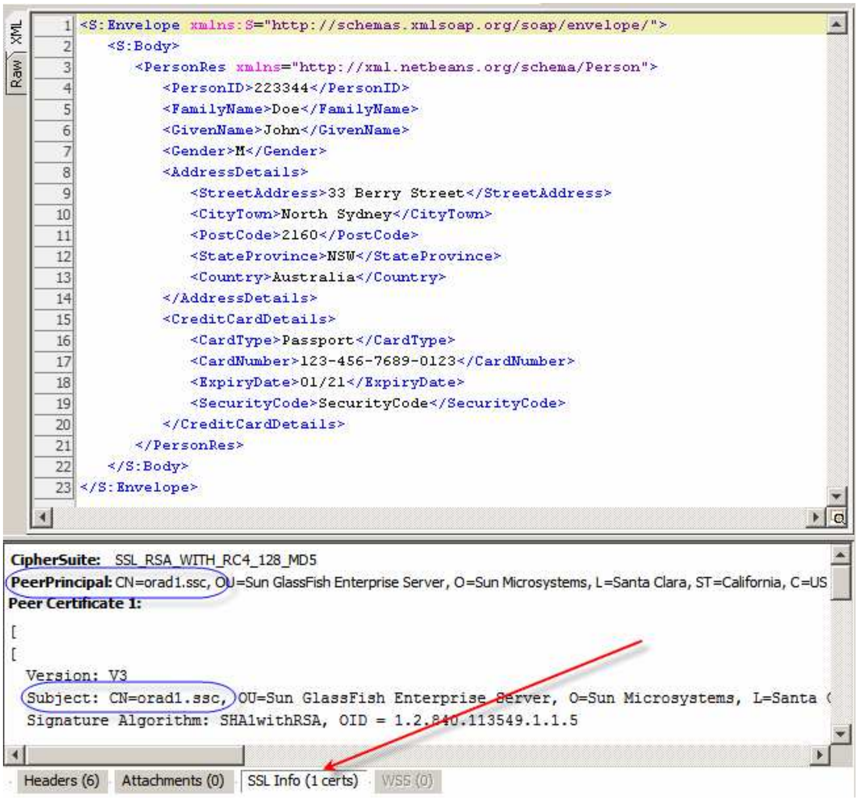


https://orad1.ssc:28181/PersonAbsSvcService/EJBPersonSSLServerAuthSvc

```
                        soapenv="http://schemas.xmlsoap.org/soap/envelo
    <soapenv:Header/>
    <soapenv:Body>
       <per:PersonReq>
          <!--type: string-->
          <per:PersonID>223344</per:PersonID>
       </per:PersonReq>
    </soapenv:Body>
</soapenv:Envelope>
```

Figure 5.11.9 *Submit SOAP request*

Figure 5.11.10 shows the SOAP Response, the X.509 Certificate and the Crypto Suite, which was provided and negotiated by the GlassFish Application Server to the SoapUI client during the SSL Handshake.



```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <PersonRes xmlns="http://xml.netbeans.org/schema/Person">
         <PersonID>223344</PersonID>
         <FamilyName>Doe</FamilyName>
         <GivenName>John</GivenName>
         <Gender>M</Gender>
         <AddressDetails>
            <StreetAddress>33 Berry Street</StreetAddress>
            <CityTown>North Sydney</CityTown>
            <PostCode>2160</PostCode>
            <StateProvince>NSW</StateProvince>
            <Country>Australia</Country>
         </AddressDetails>
         <CreditCardDetails>
            <CardType>Passport</CardType>
            <CardNumber>123-456-7689-0123</CardNumber>
            <ExpiryDate>01/21</ExpiryDate>
            <SecurityCode>SecurityCode</SecurityCode>
         </CreditCardDetails>
      </PersonRes>
   </S:Body>
</S:Envelope>
```

```
CipherSuite:   SSL_RSA_WITH_RC4_128_MD5
PeerPrincipal: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa Clara, ST=California, C=US
Peer Certificate 1:
[
[
  Version: V3
  Subject: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa (
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5
```

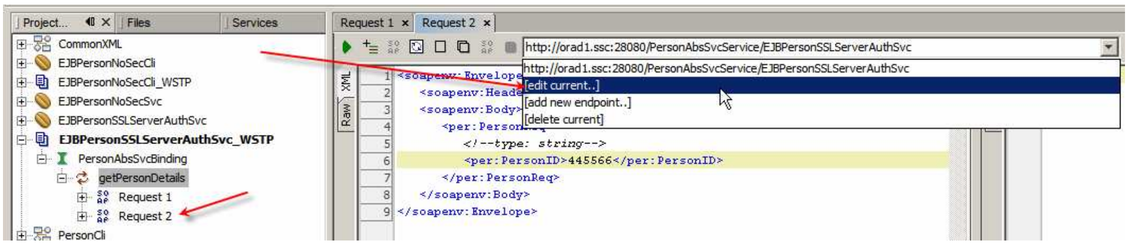
Figure 5.11.10 *SOAP Response and SSL information*

SSL Handshake can be logged to the server.log by adding "-Djavax.net.debug=ssl:handshake" to the GlassFish Application Server's JVM Options. Figure 5.8.1 illustrates this in the GlassFish Application Server Admin Console.

The web service, secured using SSL with Server-side Authentication, is operational. Let's see if we can access it using a non-secure URL, which for me would be:

`http://orad1.ssc:28080/PersonAbsSvcService/EJBPersonSSLServerAuthSvc`

Create a new SOAP Request in the EJBPersonSSLServerAuthSvc_WSTP project and change the endpoint URL to the non-secure one, as shown in Figure 5.11.11.



```
http://orad1.ssc:28080/PersonAbsSvcService/EJBPersonSSLServerAuthSvc
http://orad1.ssc:28080/PersonAbsSvcService/EJBPersonSSLServerAuthSvc
[edit current..]
[add new endpoint..]
[delete current]
```

```
<soapenv:Envelope
   <soapenv:Heade
   <soapenv:Body>
      <per:Person
         <!--type: string-->
         <per:PersonID>445566</per:PersonID>
      </per:PersonReq>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 5.11.11 *Create a new request and "[edit current …]" endpoint URL*

Submit the request.

The SoapUI will give a blank response, which is perhaps misleading, and the server.log will show the, familiar by now, WARNING-level message:

```
[#|2009-09-19T11:55:18.389+1000|WARNING|sun-
appserver2.1|javax.enterprise.system.container.ejb|_ThreadID=31;_ThreadName=http
SSLWorkerThread-28080-3;_RequestID=92f0cc01-3b4e-4e89-8776-1ee02f2b5fac;|Invalid
request scheme for Endpoint EJBPersonSSLServerAuthSvc. Expected https . Received
http|#]
```

The web service endpoint is secured, using SSL with Server-side Authentication.

5.11.2 EJB-based Web Service Client with Server-side Authentication

Let’s create an EJB-based web service client to exercise the service we just implemented.

Create a “New Project” -> “Java EE” -> “EJB Module”, named EJBPersonSSLServerAuthCli, with the local GlassFish instance as the deployment target.

Create “New” -> “Web Service From WSDL”, named EJBPersonSSLServerAuthCli, in package pkg.EJBPersonSSLServerAuthCli i, using WSDL from the CommonXML/TriggerCon.wsdl.

Create “New” -> “External WSDL Document(s)”, using the service WSDL URL. For me this will be:

```
https://orad1.ssc:28181/PersonAbsSvcService/EJBPersonSSLServerAuthSvc?WSDL
```

Figure 5.11.12 shows the new WSDL in the project structure.

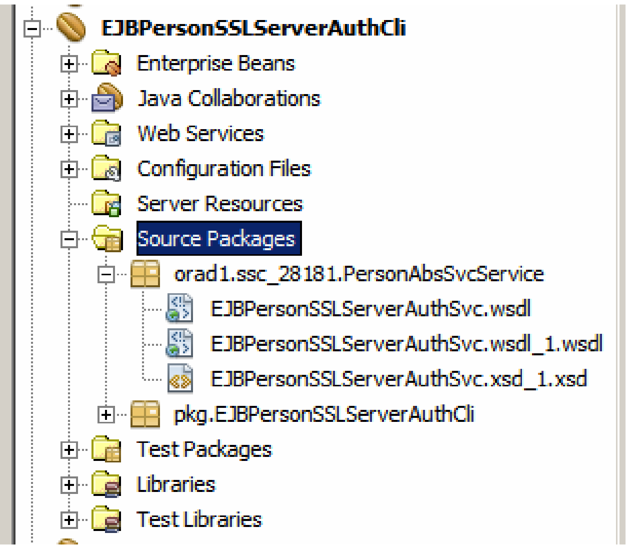
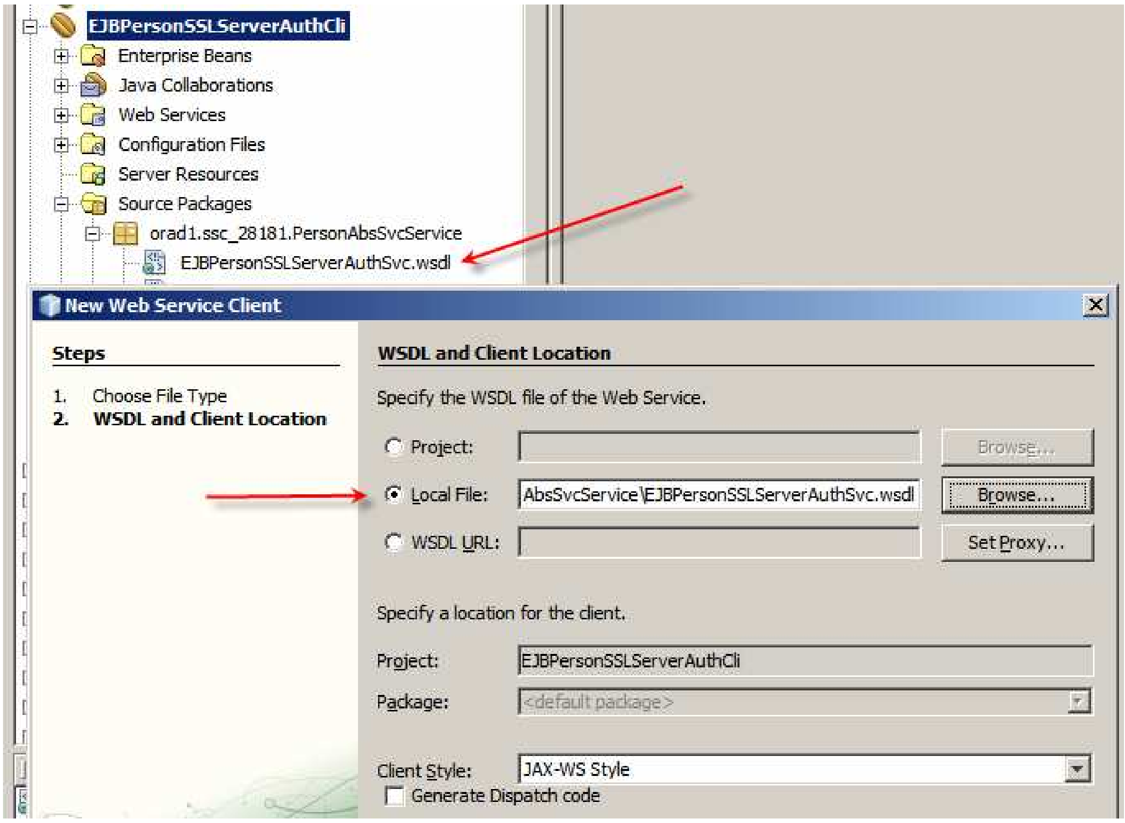

EJBPersonSSLServerAuthCli
Enterprise Beans
Java Collaborations
Web Services
Configuration Files
Server Resources
Source Packages
orad1.ssc_28181.PersonAbsSvcService
EJBPersonSSLServerAuthSvc.wsdl
EJBPersonSSLServerAuthSvc.wsdl_1.wsdl
EJBPersonSSLServerAuthSvc.xsd_1.xsd
pkg.EJBPersonSSLServerAuthCli
Test Packages
Libraries
Test Libraries

Figure 5.11.12 *WSDL I the project structure*

Create “New” -> “Web Service Client”, click “Local File”, Browse to the location of the WSDL EJBPersonSSLServerAuthSvc.wsdl and select it. Figure 5.11.13 illustrates this step.



Specify the WSDL file of the Web Service.

○ Project:

● Local File: AbsSvcService\EJBPersonSSLServerAuthSvc.wsdl

○ WSDL URL:

Specify a location for the client.

Project: EJBPersonSSLServerAuthCli

Package: <default package>

Client Style: JAX-WS Style

□ Generate Dispatch code

Figure 5.11.13 *Use WSDL in the local file*

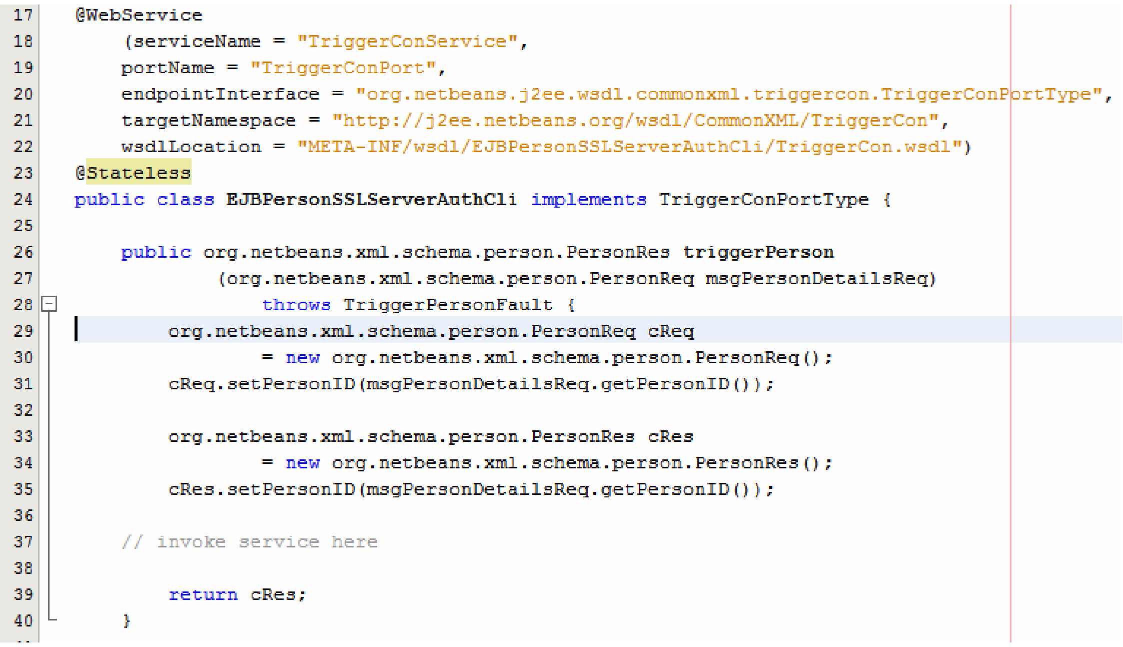
Open the Java source of the client implementation, if not already open.

Replace the two lines of code, “//TODO …” and “throw new Unsupported…” with the code shown in Listing 5.11.3.

Listing 5.11.3 Replacement code

```
        org.netbeans.xml.schema.person.PersonReq cReq
                = new org.netbeans.xml.schema.person.PersonReq();
        cReq.setPersonID(msgPersonDetailsReq.getPersonID());

        org.netbeans.xml.schema.person.PersonRes cRes
                = new org.netbeans.xml.schema.person.PersonRes();
        cRes.setPersonID(msgPersonDetailsReq.getPersonID());

       // invoke service here

        return cRes;
```

The source, reformatted for better readability, is shown in Figure 5.11.14.

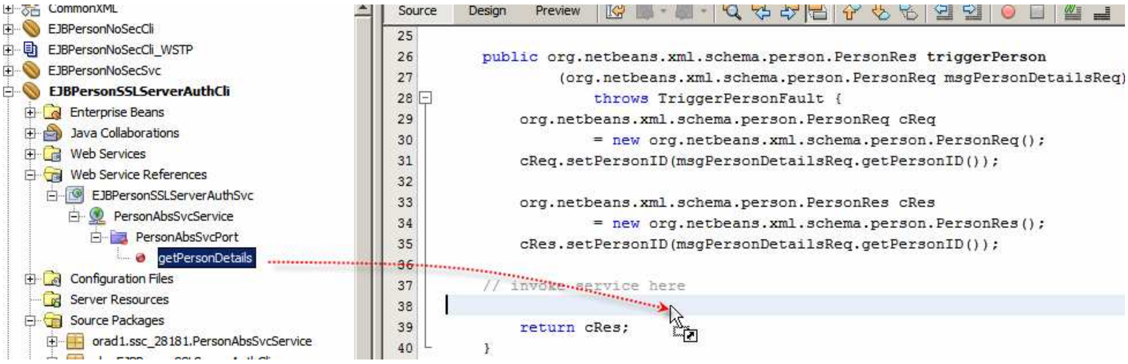


```
@WebService
    (serviceName = "TriggerConService",
    portName = "TriggerConPort",
    endpointInterface = "org.netbeans.j2ee.wsdl.commonxml.triggercon.TriggerConPortType",
    targetNamespace = "http://j2ee.netbeans.org/wsdl/CommonXML/TriggerCon",
    wsdlLocation = "META-INF/wsdl/EJBPersonSSLServerAuthCli/TriggerCon.wsdl")
@Stateless
public class EJBPersonSSLServerAuthCli implements TriggerConPortType {

    public org.netbeans.xml.schema.person.PersonRes triggerPerson
            (org.netbeans.xml.schema.person.PersonReq msgPersonDetailsReq)
                throws TriggerPersonFault {
        org.netbeans.xml.schema.person.PersonReq cReq
                = new org.netbeans.xml.schema.person.PersonReq();
        cReq.setPersonID(msgPersonDetailsReq.getPersonID());

        org.netbeans.xml.schema.person.PersonRes cRes
                = new org.netbeans.xml.schema.person.PersonRes();
        cRes.setPersonID(msgPersonDetailsReq.getPersonID());

    // invoke service here

        return cRes;
    }
```

Figure 5.11.14 *Client implementation code before service invocation is added*

Variables cReq and cRes will contain request to the service and response from the service respectively.

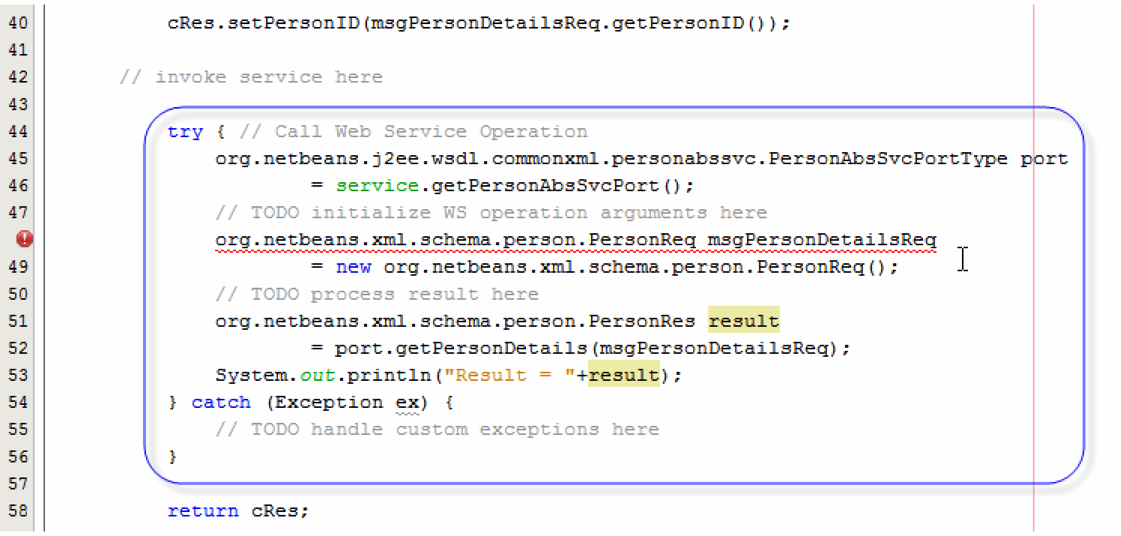
Expand Web Service References node all the way to the service operation and drag the service operation onto the source window following the comment "// invoke service here", as shown in Figure 5.11.15.



```
    public org.netbeans.xml.schema.person.PersonRes triggerPerson
            (org.netbeans.xml.schema.person.PersonReq msgPersonDetailsReq)
                throws TriggerPersonFault {
        org.netbeans.xml.schema.person.PersonReq cReq
                = new org.netbeans.xml.schema.person.PersonReq();
        cReq.setPersonID(msgPersonDetailsReq.getPersonID());

        org.netbeans.xml.schema.person.PersonRes cRes
                = new org.netbeans.xml.schema.person.PersonRes();
        cRes.setPersonID(msgPersonDetailsReq.getPersonID());

    // invoke service here

        return cRes;
    }
```

Figure 5.11.15 *Drag the service operation onto the source window*

The code fragment, which was generated by the NetBeans tooling, reformatted for better readability, is shown in Figure 5.11.16.



```
            cRes.setPersonID(msgPersonDetailsReq.getPersonID());

        // invoke service here

            try { // Call Web Service Operation
                org.netbeans.j2ee.wsdl.commonxml.personabssvc.PersonAbsSvcPortType port
                        = service.getPersonAbsSvcPort();
                // TODO initialize WS operation arguments here
                org.netbeans.xml.schema.person.PersonReq msgPersonDetailsReq
                        = new org.netbeans.xml.schema.person.PersonReq();
                // TODO process result here
                org.netbeans.xml.schema.person.PersonRes result
                        = port.getPersonDetails(msgPersonDetailsReq);
                System.out.println("Result = "+result);
            } catch (Exception ex) {
                // TODO handle custom exceptions here
            }

            return cRes;
```

Figure 5.11.16 *Service invocation skeleton code*

Replace the lines inside the try { … } catch block with the code in Listing 5.11.4.

Listing 5.11.4 Code to invoke the service and process the reply

```
org.netbeans.j2ee.wsdl.commonxml.personabssvc.PersonAbsSvcPortType port
        = service.getPersonAbsSvcPort();
org.netbeans.xml.schema.person.PersonRes result = port.getPersonDetails(cReq);

System.out.println("\n===>>> " + result.getFamilyName());

cRes.setFamilyName(result.getFamilyName());
cRes.setMiddleInitials(result.getMiddleInitials());
cRes.setGivenName(result.getGivenName());
cRes.setGender(result.getGender());

org.netbeans.xml.schema.person.PersonRes.AddressDetails cAddr
        = new org.netbeans.xml.schema.person.PersonRes.AddressDetails();
cAddr.setStreetAddress(result.getAddressDetails().getStreetAddress());
cAddr.setCityTown(result.getAddressDetails().getCityTown());
cAddr.setStateProvince(result.getAddressDetails().getStateProvince());
cAddr.setPostCode(result.getAddressDetails().getPostCode());
cAddr.setCountry(result.getAddressDetails().getCountry());
cRes.setAddressDetails(cAddr);

org.netbeans.xml.schema.person.PersonRes.CreditCardDetails cCard =
        new org.netbeans.xml.schema.person.PersonRes.CreditCardDetails();
cCard.setCardNumber(result.getCreditCardDetails().getCardNumber());
cCard.setCardType(result.getCreditCardDetails().getCardType());
cCard.setExpiryDate(result.getCreditCardDetails().getExpiryDate());
cCard.setSecurityCode(result.getCreditCardDetails().getSecurityCode());
cRes.setCreditCardDetails(cCard);
```

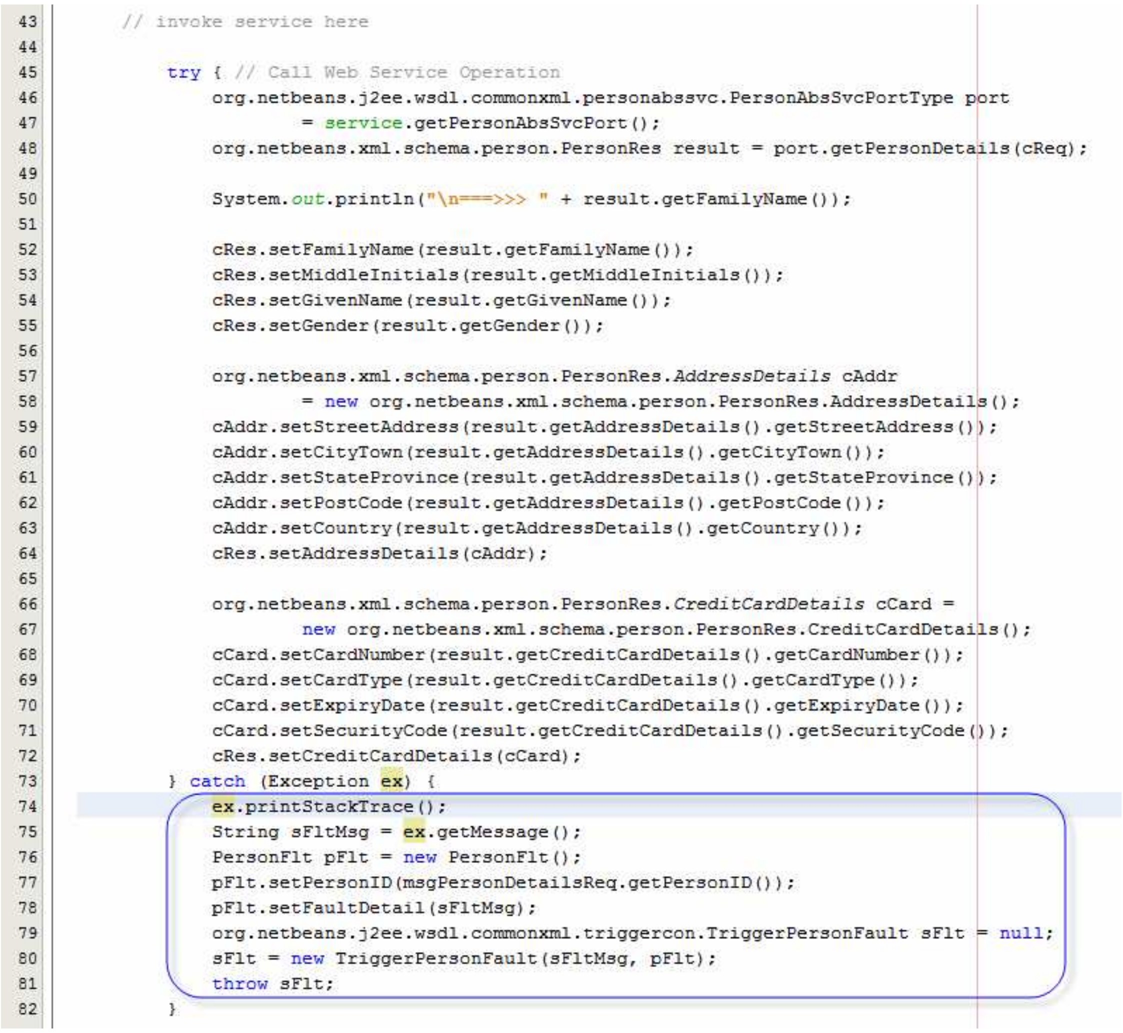
The new request, cReq, populated with the PersonID from the client, is submitted to the service and the response, cRes, is sued to populate the response of the client.

To complete client implementation let’s replace the one line comment “// TODO handle …” with the slab of code shown in Listing 5.11.5, then right-click inside the source window and choose “Fix Imports”.

Listing 5.11.5 Convert exception into a Fault

```
ex.printStackTrace();
String sFltMsg = ex.getMessage();
PersonFlt pFlt = new PersonFlt();
pFlt.setPersonID(msgPersonDetailsReq.getPersonID());
pFlt.setFaultDetail(sFltMsg);
org.netbeans.j2ee.wsdl.commonxml.triggercon.TriggerPersonFault sFlt =
null;
sFlt = new TriggerPersonFault(sFltMsg, pFlt);
throw sFlt;
```

Figure 5.11.17 shows the completed implementation code.



```
        // invoke service here

            try { // Call Web Service Operation
                org.netbeans.j2ee.wsdl.commonxml.personabssvc.PersonAbsSvcPortType port
                        = service.getPersonAbsSvcPort();
                org.netbeans.xml.schema.person.PersonRes result = port.getPersonDetails(cReq);

                System.out.println("\n===>>> " + result.getFamilyName());

                cRes.setFamilyName(result.getFamilyName());
                cRes.setMiddleInitials(result.getMiddleInitials());
                cRes.setGivenName(result.getGivenName());
                cRes.setGender(result.getGender());

                org.netbeans.xml.schema.person.PersonRes.AddressDetails cAddr
                        = new org.netbeans.xml.schema.person.PersonRes.AddressDetails();
                cAddr.setStreetAddress(result.getAddressDetails().getStreetAddress());
                cAddr.setCityTown(result.getAddressDetails().getCityTown());
                cAddr.setStateProvince(result.getAddressDetails().getStateProvince());
                cAddr.setPostCode(result.getAddressDetails().getPostCode());
                cAddr.setCountry(result.getAddressDetails().getCountry());
                cRes.setAddressDetails(cAddr);

                org.netbeans.xml.schema.person.PersonRes.CreditCardDetails cCard =
                        new org.netbeans.xml.schema.person.PersonRes.CreditCardDetails();
                cCard.setCardNumber(result.getCreditCardDetails().getCardNumber());
                cCard.setCardType(result.getCreditCardDetails().getCardType());
                cCard.setExpiryDate(result.getCreditCardDetails().getExpiryDate());
                cCard.setSecurityCode(result.getCreditCardDetails().getSecurityCode());
                cRes.setCreditCardDetails(cCard);
            } catch (Exception ex) {
                ex.printStackTrace();
                String sFltMsg = ex.getMessage();
                PersonFlt pFlt = new PersonFlt();
                pFlt.setPersonID(msgPersonDetailsReq.getPersonID());
                pFlt.setFaultDetail(sFltMsg);
                org.netbeans.j2ee.wsdl.commonxml.triggercon.TriggerPersonFault sFlt = null;
                sFlt = new TriggerPersonFault(sFltMsg, pFlt);
                throw sFlt;
            }
```

Figure 5.11.17 *Client implementation*

Build and Deploy the project.

The web service client we just built is itself a web service. We designed this client in this way so that we can use the SoapUI plugin to invoke the client, rather then go to the trouble of working out how else to invoke it and to build the appropriate implementation.

To create a SopUI project we need a WSDL URL. Expand the Web Service node in the client project, right-click the web service name and choose Test Web Service.

Once the web browser window opens with the test page, right-click the WSDL link and choose Copy Link Location, or equivalent, to copy WSDL URL to the clipboard.

For me the URL will be:

```
http://localhost:28080/TriggerConService/EJBPersonSSLServerAuthCli?WSDL
```

Default port number is 8080. I changed mine to 28080.

We can not actually test this service through the web browser. The service expects a structured message, conforming to Person XML Schema. We would have to construct a XML instance document and paste it into the text box. Even if we did go to that trouble the XML text would have gotten “escaped” by the form processor and the resulting message would have been garbled anyway.

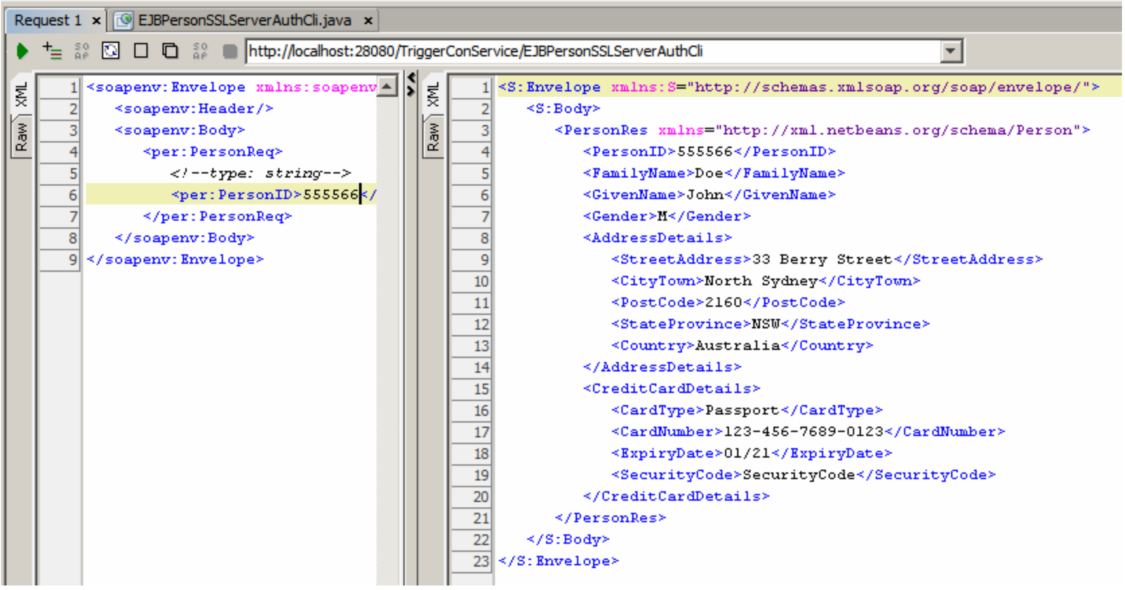
As mentioned, we will use the SoapUI to trigger the client.

Create “New Project” -> “Java EE” -> “Web Service Testing Project”, named EJBPersonSSLServerAuthCli_WSTP (this assumes that the SoapUI plugin has been installed – if it has not then now is the time to obtain and install it). Use the WSDL URL just copied to the clipboard as the service URL.

Once the project is created expand the nodes, right-click the service operation and choose “New request”.

Set PersonID to some favorite value and click the Submit request “button”.

If all goes well, the service response will look similar to that shown in Figure 5.11.18.


Request 1
EJBPersonSSLServerAuthCli.java
http://localhost:28080/TriggerConService/EJBPersonSSLServerAuthCli
XML
Raw
XML
Raw

```
<soapenv:Envelope xmlns:soapenv
   <soapenv:Header/>
   <soapenv:Body>
      <per:PersonReq>
         <!--type: string-->
         <per:PersonID>555566</
      </per:PersonReq>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <PersonRes xmlns="http://xml.netbeans.org/schema/Person">
         <PersonID>555566</PersonID>
         <FamilyName>Doe</FamilyName>
         <GivenName>John</GivenName>
         <Gender>M</Gender>
         <AddressDetails>
            <StreetAddress>33 Berry Street</StreetAddress>
            <CityTown>North Sydney</CityTown>
            <PostCode>2160</PostCode>
            <StateProvince>NSW</StateProvince>
            <Country>Australia</Country>
         </AddressDetails>
         <CreditCardDetails>
            <CardType>Passport</CardType>
            <CardNumber>123-456-7689-0123</CardNumber>
            <ExpiryDate>01/21</ExpiryDate>
            <SecurityCode>SecurityCode</SecurityCode>
         </CreditCardDetails>
      </PersonRes>
   </S:Body>
</S:Envelope>
```

Figure 5.11.18 *Service Response*

The service provider and the service consumer, communicating over secure channel using SSL with Server-side Authentication, were implemented and exercised. We can no undeploy both the client and the server.

Inspection of server.log of the local GlassFish instance will show the client-side of the SSL Handshake. Inspection of the server.log of the remote GlassFish instance will show the server-side of the SSL Handshake. The traces are worth studying at leas once to see what a normal handshake looks like. Whenever there are issues with handshake, as might be the case when server certificate is not trusted or the partner can not agree on the appropriate cipher suite to use, the SSL handshake trace will be the first place to look to figure out who is objecting and to what they are objecting.

5.12 EJB-based Person Svc with Mutual Authentication

Implement the Web Services Provider, following the steps in Section 5.11.1. As you do, name the service EJBPersonSSLMututalAuthSvc, instead of EJBPersonSSSServerAuthSvc.

When you get to Listing 5.11.2, instead of pasting the XML markup from that listing paste the XML markup from Listing 5.12.1.

Listing 5.12.1 sun-ejb-jar.xml with Mutual Authentication requirement

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD
Application Server 9.0 EJB 3.0//EN'
'http://www.sun.com/software/appserver/dtds/sun-ejb-jar_3_0-0.dtd'>
<sun-ejb-jar>
    <enterprise-beans>
        <ejb>
            <ejb-name>EJBPersonSSLMutualAuthSvc</ejb-name>
            <webservice-endpoint>
                <port-component-name>EJBPersonSSLMutualAuthSvc</port-component-name>
                  <login-config>
                    <auth-method>CLIENT-CERT</auth-method>
                    <realm>certificate</realm>
                  </login-config>
                <transport-guarantee>CONFIDENTIAL</transport-guarantee>
            </webservice-endpoint>
        </ejb>
    </enterprise-beans>
</sun-ejb-jar>
```

Stop one you have the project built and deployed.

Skip to Section 5.11.2 and follow the steps there to build the client for this service, making sure to name the client EJBPersonSSLMutualAuthCli

Once the SAOP Request is submitted and the response is received inspect the server.log on the client side to see the client view of the SSL Handshake. The handshake trace from my client, somewhat abbreviated, is shown in Listing 5.12.2. It can be seen that several handshakes were exchanged between the request being sent and the response being returned. This could be NetBeans being chatty about something or another.

Listing 5.12.2 Client-side SSL Handshake trace

```
[#|2009-09-19T17:02:23.968+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|============|#]
[#|2009-09-19T17:02:24.093+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
%% No cached client session|#]
[#|2009-09-19T17:02:24.093+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** ClientHello, TLSv1|#]
[#|2009-09-19T17:02:24.093+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
RandomCookie:  |#]
[#|2009-09-19T17:02:24.093+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|GMT: 1236500992 |#]
...
[#|2009-09-19T17:02:24.093+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Session ID:  |#]
[#|2009-09-19T17:02:24.093+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|{}|#]
[#|2009-09-19T17:02:24.093+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Cipher Suites: [SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA,
SSL_RSA_WITH_DES_CBC_SHA, SSL_DHE_RSA_WITH_DES_CBC_SHA, SSL_DHE_DSS_WITH_DES_CBC_SHA,
SSL_RSA_EXPORT_WITH_RC4_40_MD5, SSL_RSA_EXPORT_WITH_DES40_CBC_SHA,
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA, SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA]|#]
[#|2009-09-19T17:02:24.093+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Compression Methods:  { |#]
[#|2009-09-19T17:02:24.093+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|0|#]
...
[#|2009-09-19T17:02:24.093+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Handshake, length = 73|#]
[#|2009-09-19T17:02:24.093+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: SSLv2 client hello message, length = 98|#]
[#|2009-09-19T17:02:24.375+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, READ: TLSv1 Handshake, length = 794|#]
[#|2009-09-19T17:02:24.375+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** ServerHello, TLSv1|#]
```

```
[#|2009-09-19T17:02:24.375+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
RandomCookie:  |#]
[#|2009-09-19T17:02:24.375+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|GMT: 1236500992 |#]
[#|2009-09-19T17:02:24.375+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|bytes = { |#]
...
[#|2009-09-19T17:02:24.390+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Session ID:  |#]
[#|2009-09-19T17:02:24.390+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|{74, 180, 130, 0, 143, 164, 242, 159, 60, 221, 46, 129, 114, 200, 246, 71, 14,
33, 39, 208, 85, 216, 218, 234, 12, 96, 86, 28, 68, 40, 236, 150}|#]
[#|2009-09-19T17:02:24.390+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Cipher Suite: SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-19T17:02:24.390+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Compression Method: 0|#]
[#|2009-09-19T17:02:24.390+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
***|#]
[#|2009-09-19T17:02:24.390+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
%% Created:  [Session-1, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-19T17:02:24.390+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
** SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-19T17:02:24.390+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** Certificate chain|#]
[#|2009-09-19T17:02:24.390+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
chain [0] = [
[
  Version: V3
  Subject: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus:
10212154115706506957556825027043721498435033633815799809476856914850053601803189481914549 1
11331107933694939947203214020861835681616363720261872729871046530023033432450618494920847 4
44693317862325372611096817580008440551704374012071552944362437383385053028767344267433318 1
408072531272948503339758724948550982873
  public exponent: 65537
  Validity: [From: Sat Sep 05 13:48:28 EST 2009,
               To: Tue Sep 03 13:48:28 EST 2019]
  Issuer: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  SerialNumber: [    4aa1df8c]
Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 9E 79 9C E9 59 86 34 8F   FD 75 09 F7 82 D0 82 CE  .y..Y.4..u......
0010: BE 9A 44 EE                                        ..D.
]
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 23 A7 FD 51 1F 81 9E 8C   34 3A 58 01 EF 5A 04 CD  #..Q....4:X..Z..
0010: AD 35 2C 67 17 40 3A B8   EA 19 37 DB B2 B3 C8 EA  .5,g.@:...7.....
0020: 5B 4F 0E 30 4E 9D 42 23   52 FE E8 53 44 8B 64 21  [O.0N.B#R..SD.d!
0030: CF 5F EE 07 D5 60 1E F2   1B EA 68 99 E4 BB 6C 89  ._...`....h...l.
0040: 02 21 1D A5 AE 6C 26 14   8C 92 02 92 E3 C1 74 56  .!...l&.......tV
0050: 6A 69 96 8E 2D 1E 7D 6C   52 5E 99 38 20 8B 19 C4  ji..-..lR^.8 ...
0060: 52 11 89 B3 73 D0 6C 61   B2 DB BF CA 58 0A 3A 5D  R...s.la....X.:]
0070: 40 81 97 CC 3F 60 A6 1E   B5 D6 60 8A C6 6B B6 F6  @...?`....`..k..

]|#]
[#|2009-09-19T17:02:24.390+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
***|#]
[#|2009-09-19T17:02:24.390+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Found trusted certificate:|#]
[#|2009-09-19T17:02:24.390+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
[
[
  Version: V3
  Subject: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus:
102121541157065069575568250270437214984350336338157998094768569148500536018031894819145491
113311079336949399472032140208618356816163637202618727298710465300230334324506184949208474
446933178623253726110968175800084405517043740120715529443624373833850530287673442674333181
408072531272948503339758724948550982873
  public exponent: 65537
  Validity: [From: Sat Sep 05 13:48:28 EST 2009,
               To: Tue Sep 03 13:48:28 EST 2019]
  Issuer: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  SerialNumber: [    4aa1df8c]

Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 9E 79 9C E9 59 86 34 8F   FD 75 09 F7 82 D0 82 CE  .y..Y.4..u......
0010: BE 9A 44 EE                                        ..D.
]
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 23 A7 FD 51 1F 81 9E 8C   34 3A 58 01 EF 5A 04 CD  #..Q....4:X..Z..
0010: AD 35 2C 67 17 40 3A B8   EA 19 37 DB B2 B3 C8 EA  .5,g.@:...7.....
0020: 5B 4F 0E 30 4E 9D 42 23   52 FE E8 53 44 8B 64 21  [O.0N.B#R..SD.d!
0030: CF 5F EE 07 D5 60 1E F2   1B EA 68 99 E4 BB 6C 89  ._...`....h...l.
0040: 02 21 1D A5 AE 6C 26 14   8C 92 02 92 E3 C1 74 56  .!...l&.......tV
0050: 6A 69 96 8E 2D 1E 7D 6C   52 5E 99 38 20 8B 19 C4  ji..-..lR^.8 ...
0060: 52 11 89 B3 73 D0 6C 61   B2 DB BF CA 58 0A 3A 5D  R...s.la....X.:]
0070: 40 81 97 CC 3F 60 A6 1E   B5 D6 60 8A C6 6B B6 F6  @...?`....`..k..

]|#]
[#|2009-09-19T17:02:24.390+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** ServerHelloDone|#]
[#|2009-09-19T17:02:24.406+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** ClientKeyExchange, RSA PreMasterSecret, TLSv1|#]
[#|2009-09-19T17:02:24.406+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Handshake, length = 134|#]
[#|2009-09-19T17:02:24.406+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
SESSION KEYGEN:|#]
[#|2009-09-19T17:02:24.406+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
PreMaster Secret:|#]
...
...
[#|2009-09-19T17:02:24.421+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|CONNECTION KEYGEN:|#]
[#|2009-09-19T17:02:24.421+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Client Nonce:|#]
[#|2009-09-19T17:02:24.421+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:24.421+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Server Nonce:|#]
[#|2009-09-19T17:02:24.421+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:24.421+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Master Secret:|#]
[#|2009-09-19T17:02:24.421+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
```

```
[#|2009-09-19T17:02:24.437+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Client MAC write Secret:|#]
[#|2009-09-19T17:02:24.437+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:24.437+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Server MAC write Secret:|#]
[#|2009-09-19T17:02:24.437+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:24.437+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Client write key:|#]
[#|2009-09-19T17:02:24.437+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:24.437+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Server write key:|#]
[#|2009-09-19T17:02:24.437+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:24.437+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
... no IV used for this cipher|#]
[#|2009-09-19T17:02:24.437+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Change Cipher Spec, length = 1|#]
[#|2009-09-19T17:02:24.437+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** Finished|#]
[#|2009-09-19T17:02:24.437+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
verify_data:  { |#]
...
[#|2009-09-19T17:02:24.453+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Handshake, length = 32|#]
[#|2009-09-19T17:02:24.703+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, READ: TLSv1 Change Cipher Spec, length = 1|#]
[#|2009-09-19T17:02:24.703+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, READ: TLSv1 Handshake, length = 32|#]
```

```
[#|2009-09-19T17:02:24.703+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** Finished|#]
[#|2009-09-19T17:02:24.703+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
verify_data:  { |#]
...
[#|2009-09-19T17:02:24.703+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
%% Cached client session: [Session-1, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-19T17:02:24.718+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Application Data, length = 222|#]
[#|2009-09-19T17:02:24.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, READ: TLSv1 Application Data, length = 2056|#]
[#|2009-09-19T17:02:24.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, called close()|#]
[#|2009-09-19T17:02:24.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, called closeInternal(true)|#]
[#|2009-09-19T17:02:24.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3|#]
[#|2009-09-19T17:02:24.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|, SEND TLSv1 ALERT:  |#]
[#|2009-09-19T17:02:24.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|warning, |#]
[#|2009-09-19T17:02:24.828+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|description = close_notify|#]
[#|2009-09-19T17:02:24.828+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Alert, length = 18|#]
```

...

XML Request sent

...

```
[#|2009-09-19T17:02:24.984+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
%% Client cached [Session-1, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-19T17:02:24.984+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
%% Try resuming [Session-1, SSL_RSA_WITH_RC4_128_MD5] from port 1481|#]
[#|2009-09-19T17:02:24.984+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** ClientHello, TLSv1|#]
[#|2009-09-19T17:02:24.984+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
RandomCookie:  |#]
[#|2009-09-19T17:02:24.984+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|GMT: 1236500992 |#]
[#|2009-09-19T17:02:24.984+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|bytes = { |#]
...
[#|2009-09-19T17:02:25.000+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Session ID:  |#]
[#|2009-09-19T17:02:25.000+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|{74, 180, 130, 0, 143, 164, 242, 159, 60, 221, 46, 129, 114, 200, 246, 71, 14,
33, 39, 208, 85, 216, 218, 234, 12, 96, 86, 28, 68, 40, 236, 150}|#]
[#|2009-09-19T17:02:25.000+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Cipher Suites: [SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA,
SSL_RSA_WITH_DES_CBC_SHA, SSL_DHE_RSA_WITH_DES_CBC_SHA, SSL_DHE_DSS_WITH_DES_CBC_SHA,
SSL_RSA_EXPORT_WITH_RC4_40_MD5, SSL_RSA_EXPORT_WITH_DES40_CBC_SHA,
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA, SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA]|#]
[#|2009-09-19T17:02:25.000+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Compression Methods:  { |#]
[#|2009-09-19T17:02:25.000+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|0|#]
...
[#|2009-09-19T17:02:25.000+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Handshake, length = 105|#]
[#|2009-09-19T17:02:25.250+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, READ: TLSv1 Handshake, length = 74|#]
[#|2009-09-19T17:02:25.250+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** ServerHello, TLSv1|#]
[#|2009-09-19T17:02:25.250+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
RandomCookie:  |#]
[#|2009-09-19T17:02:25.250+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|GMT: 1236500993 |#]
...
[#|2009-09-19T17:02:25.250+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Session ID:  |#]
[#|2009-09-19T17:02:25.250+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|{74, 180, 130, 0, 143, 164, 242, 159, 60, 221, 46, 129, 114, 200, 246, 71, 14,
33, 39, 208, 85, 216, 218, 234, 12, 96, 86, 28, 68, 40, 236, 150}|#]
[#|2009-09-19T17:02:25.250+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Cipher Suite: SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-19T17:02:25.250+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Compression Method: 0|#]
[#|2009-09-19T17:02:25.250+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
***|#]
[#|2009-09-19T17:02:25.250+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
CONNECTION KEYGEN:|#]
[#|2009-09-19T17:02:25.250+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Client Nonce:|#]
...
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Server Nonce:|#]
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Master Secret:|#]
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Client MAC write Secret:|#]
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|Server MAC write Secret:|#]
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Client write key:|#]
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Server write key:|#]
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
... no IV used for this cipher|#]
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
%% Server resumed [Session-1, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-19T17:02:25.265+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, READ: TLSv1 Change Cipher Spec, length = 1|#]
[#|2009-09-19T17:02:25.281+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, READ: TLSv1 Handshake, length = 32|#]
[#|2009-09-19T17:02:25.281+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** Finished|#]
[#|2009-09-19T17:02:25.281+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
verify_data:  { |#]
...
[#|2009-09-19T17:02:25.281+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Change Cipher Spec, length = 1|#]
[#|2009-09-19T17:02:25.281+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** Finished|#]
[#|2009-09-19T17:02:25.281+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
verify_data:  { |#]
...
[#|2009-09-19T17:02:25.281+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Handshake, length = 32|#]
[#|2009-09-19T17:02:25.281+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Application Data, length = 369|#]
[#|2009-09-19T17:02:25.281+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Application Data, length = 227|#]
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, READ: TLSv1 Handshake, length = 20|#]
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** HelloRequest (empty)|#]
```

```
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
%% Client cached [Session-1, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
%% Try resuming [Session-1, SSL_RSA_WITH_RC4_128_MD5] from port 1481|#]
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** ClientHello, TLSv1|#]
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
RandomCookie:  |#]
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|GMT: 1236500993 |#]
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|bytes = { |#]
...
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Session ID:  |#]
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|{74, 180, 130, 0, 143, 164, 242, 159, 60, 221, 46, 129, 114, 200, 246, 71, 14,
33, 39, 208, 85, 216, 218, 234, 12, 96, 86, 28, 68, 40, 236, 150}|#]
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Cipher Suites: [SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA,
SSL_RSA_WITH_DES_CBC_SHA, SSL_DHE_RSA_WITH_DES_CBC_SHA, SSL_DHE_DSS_WITH_DES_CBC_SHA,
SSL_RSA_EXPORT_WITH_RC4_40_MD5, SSL_RSA_EXPORT_WITH_DES40_CBC_SHA,
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA, SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA]|#]
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Compression Methods:  { |#]
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|0|#]
...
[#|2009-09-19T17:02:25.578+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Handshake, length = 121|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, READ: TLSv1 Handshake, length = 6007|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** ServerHello, TLSv1|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
RandomCookie:  |#]
```

```
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|GMT: 1236500993 |#]
...
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Session ID:  |#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|{74, 180, 130, 1, 45, 141, 239, 238, 97, 61, 122, 55, 157, 168, 211, 13, 52,
40, 234, 35, 145, 230, 94, 127, 56, 108, 221, 85, 144, 175, 198, 155}|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Cipher Suite: SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Compression Method: 0|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
***|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
%% Created:  [Session-2, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
** SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** Certificate chain|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
chain [0] = [
[
  Version: V3
  Subject: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus:
102121541157065069575568250270437214984350336338157998094768569148500536018031894819145491
113311079336949399472032140208618356816163637202618727298710465300230334324506184949208474
446933178623253726110968175800084405517043740120715529443624373833850530287673442674333181
408072531272948503339758724948550982873
  public exponent: 65537
  Validity: [From: Sat Sep 05 13:48:28 EST 2009,
               To: Tue Sep 03 13:48:28 EST 2019]
  Issuer: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  SerialNumber: [    4aa1df8c]

Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 9E 79 9C E9 59 86 34 8F   FD 75 09 F7 82 D0 82 CE  .y..Y.4..u......
0010: BE 9A 44 EE                                        ..D.
]
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 23 A7 FD 51 1F 81 9E 8C   34 3A 58 01 EF 5A 04 CD  #..Q....4:X..Z..
0010: AD 35 2C 67 17 40 3A B8   EA 19 37 DB B2 B3 C8 EA  .5,g.@:...7.....
0020: 5B 4F 0E 30 4E 9D 42 23   52 FE E8 53 44 8B 64 21  [O.0N.B#R..SD.d!
0030: CF 5F EE 07 D5 60 1E F2   1B EA 68 99 E4 BB 6C 89  ._...`....h...l.
0040: 02 21 1D A5 AE 6C 26 14   8C 92 02 92 E3 C1 74 56  .!...l&.......tV
0050: 6A 69 96 8E 2D 1E 7D 6C   52 5E 99 38 20 8B 19 C4  ji..-..lR^.8 ...
0060: 52 11 89 B3 73 D0 6C 61   B2 DB BF CA 58 0A 3A 5D  R...s.la....X.:]
0070: 40 81 97 CC 3F 60 A6 1E   B5 D6 60 8A C6 6B B6 F6  @...?`....`..k..

]|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
***|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID                              hre
ad-28080-3;|
Found trusted certificate:|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
[
[
  Version: V3
  Subject: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus:
1021215411570650695755682502704372149843503363381579980947685691485005360180318948191454911
1133110793369493994720321402086183568161636372026187272987104653002303343245061849492084744
4469331786232537261109681758000844055170437401207155294436243738338505302876734426743331814
408072531272948503339758724948550982873
  public exponent: 65537
  Validity: [From: Sat Sep 05 13:48:28 EST 2009,
               To: Tue Sep 03 13:48:28 EST 2019]
  Issuer: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  SerialNumber: [    4aa1df8c]

Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 9E 79 9C E9 59 86 34 8F   FD 75 09 F7 82 D0 82 CE  .y..Y.4..u......
0010: BE 9A 44 EE                                        ..D.
]
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 23 A7 FD 51 1F 81 9E 8C   34 3A 58 01 EF 5A 04 CD  #..Q....4:X..Z..
0010: AD 35 2C 67 17 40 3A B8   EA 19 37 DB B2 B3 C8 EA  .5,g.@:...7.....
0020: 5B 4F 0E 30 4E 9D 42 23   52 FE E8 53 44 8B 64 21  [O.0N.B#R..SD.d!
0030: CF 5F EE 07 D5 60 1E F2   1B EA 68 99 E4 BB 6C 89  ._...`....h...l.
0040: 02 21 1D A5 AE 6C 26 14   8C 92 02 92 E3 C1 74 56  .!...l&.......tV
0050: 6A 69 96 8E 2D 1E 7D 6C   52 5E 99 38 20 8B 19 C4  ji..-..lR^.8 ...
```

Server Certificate
Remote GlassFish Instance

```
0060: 52 11 89 B3 73 D0 6C 61   B2 DB BF CA 58 0A 3A 5D  R...s.la....X.:]
0070: 40 81 97 CC 3F 60 A6 1E   B5 D6 60 8A C6 6B B6 F6  @...?`....`..k..

]|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** CertificateRequest|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Cert Types: |#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|RSA|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|, |#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|DSS|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Cert Authorities:|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
<OU=Equifax Secure Certificate Authority, O=Equifax, C=US>|#]
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
<OU=Starfield Class 2 Certification Authority, O="Starfield Technologies, Inc.", C=US>|#]
...
[#|2009-09-19T17:02:25.781+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
<CN=mcz02.aus.sun.com, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US>|#]
...
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
<EMAILADDRESS=info@valicert.com, CN=http://www.valicert.com/, OU=ValiCert Class 2 Policy
Validation Authority, O="ValiCert, Inc.", L=ValiCert Validation Network>|#]
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** ServerHelloDone|#]
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
matching alias: s1as|#]
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** Certificate chain|#]
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
chain [0] = [
[
  Version: V3
  Subject: CN=mcz02.aus.sun.com, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems,
L=Santa Clara, ST=California, C=US
```

Client Certificate
Local GlassFish Instance

```
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus:
1746102859671300769294336704019313699790561042748217269266583416724780785151439447464481 69
8897896288709765616514514195820377037054670957725633335611206988819359344975286317860424 57
4191542908592136669280383221023789302208617993989204068477274915593866491349676634055977 93
77027936960206647078681906925040575 0121
  public exponent: 65537
  Validity: [From: Wed Sep 02 16:05:22 EST 2009,
               To: Sat Aug 31 16:05:22 EST 2019]
  Issuer: CN=mcz02.aus.sun.com, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems,
L=Santa Clara, ST=California, C=US
  SerialNumber: [    4a9e0b22]

Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: F1 9E 66 16 11 83 58 9B   B7 1F 3E 8B BE 44 43 4D  ..f...X...>..DCM
0010: A8 BA 92 12                                        ....
]
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: DA 7B E3 80 06 24 01 A1   59 7D 97 BB 26 C7 3D BF  .....$..Y...&.=.
0010: B2 17 97 83 71 38 95 FA   DD 0E D8 A7 B7 E4 03 66  ....q8.........f
0020: 17 A7 69 89 5F 54 FD 96   22 BE 92 DE D0 C3 98 90  ..i._T.."........
0030: 22 B1 6A FE CE 38 9C 00   AD A7 3A 28 21 10 62 BE  ".j..8....:(!.b.
0040: 1D A9 58 B3 DA CE 3C 30   D0 7C 67 F3 CE 98 21 8A  ..X...<0..g...!.
0050: 62 A8 3B 88 ED 5C 6F 0F   C6 11 A0 0C 64 2E F1 13  b.;..\o.....d...
0060: 06 D1 A6 74 9B 63 81 56   DB 60 EE 22 92 A8 38 09  ...t.c.V.`."..8.
0070: B8 76 17 59 C0 5E 01 17   D3 AE AC 8F A2 61 48 4C  .v.Y.^.......aHL

]|#]
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
***|#]
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** ClientKeyExchange, RSA PreMasterSecret, TLSv1|#]
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Handshake, length = 882|#]
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
SESSION KEYGEN:|#]
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
PreMaster Secret:|#]
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
CONNECTION KEYGEN:|#]
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Client Nonce:|#]
[#|2009-09-19T17:02:25.796+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:25.812+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|Server Nonce:|#]
[#|2009-09-19T17:02:25.812+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:25.812+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Master Secret:|#]
[#|2009-09-19T17:02:25.812+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
[#|2009-09-19T17:02:25.812+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Client MAC write Secret:|#]
[#|2009-09-19T17:02:25.812+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
..
[#|2009-09-19T17:02:25.812+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Server MAC write Secret:|#]
[#|2009-09-19T17:02:25.812+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
..
[#|2009-09-19T17:02:25.812+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Client write key:|#]
[#|2009-09-19T17:02:25.812+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
..
[#|2009-09-19T17:02:25.812+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
Server write key:|#]
[#|2009-09-19T17:02:25.812+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
0000: |#]
...
```

```
[#|2009-09-19T17:02:25.812+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
... no IV used for this cipher|#]
[#|2009-09-19T17:02:25.843+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** CertificateVerify|#]
[#|2009-09-19T17:02:25.843+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Handshake, length = 150|#]
[#|2009-09-19T17:02:25.843+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Change Cipher Spec, length = 17|#]
[#|2009-09-19T17:02:25.843+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** Finished|#]
[#|2009-09-19T17:02:25.843+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
verify_data:  { |#]
...
[#|2009-09-19T17:02:25.843+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, WRITE: TLSv1 Handshake, length = 32|#]
[#|2009-09-19T17:02:26.062+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, READ: TLSv1 Change Cipher Spec, length = 17|#]
[#|2009-09-19T17:02:26.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, READ: TLSv1 Handshake, length = 32|#]
[#|2009-09-19T17:02:26.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
*** Finished|#]
[#|2009-09-19T17:02:26.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
verify_data:  { |#]
...
[#|2009-09-19T17:02:26.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
%% Cached client session: [Session-2, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-19T17:02:26.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
httpSSLWorkerThread-28080-3, READ: TLSv1 Application Data, length = 870|#]
[#|2009-09-19T17:02:26.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
---[HTTP response 200]---|#]
[#|2009-09-19T17:02:26.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=71;_ThreadName=httpSSLWorkerThre
ad-28080-3;|
null: HTTP/1.1 200 OK|#]
```

...

XML Response from the server

```
...
[#|2009-09-19T17:02:36.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=72;_ThreadName=Keep-Alive-
Timer;|
Keep-Alive-Timer, called close()|#]
[#|2009-09-19T17:02:36.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=72;_ThreadName=Keep-Alive-
Timer;|
Keep-Alive-Timer, called closeInternal(true)|#]
[#|2009-09-19T17:02:36.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=72;_ThreadName=Keep-Alive-
Timer;|
Keep-Alive-Timer|#]
[#|2009-09-19T17:02:36.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=72;_ThreadName=Keep-Alive-
Timer;|, SEND TLSv1 ALERT:  |#]
[#|2009-09-19T17:02:36.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=72;_ThreadName=Keep-Alive-
Timer;|warning, |#]
[#|2009-09-19T17:02:36.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=72;_ThreadName=Keep-Alive-
Timer;|description = close_notify|#]
[#|2009-09-19T17:02:36.078+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=72;_ThreadName=Keep-Alive-
Timer;|
Keep-Alive-Timer, WRITE: TLSv1 Alert, length = 18|#]
```

Looking at the server-side, remote GlassFish instance’s server.log we an exchange involving both the server and the client certificates. Listing 5.12.3 is the abbreviated server.log transcript.

Listing 5.12.3 Transcript of server.log with server-side SSL Handshake

```
[#|2009-09-19T17:02:25.431+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
***|#]
[#|2009-09-19T17:02:25.435+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
%% Invalidated:  [Session-20, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-19T17:02:25.435+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
*** HelloRequest (empty)|#]
[#|2009-09-19T17:02:25.435+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
httpSSLWorkerThread-28181-2, WRITE: TLSv1 Handshake, length = 20|#]
[#|2009-09-19T17:02:25.704+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
httpSSLWorkerThread-28181-2, READ: TLSv1 Handshake, length = 121|#]
[#|2009-09-19T17:02:25.704+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
*** ClientHello, TLSv1|#]
[#|2009-09-19T17:02:25.704+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
RandomCookie:  |#]
```

```
[#|2009-09-19T17:02:25.704+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|GMT: 1236500993 |#]
...
[#|2009-09-19T17:02:25.708+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Session ID:  |#]
[#|2009-09-19T17:02:25.708+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|{74, 180, 130, 0, 143, 164, 242, 159, 60, 221, 46, 129, 114, 200, 246, 71, 14,
33, 39, 208, 85, 216, 218, 234, 12, 96, 86, 28, 68, 40, 236, 150}|#]
[#|2009-09-19T17:02:25.708+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Cipher Suites: [SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA,
SSL_RSA_WITH_DES_CBC_SHA, SSL_DHE_RSA_WITH_DES_CBC_SHA, SSL_DHE_DSS_WITH_DES_CBC_SHA,
SSL_RSA_EXPORT_WITH_RC4_40_MD5, SSL_RSA_EXPORT_WITH_DES40_CBC_SHA,
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA, SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA]|#]
[#|2009-09-19T17:02:25.708+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Compression Methods:  { |#]
[#|2009-09-19T17:02:25.708+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|0|#]
[#|2009-09-19T17:02:25.708+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;| }|#]
[#|2009-09-19T17:02:25.708+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
***|#]
[#|2009-09-19T17:02:25.709+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
%% Created:  [Session-21, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-19T17:02:25.709+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
*** ServerHello, TLSv1|#]
[#|2009-09-19T17:02:25.709+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
RandomCookie:  |#]
...
[#|2009-09-19T17:02:25.712+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Session ID:  |#]
[#|2009-09-19T17:02:25.712+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|{74, 180, 130, 1, 45, 141, 239, 238, 97, 61, 122, 55, 157, 168, 211, 13, 52,
40, 234, 35, 145, 230, 94, 127, 56, 108, 221, 85, 144, 175, 198, 155}|#]
[#|2009-09-19T17:02:25.712+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Cipher Suite: SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-19T17:02:25.712+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Compression Method: 0|#]
```

```
[#|2009-09-19T17:02:25.712+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
***|#]
[#|2009-09-19T17:02:25.713+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Cipher suite:  SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-19T17:02:25.713+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
*** Certificate chain|#]
[#|2009-09-19T17:02:25.714+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
chain [0] = [
[
  Version: V3
  Subject: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus:
102121541157065069575568250270437214984350336338157998094768569148500536018031894819145491
113311079336949399472032140208618356816163637202618727298710465300230334324506184949208474
446933178623253726110968175800084405517043740120715529443624373833850530287673442674333181
408072531272948503339758724948550982873
  public exponent: 65537
  Validity: [From: Sat Sep 05 13:48:28 EST 2009,
               To: Tue Sep 03 13:48:28 EST 2019]
  Issuer: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  SerialNumber: [    4aa1df8c]

Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 9E 79 9C E9 59 86 34 8F   FD 75 09 F7 82 D0 82 CE  .y..Y.4..u......
0010: BE 9A 44 EE                                        ..D.
]
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 23 A7 FD 51 1F 81 9E 8C   34 3A 58 01 EF 5A 04 CD  #..Q....4:X..Z..
0010: AD 35 2C 67 17 40 3A B8   EA 19 37 DB B2 B3 C8 EA  .5,g.@:...7.....
0020: 5B 4F 0E 30 4E 9D 42 23   52 FE E8 53 44 8B 64 21  [O.0N.B#R..SD.d!
0030: CF 5F EE 07 D5 60 1E F2   1B EA 68 99 E4 BB 6C 89  ._...`....h...l.
0040: 02 21 1D A5 AE 6C 26 14   8C 92 02 92 E3 C1 74 56  .!...l&.......tV
0050: 6A 69 96 8E 2D 1E 7D 6C   52 5E 99 38 20 8B 19 C4  ji..-..lR^.8 ...
0060: 52 11 89 B3 73 D0 6C 61   B2 DB BF CA 58 0A 3A 5D  R...s.la....X.:]
0070: 40 81 97 CC 3F 60 A6 1E   B5 D6 60 8A C6 6B B6 F6  @...?`....`..k..

]|#]
[#|2009-09-19T17:02:25.714+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
***|#]
[#|2009-09-19T17:02:25.717+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
*** CertificateRequest|#]
```

Ask for client certificate

```
[#|2009-09-19T17:02:25.717+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Cert Types: |#]
[#|2009-09-19T17:02:25.717+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|RSA|#]
[#|2009-09-19T17:02:25.717+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|, |#]
[#|2009-09-19T17:02:25.717+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|DSS|#]
[#|2009-09-19T17:02:25.718+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Cert Authorities:|#]
[#|2009-09-19T17:02:25.718+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
<OU=Equifax Secure Certificate Authority, O=Equifax, C=US>|#]
...
[#|2009-09-19T17:02:25.723+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
<CN=mcz02.aus.sun.com, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US>|#]
...
[#|2009-09-19T17:02:25.726+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
<CN=VeriSign Class 3 Public Primary Certification Authority - G3, OU="(c) 1999 VeriSign,
Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US>|#]
[#|2009-09-19T17:02:25.760+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
<EMAILADDRESS=info@valicert.com, CN=http://www.valicert.com/, OU=ValiCert Class 2 Policy
Validation Authority, O="ValiCert, Inc.", L=ValiCert Validation Network>|#]
[#|2009-09-19T17:02:25.761+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
*** ServerHelloDone|#]
[#|2009-09-19T17:02:25.762+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
httpSSLWorkerThread-28181-2, WRITE: TLSv1 Handshake, length = 6007|#]
[#|2009-09-19T17:02:25.971+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
httpSSLWorkerThread-28181-2, READ: TLSv1 Handshake, length = 882|#]
[#|2009-09-19T17:02:25.971+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
*** Certificate chain|#]
[#|2009-09-19T17:02:26.007+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
chain [0] = [
[
  Version: V3
  Subject: CN=mcz02.aus.sun.com, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems,
L=Santa Clara, ST=California, C=US
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
```

Receive and validate client certificate

```
  modulus:
174610285967130076929433670401931369979056104274821726926658341672478078515143944746448169
889789628870976561651451419582037703705467095772563333561120698881935934497528631786042457
419154290859213666928038322102378930220861799398920406847727491559386649134967663405597793
770279369602066470786819069250405750121
  public exponent: 65537
  Validity: [From: Wed Sep 02 16:05:22 EST 2009,
               To: Sat Aug 31 16:05:22 EST 2019]
  Issuer: CN=mcz02.aus.sun.com, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems,
L=Santa Clara, ST=California, C=US
  SerialNumber: [    4a9e0b22]

Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: F1 9E 66 16 11 83 58 9B   B7 1F 3E 8B BE 44 43 4D  ..f...X...>..DCM
0010: A8 BA 92 12                                        ....
]
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: DA 7B E3 80 06 24 01 A1   59 7D 97 BB 26 C7 3D BF  .....$..Y...&.=.
0010: B2 17 97 83 71 38 95 FA   DD 0E D8 A7 B7 E4 03 66  ....q8.........f
0020: 17 A7 69 89 5F 54 FD 96   22 BE 92 DE D0 C3 98 90  ..i._T..".......
0030: 22 B1 6A FE CE 38 9C 00   AD A7 3A 28 21 10 62 BE  ".j..8....:(!.b.
0040: 1D A9 58 B3 DA CE 3C 30   D0 7C 67 F3 CE 98 21 8A  ..X...<0..g...!.
0050: 62 A8 3B 88 ED 5C 6F 0F   C6 11 A0 0C 64 2E F1 13  b.;..\o.....d...
0060: 06 D1 A6 74 9B 63 81 56   DB 60 EE 22 92 A8 38 09  ...t.c.V.`."..8.
0070: B8 76 17 59 C0 5E 01 17   D3 AE AC 8F A2 61 48 4C  .v.Y.^.......aHL

]|#]
[#|2009-09-19T17:02:26.007+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
***|#]
[#|2009-09-19T17:02:26.029+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Found trusted certificate:|#]
[#|2009-09-19T17:02:26.030+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
[
[
  Version: V3
  Subject: CN=mcz02.aus.sun.com, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems,
L=Santa Clara, ST=California, C=US
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus:
174610285967130076929433670401931369979056104274821726926658341672478078515143944746448169
889789628870976561651451419582037703705467095772563333561120698881935934497528631786042457
419154290859213666928038322102378930220861799398920406847727491559386649134967663405597793
770279369602066470786819069250405750121
  public exponent: 65537
  Validity: [From: Wed Sep 02 16:05:22 EST 2009,
               To: Sat Aug 31 16:05:22 EST 2019]
  Issuer: CN=mcz02.aus.sun.com, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems,
L=Santa Clara, ST=California, C=US
  SerialNumber: [    4a9e0b22]
```

I am happy with the client certificate

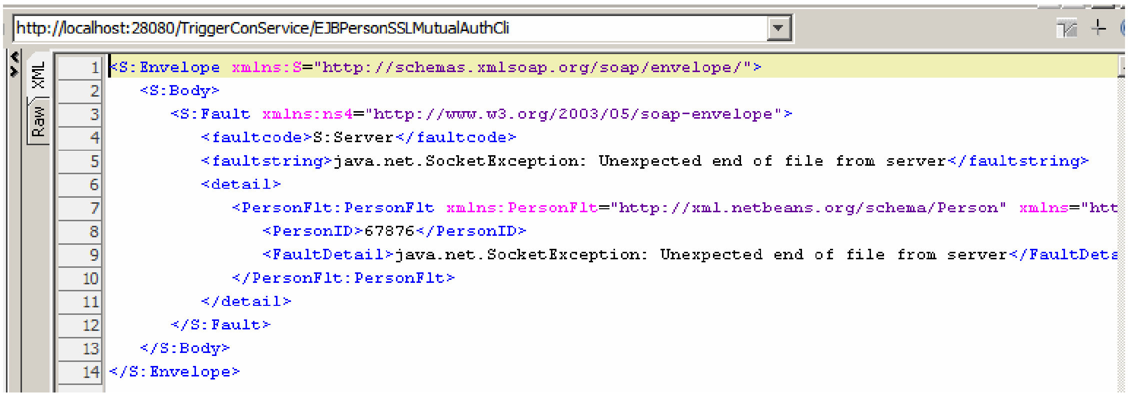
```
Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: F1 9E 66 16 11 83 58 9B   B7 1F 3E 8B BE 44 43 4D  ..f...X...>..DCM
0010: A8 BA 92 12                                        ....
]
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: DA 7B E3 80 06 24 01 A1   59 7D 97 BB 26 C7 3D BF  .....$..Y...&.=.
0010: B2 17 97 83 71 38 95 FA   DD 0E D8 A7 B7 E4 03 66  ....q8.........f
0020: 17 A7 69 89 5F 54 FD 96   22 BE 92 DE D0 C3 98 90  ..i._T..".......
0030: 22 B1 6A FE CE 38 9C 00   AD A7 3A 28 21 10 62 BE  ".j..8....:(!.b.
0040: 1D A9 58 B3 DA CE 3C 30   D0 7C 67 F3 CE 98 21 8A  ..X...<0..g...!.
0050: 62 A8 3B 88 ED 5C 6F 0F   C6 11 A0 0C 64 2E F1 13  b.;..\o.....d...
0060: 06 D1 A6 74 9B 63 81 56   DB 60 EE 22 92 A8 38 09  ...t.c.V.`."..8.
0070: B8 76 17 59 C0 5E 01 17   D3 AE AC 8F A2 61 48 4C  .v.Y.^.......aHL

]|#]
[#|2009-09-19T17:02:26.033+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
*** ClientKeyExchange, RSA PreMasterSecret, TLSv1|#]
[#|2009-09-19T17:02:26.033+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
SESSION KEYGEN:|#]
[#|2009-09-19T17:02:26.033+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
PreMaster Secret:|#]
...
[#|2009-09-19T17:02:26.040+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
CONNECTION KEYGEN:|#]
[#|2009-09-19T17:02:26.040+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Client Nonce:|#]
...
[#|2009-09-19T17:02:26.043+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|Server Nonce:|#]
...
[#|2009-09-19T17:02:26.047+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Master Secret:|#]
...
[#|2009-09-19T17:02:26.052+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Client MAC write Secret:|#]
...
[#|2009-09-19T17:02:26.054+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Server MAC write Secret:|#]
...
```

```
[#|2009-09-19T17:02:26.056+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Client write key:|#]
...
[#|2009-09-19T17:02:26.058+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
Server write key:|#]
...
[#|2009-09-19T17:02:26.059+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
... no IV used for this cipher|#]
[#|2009-09-19T17:02:26.060+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
httpSSLWorkerThread-28181-2, READ: TLSv1 Handshake, length = 150|#]
[#|2009-09-19T17:02:26.061+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
*** CertificateVerify|#]
[#|2009-09-19T17:02:26.062+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
httpSSLWorkerThread-28181-2, READ: TLSv1 Change Cipher Spec, length = 17|#]
[#|2009-09-19T17:02:26.062+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
httpSSLWorkerThread-28181-2, READ: TLSv1 Handshake, length = 32|#]
[#|2009-09-19T17:02:26.062+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
*** Finished|#]
[#|2009-09-19T17:02:26.062+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
verify_data:  { |#]
...
[#|2009-09-19T17:02:26.064+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
httpSSLWorkerThread-28181-2, WRITE: TLSv1 Change Cipher Spec, length = 17|#]
[#|2009-09-19T17:02:26.064+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
*** Finished|#]
[#|2009-09-19T17:02:26.064+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
verify_data:  { |#]
...
[#|2009-09-19T17:02:26.066+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
httpSSLWorkerThread-28181-2, WRITE: TLSv1 Handshake, length = 32|#]
[#|2009-09-19T17:02:26.066+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
%% Cached server session: [Session-21, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-19T17:02:26.090+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
---[HTTP request]---|#]
...
```

```
produce response
...
[#|2009-09-19T17:02:26.117+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
---[HTTP response 200]---|#]
[#|2009-09-19T17:02:26.117+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
<?xml version="1.0" ?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"><S:Body><PersonRes
xmlns="http://xml.netbeans.org/schema/Person"><PersonID>67876</PersonID><FamilyName>Doe</F
amilyName><GivenName>John</GivenName><Gender>M</Gender><AddressDetails><StreetAddress>33
Berry Street</StreetAddress><CityTown>North
Sydney</CityTown><PostCode>2160</PostCode><StateProvince>NSW</StateProvince><Country>Austr
alia</Country></AddressDetails><CreditCardDetails><CardType>Passport</CardType><CardNumber
>123-456-7689-
0123</CardNumber><ExpiryDate>01/21</ExpiryDate><SecurityCode>SecurityCode</SecurityCode></
CreditCardDetails></PersonRes></S:Body></S:Envelope>|#]
...
[#|2009-09-19T17:02:26.118+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=46;_ThreadName=httpSSLWorkerThre
ad-28181-2;|
httpSSLWorkerThread-28181-2, WRITE: TLSv1 Application Data, length = 854|#]
[#|2009-09-19T17:02:36.193+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=43;_ThreadName=httpSSLWorkerThre
ad-28181-0;|
httpSSLWorkerThread-28181-0, READ: TLSv1 Alert, length = 18|#]
[#|2009-09-19T17:02:36.194+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=43;_ThreadName=httpSSLWorkerThre
ad-28181-0;|
httpSSLWorkerThread-28181-0|#]
[#|2009-09-19T17:02:36.194+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=43;_ThreadName=httpSSLWorkerThre
ad-28181-0;|, RECV TLSv1 ALERT:  |#]
[#|2009-09-19T17:02:36.194+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=43;_ThreadName=httpSSLWorkerThre
ad-28181-0;|warning, |#]
[#|2009-09-19T17:02:36.194+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=43;_ThreadName=httpSSLWorkerThre
ad-28181-0;|close_notify|#]
[#|2009-09-19T17:02:36.194+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=43;_ThreadName=httpSSLWorkerThre
ad-28181-0;|
httpSSLWorkerThread-28181-0, closeInboundInternal()|#]
[#|2009-09-19T17:02:36.194+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=43;_ThreadName=httpSSLWorkerThre
ad-28181-0;|
httpSSLWorkerThread-28181-0, closeOutboundInternal()|#]
[#|2009-09-19T17:02:36.194+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=43;_ThreadName=httpSSLWorkerThre
ad-28181-0;|
httpSSLWorkerThread-28181-0|#]
[#|2009-09-19T17:02:36.194+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=43;_ThreadName=httpSSLWorkerThre
ad-28181-0;|, SEND TLSv1 ALERT:  |#]
[#|2009-09-19T17:02:36.194+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=43;_ThreadName=httpSSLWorkerThre
ad-28181-0;|warning, |#]
[#|2009-09-19T17:02:36.194+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=43;_ThreadName=httpSSLWorkerThre
ad-28181-0;|description = close_notify|#]
[#|2009-09-19T17:02:36.194+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=43;_ThreadName=httpSSLWorkerThre
ad-28181-0;|
httpSSLWorkerThread-28181-0, WRITE: TLSv1 Alert, length = 18|#]
```

The Client and Server communicated over a secure channel using SSL with Mutual Authentication. The server provided its X.509 Certificate to the client and requested the client to provide its X.509 Certificate back. Server and client validated each other’s certificates.

Let’s remove the client’s certificate from the server’s cacerts.jks truststore and submit the request. The SSL Handshake fails. The client receives a Fault, shown in Figure 5.12.1.



http://localhost:28080/TriggerConService/EJBPersonSSLMutualAuthCli

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <S:Fault xmlns:ns4="http://www.w3.org/2003/05/soap-envelope">
         <faultcode>S:Server</faultcode>
         <faultstring>java.net.SocketException: Unexpected end of file from server</faultstring>
         <detail>
            <PersonFlt:PersonFlt xmlns:PersonFlt="http://xml.netbeans.org/schema/Person" xmlns="htt
               <PersonID>67876</PersonID>
               <FaultDetail>java.net.SocketException: Unexpected end of file from server</FaultDeta
            </PersonFlt:PersonFlt>
         </detail>
      </S:Fault>
   </S:Body>
</S:Envelope>
```

Figure 5.12.1 *SOAP Fault on SSL Handshake failure – not particularly enlightening*

Listing 5.12.4 shows the fragment of the server.log at the client side – as far as the client is concerned the server simply dropped the connection – this is deliberately not very enlightening to the client since any indication of what might have gone wrong might help a potential attacker.

Listing 5.12.4 Client-side failed handshake log

```
[#|2009-09-19T23:48:57.890+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|... no IV used for this cipher|#]
[#|2009-09-19T23:48:57.890+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|
httpSSLWorkerThread-28080-2, WRITE: TLSv1 Change Cipher Spec, length = 17|#]
[#|2009-09-19T23:48:57.890+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|
*** Finished|#]
[#|2009-09-19T23:48:57.890+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|
verify_data:  { |#]
…
[#|2009-09-19T23:48:57.890+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|
httpSSLWorkerThread-28080-2, WRITE: TLSv1 Handshake, length = 32|#]
 [#|2009-09-19T23:48:58.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|
httpSSLWorkerThread-28080-2, received EOFException: ignored|#]
 [#|2009-09-19T23:48:58.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|
httpSSLWorkerThread-28080-2, called closeInternal(false)|#]
```

```
[#|2009-09-19T23:48:58.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|
httpSSLWorkerThread-28080-2|#]
[#|2009-09-19T23:48:58.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|, SEND TLSv1 ALERT:  |#]
[#|2009-09-19T23:48:58.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|warning, |#]
[#|2009-09-19T23:48:58.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|description = close_notify|#]
[#|2009-09-19T23:48:58.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|
httpSSLWorkerThread-28080-2, WRITE: TLSv1 Alert, length = 18|#]
[#|2009-09-19T23:48:58.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|
httpSSLWorkerThread-28080-2, called close()|#]
[#|2009-09-19T23:48:58.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|
httpSSLWorkerThread-28080-2, called closeInternal(true)|#]
[#|2009-09-19T23:48:58.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|
httpSSLWorkerThread-28080-2, called close()|#]
[#|2009-09-19T23:48:58.171+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;|
httpSSLWorkerThread-28080-2, called closeInternal(true)|#]
[#|2009-09-19T23:48:58.171+1000|WARNING|sun-
appserver2.1|javax.enterprise.system.stream.err|_ThreadID=73;_ThreadName=httpSSLWorkerThre
ad-28080-2;_RequestID=062f4dfb-54ab-4465-bec9-2d56a3a58bfd;|
javax.xml.ws.WebServiceException: java.net.SocketException: Unexpected end of file from
server
        at
com.sun.xml.ws.transport.http.client.HttpClientTransport.checkResponseCode(HttpClientTrans
port.java:238)
        at
com.sun.xml.ws.transport.http.client.HttpTransportPipe.process(HttpTransportPipe.java:151)
        at
com.sun.xml.ws.transport.http.client.HttpTransportPipe.processRequest(HttpTransportPipe.ja
va:88)
        at com.sun.xml.ws.api.pipe.Fiber.__doRun(Fiber.java:595)
        at com.sun.xml.ws.api.pipe.Fiber._doRun(Fiber.java:554)
        at com.sun.xml.ws.api.pipe.Fiber.doRun(Fiber.java:539)
```

Listing 5.12.5 shows the SSL Handshake failure at the server side. The handshake proceeds as normal until the server requests client's certificate. It does not get one. It tries to validate the certificate and fails. On failure it abort SSL Handshake.

Listing 5.12.5 Handshake failure at the server side

```
…

[#|2009-09-19T23:48:58.008+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
Session ID:  |#]
```

```
[#|2009-09-19T23:48:58.009+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|{74, 180, 225, 74, 185, 10, 114, 127, 188, 253, 132, 18, 240, 40, 62, 214,
109, 234, 13, 198, 51, 207, 105, 149, 16, 97, 123, 117, 240, 157, 161, 240}|#]
[#|2009-09-19T23:48:58.009+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
Cipher Suite: SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-19T23:48:58.009+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
Compression Method: 0|#]
[#|2009-09-19T23:48:58.009+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
***|#]
[#|2009-09-19T23:48:58.009+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
Cipher suite:  SSL_RSA_WITH_RC4_128_MD5|#]
[#|2009-09-19T23:48:58.009+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
*** Certificate chain|#]
[#|2009-09-19T23:48:58.011+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
chain [0] = [
[
  Version: V3
  Subject: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus:
102121541157065069575568250270437214984350336338157998094768569148500536018031894819145491
113311079336949399472032140208618356816163637202618727298710465300230334324506184949208474
446933178623253726110968175800084405517043740120715529443624373833850530287673442674333181
408072531272948503339758724948550982873
  public exponent: 65537
  Validity: [From: Sat Sep 05 13:48:28 EST 2009,
               To: Tue Sep 03 13:48:28 EST 2019]
  Issuer: CN=orad1.ssc, OU=Sun GlassFish Enterprise Server, O=Sun Microsystems, L=Santa
Clara, ST=California, C=US
  SerialNumber: [    4aa1df8c]

Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 9E 79 9C E9 59 86 34 8F   FD 75 09 F7 82 D0 82 CE  .y..Y.4..u......
0010: BE 9A 44 EE                                        ..D.
]
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 23 A7 FD 51 1F 81 9E 8C   34 3A 58 01 EF 5A 04 CD  #..Q....4:X..Z..
0010: AD 35 2C 67 17 40 3A B8   EA 19 37 DB B2 B3 C8 EA  .5,g.@:...7.....
0020: 5B 4F 0E 30 4E 9D 42 23   52 FE E8 53 44 8B 64 21  [O.0N.B#R..SD.d!
0030: CF 5F EE 07 D5 60 1E F2   1B EA 68 99 E4 BB 6C 89  ._...`....h...l.
0040: 02 21 1D A5 AE 6C 26 14   8C 92 02 92 E3 C1 74 56  .!...l&.......tV
0050: 6A 69 96 8E 2D 1E 7D 6C   52 5E 99 38 20 8B 19 C4  ji..-..lR^.8 ...
0060: 52 11 89 B3 73 D0 6C 61   B2 DB BF CA 58 0A 3A 5D  R...s.la....X.:]
0070: 40 81 97 CC 3F 60 A6 1E   B5 D6 60 8A C6 6B B6 F6  @...?`....`..k..

]|#]
[#|2009-09-19T23:48:58.011+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
***|#]
[#|2009-09-19T23:48:58.011+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
*** CertificateRequest|#]
[#|2009-09-19T23:48:58.011+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
Cert Types: |#]
[#|2009-09-19T23:48:58.012+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|RSA|#]
[#|2009-09-19T23:48:58.012+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|, |#]
[#|2009-09-19T23:48:58.012+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|DSS|#]
[#|2009-09-19T23:48:58.012+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
Cert Authorities:|#]
[#|2009-09-19T23:48:58.047+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
<OU=Equifax Secure Certificate Authority, O=Equifax, C=US>|#]
...
[#|2009-09-19T23:48:58.102+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
<EMAILADDRESS=info@valicert.com, CN=http://www.valicert.com/, OU=ValiCert Class 2 Policy
Validation Authority, O="ValiCert, Inc.", L=ValiCert Validation Network>|#]
[#|2009-09-19T23:48:58.102+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
*** ServerHelloDone|#]
[#|2009-09-19T23:48:58.106+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
httpSSLWorkerThread-28181-1, WRITE: TLSv1 Handshake, length = 5849|#]
[#|2009-09-19T23:48:58.298+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
httpSSLWorkerThread-28181-1, READ: TLSv1 Handshake, length = 157|#]
[#|2009-09-19T23:48:58.309+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
*** Certificate chain|#]
[#|2009-09-19T23:48:58.309+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
***|#]
[#|2009-09-19T23:48:58.309+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
httpSSLWorkerThread-28181-1, fatal error: 42: null cert chain
javax.net.ssl.SSLHandshakeException: null cert chain|#]
```

```
[#|2009-09-19T23:48:58.309+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
%% Invalidated:  [Session-3, SSL_RSA_WITH_RC4_128_MD5]|#]
[#|2009-09-19T23:48:58.309+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
httpSSLWorkerThread-28181-1|#]
[#|2009-09-19T23:48:58.309+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|, SEND TLSv1 ALERT:  |#]
[#|2009-09-19T23:48:58.309+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|fatal, |#]
[#|2009-09-19T23:48:58.309+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|description = bad_certificate|#]
[#|2009-09-19T23:48:58.309+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
httpSSLWorkerThread-28181-1, WRITE: TLSv1 Alert, length = 18|#]
[#|2009-09-19T23:48:58.310+1000|INFO|sun-
appserver2.1|javax.enterprise.system.stream.out|_ThreadID=27;_ThreadName=httpSSLWorkerThre
ad-28181-1;|
httpSSLWorkerThread-28181-1, fatal: engine already closed.  Rethrowing
javax.net.ssl.SSLHandshakeException: null cert chain|#]
[#|2009-09-19T23:48:58.310+1000|WARNING|sun-
appserver2.1|javax.enterprise.system.container.ejb|_ThreadID=27;_ThreadName=httpSSLWorkerT
hread-28181-1;_RequestID=a74d12d5-b6e4-4326-9fff-efd3253d61a8;|CLIENT CERT authentication
error for EJBPersonSSLMutualAuthSvc|#]
```

We created and exercised a pair of cooperating EJB-based web services. The client invoked the provider over a secure channel using SSL with Mutual Authentication. We saw the client and server SSL Handshake traces both when the handshake succeeded and when it failed.

Undeploy both the client and the service so they don’t get in the way as we explore further.

5.13 Snooping on SOAP Message Processing

By default very little feedback is provided to the developer of web services. As various policies are applied to services a great deal of processing is going on. The Metro project-provided infrastructure takes care of adding policy markup, processing policy markup, exception generation and the like. To better understand what is happening it will be good to configure the infrastructure to log messages and processing steps at various stages of processing.

To log WS-* processing for various policy variants, add directives listed in Listing 5.13.1 to the GlassFish Application Server’s “JVM Settings” -> “JVM Options”.

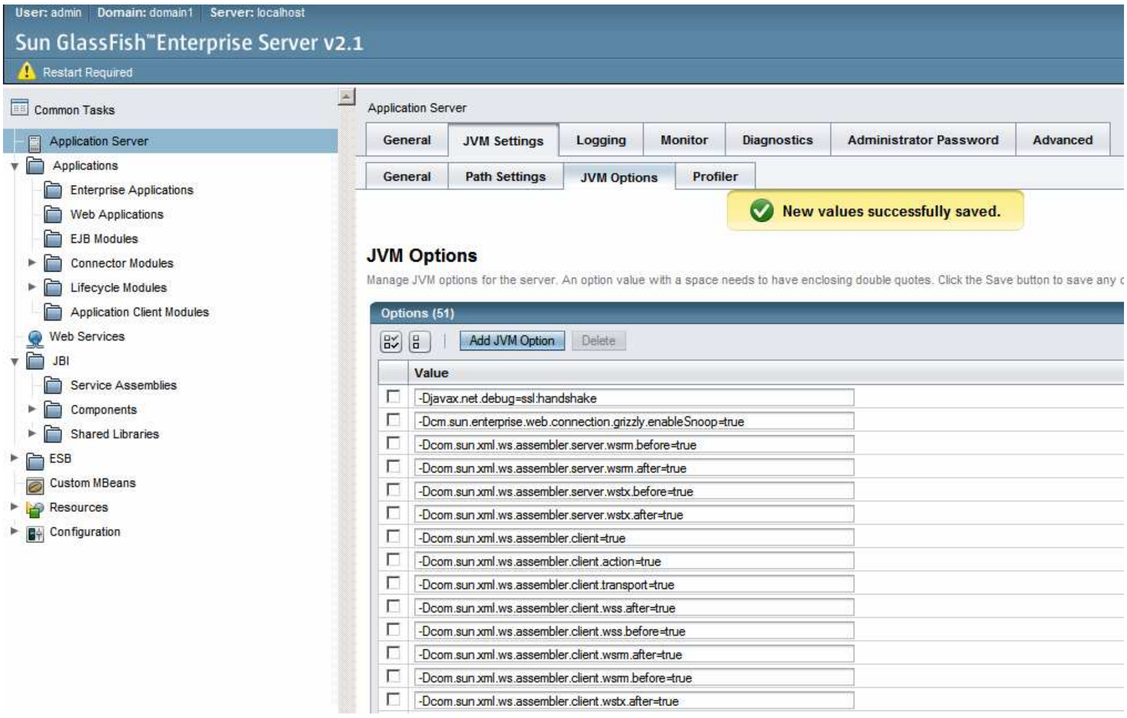
Listing 5.13.1 Metro processing and other logging directives

```
-Dcm.sun.enterprise.web.connection.grizzly.enableSnoop=true
-Dcom.sun.xml.ws.assembler.client.action=true
-Dcom.sun.xml.ws.assembler.client.transport=true
-Dcom.sun.xml.ws.assembler.client.wsa.after=true
-Dcom.sun.xml.ws.assembler.client.wsa.before=true
-Dcom.sun.xml.ws.assembler.client.wsrm.after=true
```

```
-Dcom.sun.xml.ws.assembler.client.wsrm.before=true
-Dcom.sun.xml.ws.assembler.client.wss.after=true
-Dcom.sun.xml.ws.assembler.client.wss.before=true
-Dcom.sun.xml.ws.assembler.client.wstx.after=true
-Dcom.sun.xml.ws.assembler.client.wstx.before=true
-Dcom.sun.xml.ws.assembler.client=true
-Dcom.sun.xml.ws.assembler.server.action=true
-Dcom.sun.xml.ws.assembler.server.transport=true
-Dcom.sun.xml.ws.assembler.server.transport=true
-Dcom.sun.xml.ws.assembler.server.wsa.after=true
-Dcom.sun.xml.ws.assembler.server.wsa.before=true
-Dcom.sun.xml.ws.assembler.server.wsmex.after=true
-Dcom.sun.xml.ws.assembler.server.wsmex.before=true
-Dcom.sun.xml.ws.assembler.server.wsrm.after=true
-Dcom.sun.xml.ws.assembler.server.wsrm.before=true
-Dcom.sun.xml.ws.assembler.server.wss.after=true
-Dcom.sun.xml.ws.assembler.server.wss.before=true
-Dcom.sun.xml.ws.assembler.server.wstx.after=true
-Dcom.sun.xml.ws.assembler.server.wstx.before=true
-Dcom.sun.xml.ws.assembler.server=true
-Dcom.sun.xml.ws.transport.http.HttpAdapter.dump=true
-Dcom.sun.xml.ws.transport.http.client.HttpTransportPipe.dump=true
-Dcom.sun.xml.ws.transport.local.LocalTransportPipe.dump=true
```

Here the “was” will log processing related to addressing, “wstx” will log processing related to transactionality, “wss” will log processing related to security, “wsrm” will log processing related to reliable messaging, “wsmex” will log processing related to message exchange and transport will log the wire messages. “server” and “client” have obvious meanings. Other directives log wire messages at different places in the processing stack.

Enabling all these logging facilities requires application server restart and will result in multiple log entries for the same message exchange. Figure 5.13.2 illustrates the GlassFish Application Server Console Restart warning.



Restart Required

New values successfully saved.

JVM Options

Manage JVM options for the server. An option value with a space needs to have enclosing double quotes. Click the Save button to save any c

Options (51)

	Value
□	-Djavax.net.debug=ssl:handshake
□	-Dcm.sun.enterprise.web.connection.grizzly.enableSnoop=true
□	-Dcom.sun.xml.ws.assembler.server.wsrn.before=true
□	-Dcom.sun.xml.ws.assembler.server.wsrn.after=true
□	-Dcom.sun.xml.ws.assembler.server.wstx.before=true
□	-Dcom.sun.xml.ws.assembler.server.wstx.after=true
□	-Dcom.sun.xml.ws.assembler.client=true
□	-Dcom.sun.xml.ws.assembler.client.action=true
□	-Dcom.sun.xml.ws.assembler.client.transport=true
□	-Dcom.sun.xml.ws.assembler.client.wss.after=true
□	-Dcom.sun.xml.ws.assembler.client.wss.before=true
□	-Dcom.sun.xml.ws.assembler.client.wsrn.after=true
□	-Dcom.sun.xml.ws.assembler.client.wsrn.before=true
□	-Dcom.sun.xml.ws.assembler.client.wstx.after=true

Figure 5.13.2 *Restart Required*

To log other events of interest consider adding the logging categories in Table 5.13.1.

Table 5.13.1 *Logging other processing*

Logging Category	Level
com.sun.org.apache.xml	FINEST
com.sun.xml	FINEST
com.sun.xml.bind.v2.runtime.reflect	INFO
com.sun.xml.bind.v2.runtime.reflect.opt	INFO
com.sun.xml.bind.v2.runtime.reflect.opt.Injector	INFO
com.sun.xml.messaging	FINER
com.sun.xml.rpc	FINEST
com.sun.xml.ws	FINEST
com.sun.xml.ws.api.pipe.Fiber	INFO
com.sun.xml.ws.assembler	FINEST
com.sun.xml.wss	FINEST
com.sun.xml.wss.impl	FINEST
com.sun.xml.wss.logging	FINEST
com.sun.xml.wss.logging.impl.opt	FINEST
com.sun.xml.wss.logging.impl.opt.crypto.level	FINEST
com.sun.xml.wss.logging.impl.opt.level	FINEST
com.sun.xml.wss.logging.impl.opt.signature.level	FINEST
com.sun.xml.wss.logging.impl.opt.token.level	FINEST
java.security.cert	FINEST

javax.enterprise.resource.webservices.jaxws	FINEST
javax.xml.messaging	FINEST
javax.xml.ws	FINEST
org.apache.xerces.dom	FINEST
org.jcp.xml	FINEST
org.jcp.xml.dsig.internal.dom.level	FINEST
sun.security.provider	FINEST
sun.security.provider.certpath	FINEST
sun.security.validator	FINEST

Some of these may be redundant or not triggered for specific scenarios.

Adding these logging categories and changing their values does not require restart of the Application Server.

5.14 Adding WS-Addressing to HTTP BC-based service

WS-Addressing "… provides transport-neutral mechanisms to address Web Services and messages".

For reference, WS-Addressing Specifications, applicable to the Metro implementation, are:

1. Web Services Addressing 1.0 - Core, W3C Recommendation 9 May 2006, Available: http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/, Accessed: September 2009

2. Web Services Addressing 1.0 - SOAP Binding, W3C Recommendation 9 May 2006, Available: http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/, Accessed: September 2009

3. Web Services Addressing 1.0 - Metadata, W3C Recommendation 4 September 2007, Available: http://www.w3.org/TR/2007/REC-ws-addr-metadata-20070904/, Accessed: September 2009

In this section only SOAP 1.1 Addressing Extensions are discussed and used in examples. See referenced specifications [2] for SOAP 1.2-related discussion.

In this section only WSDL 1.1 is used. See referenced specifications [2] for SOAP 1.2-related discussion.

Note that using non-anonymous addressing implies asynchronous messaging, in which response to a request, if any, is sent to a distinct endpoint and explicitly correlated with the request, instead of being returned in a HTTP response.

The Metro stack takes care of all aspects of the WS-Addressing work. The Metro Configuration Wizard, built into the HTTP BC, automatically configures WS-Addressing when certain security policies are chosen. It does so in a manner which makes the Addressing optional or ignored. For this reason WS-Addressing-related WS Policy must be modified if WS-Addressing is to be mandatory.

5.14.1 Enabling WS-Addressing

Create the PersonSvc_CA_WSAddressing Composite Application, drag the PersonSvc BPEL Module onto the CASA canvas, add and connect a SOAP Binding and Build the project.

The original WSDL does not use any security policies at all. It can not, because most security policies are applied to the concrete part of he WSDL and our original WSDL does not have a concrete part. By dragging the SOAP BC onto the CASAS canvas and connecting it to the BPEL Module we created a WSDL which imports our original WSDL and adds the concrete part.

Click the “Paper-and-key” icon and choose “Server Configuration”, as illustrated in Figure 5.14.1.

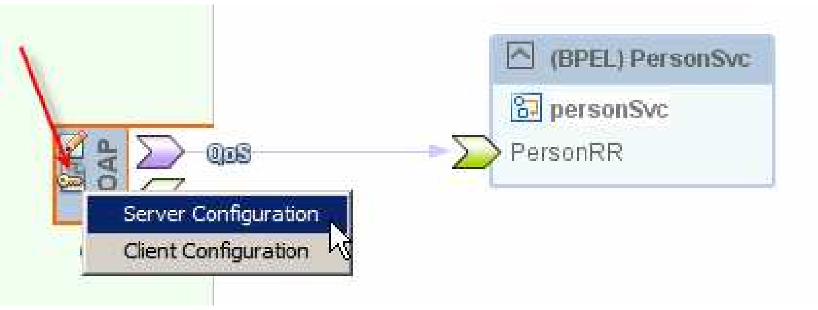
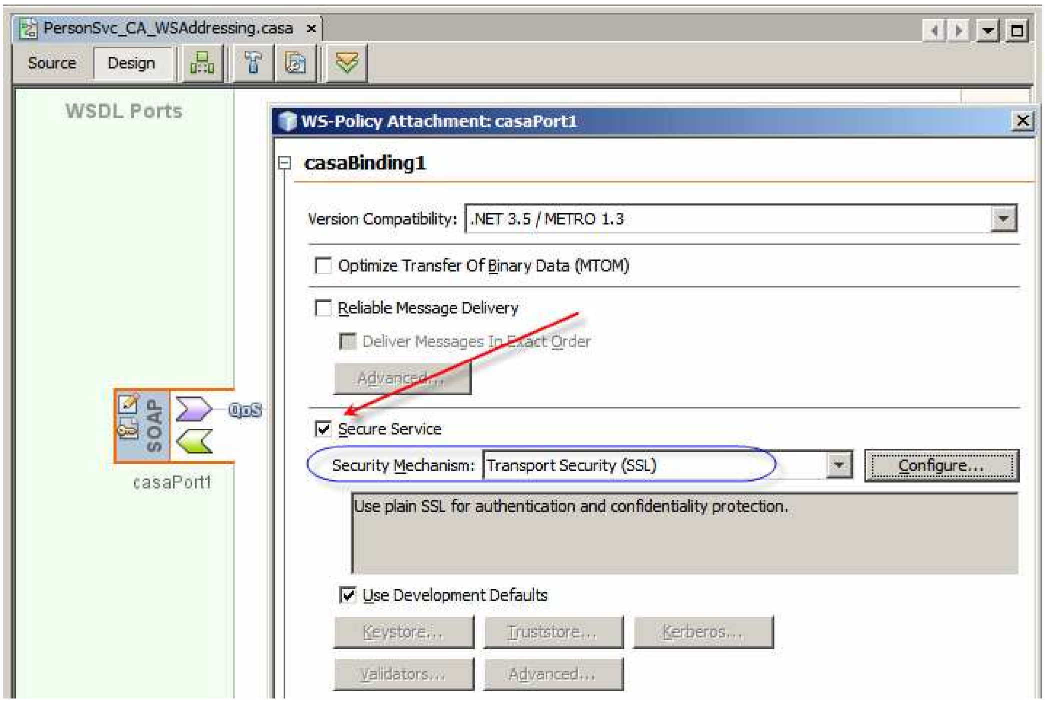


Figure 5.14.1 *Start the Server Configuration Wizard*

Check the “Secure Service” checkbox and choose “Transport Security (SSL)” from the “Security Mechanism” drop-down. Figure 5.14.2 shows the wizard at this point.


PersonSvc_CA_WSAddressing.casa
Source
Design
WSDL Ports
SOAP
QoS
casaPort1
WS-Policy Attachment: casaPort1

casaBinding1

Version Compatibility: .NET 3.5 / METRO 1.3

□ Optimize Transfer Of Binary Data (MTOM)

□ Reliable Message Delivery

□ Deliver Messages In Exact Order

Advanced...

☑ Secure Service

Security Mechanism: Transport Security (SSL) Configure...

Use plain SSL for authentication and confidentiality protection.

☑ Use Development Defaults

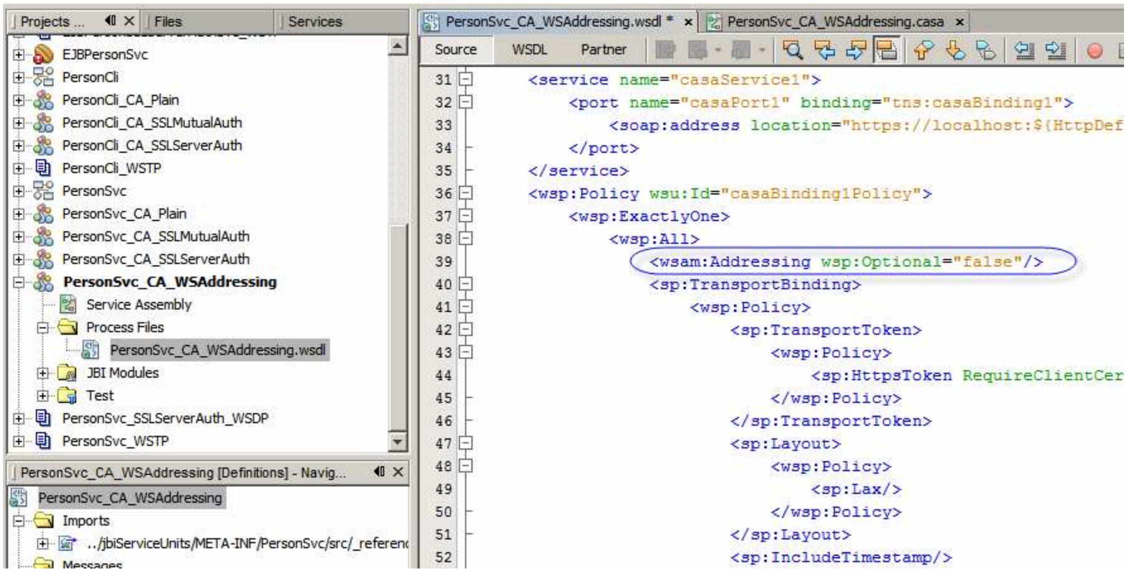
Keystore... Truststore... Kerberos...

Validators... Advanced...

Figure 5.14.2 *Create Transport Security Policy*

We chose the “Transport Security (SSL)” because the wizard does not allow us to choose individual policy formulations, like WS-Addressing policy. Instead, it offers a series of pre-configured policy combinations, which somebody or another decided were good, most useful, or whatever criteria they used to decide what to offer. The “Transport Security (SSL)” policy template adds minimum extraneous policies in addition to WS-Addressing policy.

Open the Concrete WSDL in the editor and look at the Policy stanza, paying particular attention to the WS-Addressing policy, highlighted in Figure 5.14.3.

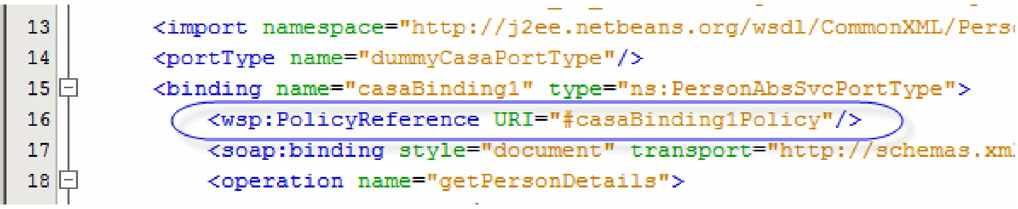

Projects ...
Files
Services
EJBPersonSvc
PersonCli
PersonCli_CA_Plain
PersonCli_CA_SSLMutualAuth
PersonCli_CA_SSLServerAuth
PersonCli_WSTP
PersonSvc
PersonSvc_CA_Plain
PersonSvc_CA_SSLMutualAuth
PersonSvc_CA_SSLServerAuth
PersonSvc_CA_WSAddressing
Service Assembly
Process Files
PersonSvc_CA_WSAddressing.wsdl
JBI Modules
Test
PersonSvc_SSLServerAuth_WSDP
PersonSvc_WSTP
PersonSvc_CA_WSAddressing [Definitions] - Navig...
PersonSvc_CA_WSAddressing
Imports
../jbiServiceUnits/META-INF/PersonSvc/src/_referenc
Messages
PersonSvc_CA_WSAddressing.wsdl *
PersonSvc_CA_WSAddressing.casa
Source
WSDL
Partner

```
31      <service name="casaService1">
32          <port name="casaPort1" binding="tns:casaBinding1">
33              <soap:address location="https://localhost:${HttpDef
34          </port>
35      </service>
36      <wsp:Policy wsu:Id="casaBinding1Policy">
37          <wsp:ExactlyOne>
38              <wsp:All>
39                  <wsam:Addressing wsp:Optional="false"/>
40                  <sp:TransportBinding>
41                      <wsp:Policy>
42                          <sp:TransportToken>
43                              <wsp:Policy>
44                                  <sp:HttpsToken RequireClientCer
45                              </wsp:Policy>
46                          </sp:TransportToken>
47                          <sp:Layout>
48                              <wsp:Policy>
49                                  <sp:Lax/>
50                              </wsp:Policy>
51                          </sp:Layout>
52                          <sp:IncludeTimestamp/>
```

Figure 5.14.3 *WS-Addressing Policy*

Inspect more of the WSDL before going on to modify it.

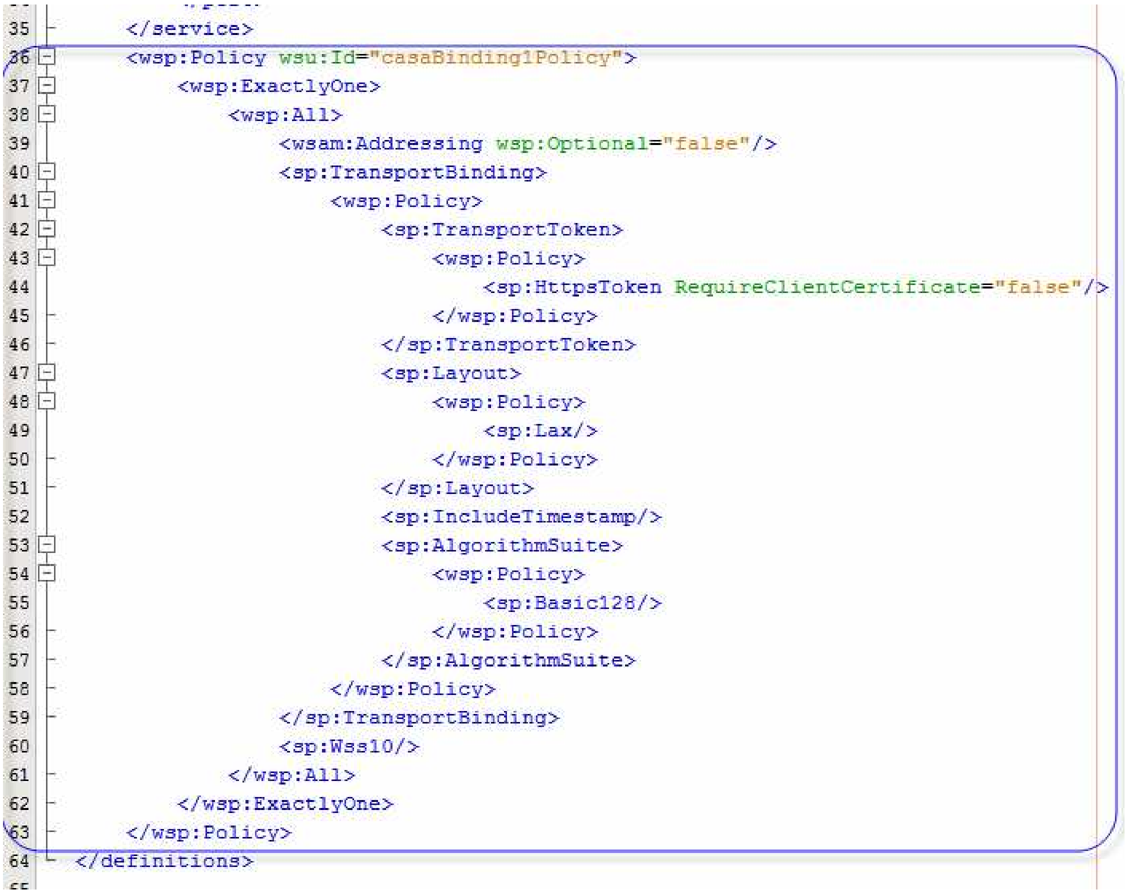
Note the PolicyReference Tag, at Line 16 in my WSDL, shown in Figure 5.14.4. It names the policy which is to be applied to this binding.



```
13      <import namespace="http://j2ee.netbeans.org/wsdl/CommonXML/Pers
14      <portType name="dummyCasaPortType"/>
15      <binding name="casaBinding1" type="ns:PersonAbsSvcPortType">
16          <wsp:PolicyReference URI="#casaBinding1Policy"/>
17          <soap:binding style="document" transport="http://schemas.xml
18          <operation name="getPersonDetails">
```

Figure 5.14.4 *Policy Reference applied to the binding*

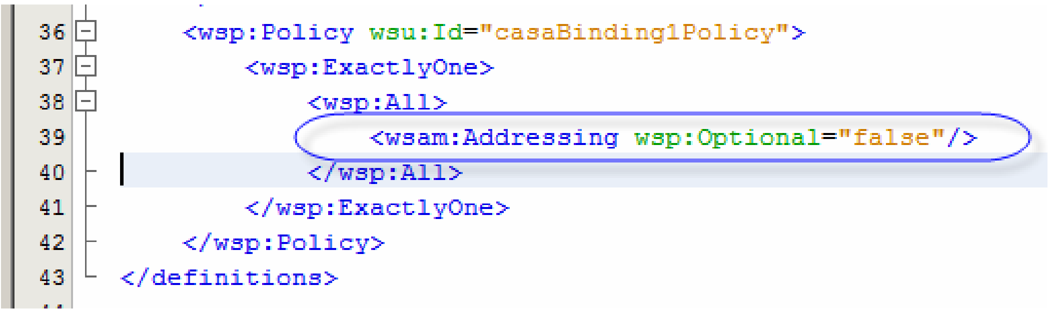
This reference names the wsp:Policy structure whose wsu:Id attribute value is the literal following the # sign, shown on lines 36-63 in my WSDL, Figure 5.14.5.



```
    </service>
    <wsp:Policy wsu:Id="casaBinding1Policy">
        <wsp:ExactlyOne>
            <wsp:All>
                <wsam:Addressing wsp:Optional="false"/>
                <sp:TransportBinding>
                    <wsp:Policy>
                        <sp:TransportToken>
                            <wsp:Policy>
                                <sp:HttpsToken RequireClientCertificate="false"/>
                            </wsp:Policy>
                        </sp:TransportToken>
                        <sp:Layout>
                            <wsp:Policy>
                                <sp:Lax/>
                            </wsp:Policy>
                        </sp:Layout>
                        <sp:IncludeTimestamp/>
                        <sp:AlgorithmSuite>
                            <wsp:Policy>
                                <sp:Basic128/>
                            </wsp:Policy>
                        </sp:AlgorithmSuite>
                    </wsp:Policy>
                </sp:TransportBinding>
                <sp:Wss10/>
            </wsp:All>
        </wsp:ExactlyOne>
    </wsp:Policy>
</definitions>
```

Figure 5.14.5 *Policy structure*

Delete lines starting with "<sp:TransportBinding" and ending with "<sp:Wss10/>", both inclusive, to obtain a policy formulation with jus the WS-Addressing policy, as shown in Figure 5.14.6.



```
    <wsp:Policy wsu:Id="casaBinding1Policy">
        <wsp:ExactlyOne>
            <wsp:All>
                <wsam:Addressing wsp:Optional="false"/>
            </wsp:All>
        </wsp:ExactlyOne>
    </wsp:Policy>
</definitions>
```

Figure 5.14.6 *WS-Addressing policy*

Modify the URL in the soap:address tag to use the "http", rather then the "https" scheme / protocol. The Metro wizard is "clever" in that it changes http to https when we select the "Transport Security (SSL)" but is not clever enough to change ${HttpDefaultPort} to ${HttpsDefaultPort}. Since we modified the policy such that it no longer uses transport security the scheme has to be manually set to http.

Build and Deploy the Composite Application.

The service is built and deployed.

Create a Client Composite Application, PersonCli_CA_WSAddressing, much as we have done previously, using the BPEL Module PersonCli. Build the project.

In the new project's Process Files folder create New -> External WSDL Document(s), using the service WSDL URL. For me this will be:

```
http://localhost:29080/casaService1/casaPort1?WSDL
```

Right-click in the WSDL Ports and "Load WSDL Port".

Connect the new SOAP BC "Provides" connector to the BPEL Module "Consumes" connector. Click the "paper and key" icon and choose "Configure Client". This is shown in Figure 5.14.7.

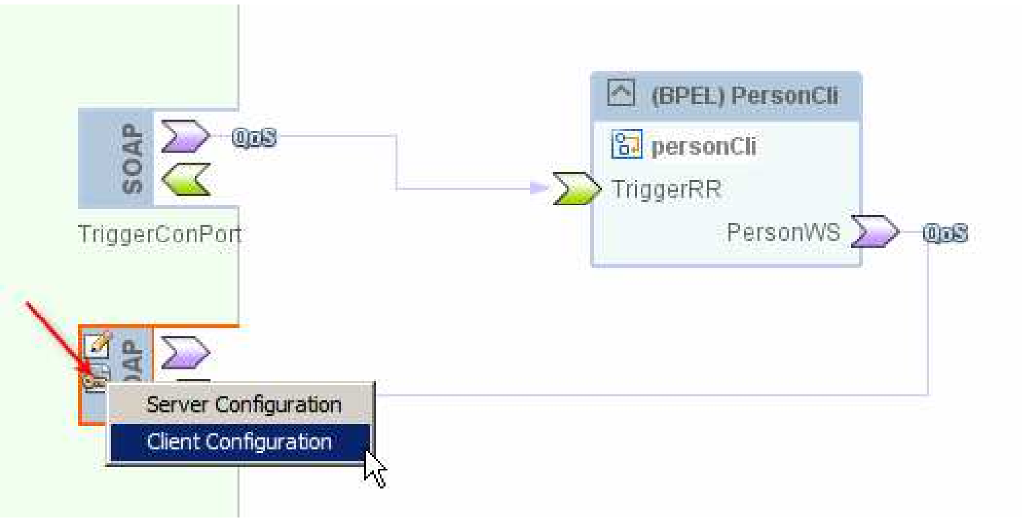
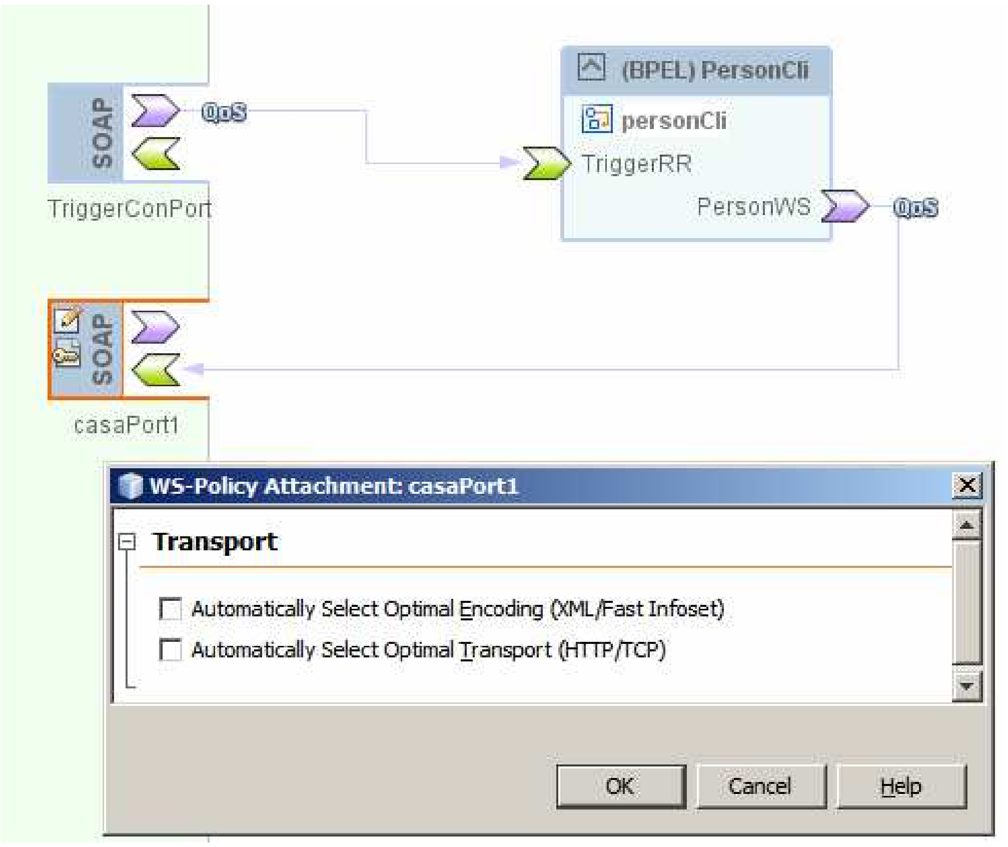


Figure 5.14.7 *Configure Client WSDL*

Notice, as shown in Figure 5.14.8, that there is nothing to configure for the WS-Addressing policy.



WS-Policy Attachment: casaPort1

Transport

□ Automatically Select Optimal Encoding (XML/Fast Infoset)

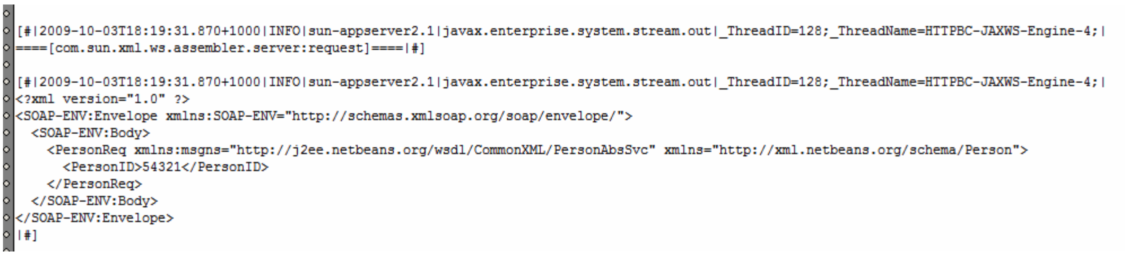
□ Automatically Select Optimal Transport (HTTP/TCP)

Figure 5.14.8 *No WS-Addressing policy objects in the Client wizard*

Build and Deploy this project.

Use the PersonCli_WSTP Web Service Testing Projet to exercise the solution.

In the serever.log look for the line with the literal “---[HTTP Request]---“. After this line look for a trace that looks similar to that shown in Figure 5.14.9.

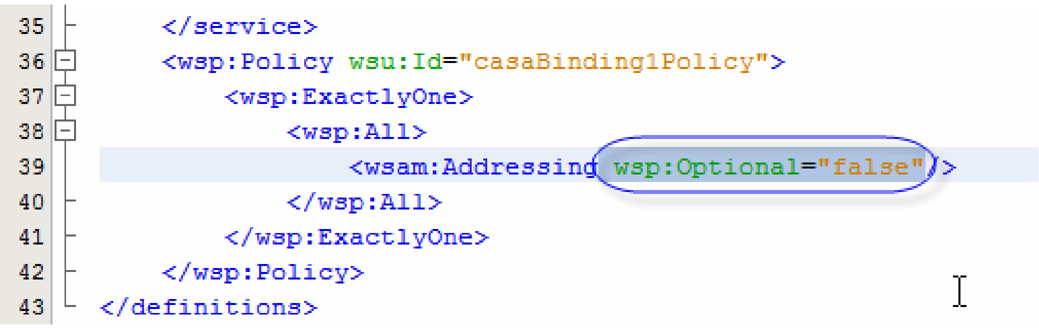


```
[#|2009-10-03T18:19:31.870+1000|INFO|sun-appserver2.1|javax.enterprise.system.stream.out|_ThreadID=128;_ThreadName=HTTPBC-JAXWS-Engine-4;|
====[com.sun.xml.ws.assembler.server:request]====|#]

[#|2009-10-03T18:19:31.870+1000|INFO|sun-appserver2.1|javax.enterprise.system.stream.out|_ThreadID=128;_ThreadName=HTTPBC-JAXWS-Engine-4;|
<?xml version="1.0" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <PersonReq xmlns:msgns="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc" xmlns="http://xml.netbeans.org/schema/Person">
      <PersonID>54321</PersonID>
    </PersonReq>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
|#]
```

Figure 5.14.9 *Undecorated request*

Notice that the SOAP Request has no markup that has anything to do with WS-Addressing. The policy is ineffective, it would seem. Confirm this by looking at the XML produced by the -Dcom.sun.xml.ws.assembler.server.wsa.before directive, and elsewhere in the log.

Open the PersonSvc_CA_WSAddressing.wsdl, in the PersonSvc_CA_WSAddressing Composite Application project and inspect line 39, shown in Figure 5.14.10.



```
    </service>
    <wsp:Policy wsu:Id="casaBinding1Policy">
        <wsp:ExactlyOne>
            <wsp:All>
                <wsam:Addressing wsp:Optional="false"/>
            </wsp:All>
        </wsp:ExactlyOne>
    </wsp:Policy>
</definitions>
```

Figure 5.14.10 *wsam:Addressing tag with wsp:Optional attribute*

From practical experience I know that the presence of this tag causes the infrastructure to ignore the Addressing policy regardless of the value of the attribute. Remove the attribute and its value, leaving just the "<wsam:Addressing/>" tag. Build and Deploy the service project.

Build and Deploy the client project, PersonCli_CA_WSAddressing.

Submit the request. Note the SOAP Fault. Inspect the log to see what the issue is. Observe:

"A required header representing a Message Addressing Property is not present"

The client is not decorating the SOAP Request with WS-Addressing markup where the serevice policy requires such decoration.

At present there is no graceful way to "refresh" the service WSDL in the client project. Delete all folders and file under the Process Files folder in the client project. Figure 5.14.11 illustrates the project hierarchy and points out the files to be deleted.

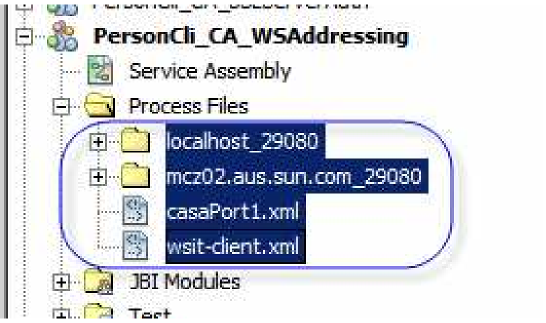


Figure 5.14.11 *Delete content of the Project Files folder*

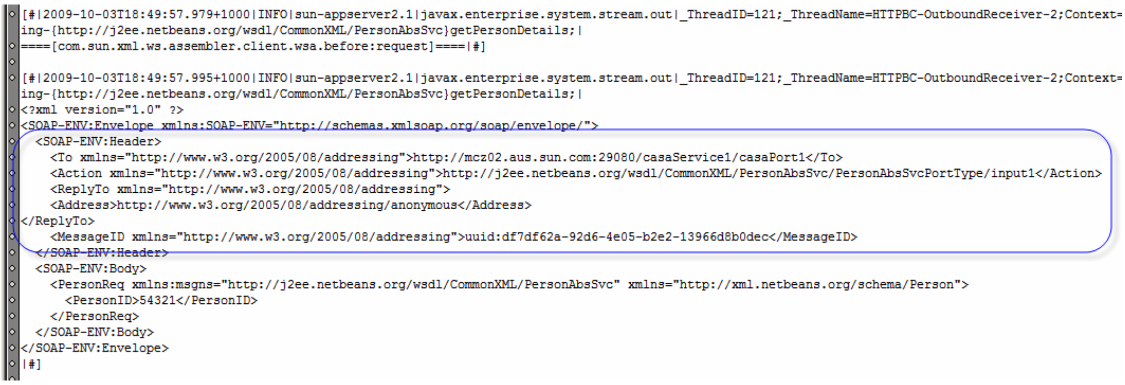
Create the External WSDL Document, using service WSDL URL, all over again. Create Client Configuration in the CASA map all over again.

Build and Deploy the project.

Submit the request.

Notice that the response comes as expected.

Inspect the server.log, looking for com.sun.xml.ws.assembler.client.wsa.before:request literal. The XML markup will be similar to that shown in Figure 5.14.12.

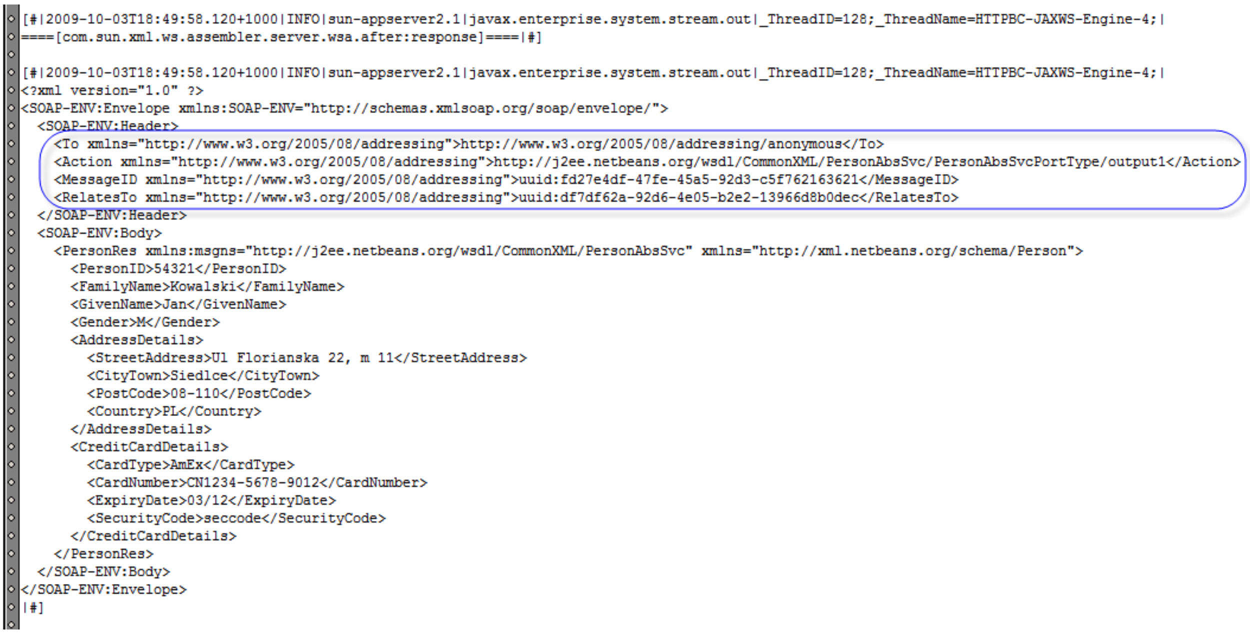


```
[#|2009-10-03T18:49:57.979+1000|INFO|sun-appserver2.1|javax.enterprise.system.stream.out|_ThreadID=121;_ThreadName=HTTPBC-OutboundReceiver-2;Context=
ing-{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
====[com.sun.xml.ws.assembler.client.wsa.before:request]====|#]

[#|2009-10-03T18:49:57.995+1000|INFO|sun-appserver2.1|javax.enterprise.system.stream.out|_ThreadID=121;_ThreadName=HTTPBC-OutboundReceiver-2;Context=
ing-{http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc}getPersonDetails;|
<?xml version="1.0" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <To xmlns="http://www.w3.org/2005/08/addressing">http://mcz02.aus.sun.com:29080/casaService1/casaPort1</To>
    <Action xmlns="http://www.w3.org/2005/08/addressing">http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc/PersonAbsSvcPortType/input1</Action>
    <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
    <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
</ReplyTo>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:df7df62a-92d6-4e05-b2e2-13966d8b0dec</MessageID>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <PersonReq xmlns:msgns="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc" xmlns="http://xml.netbeans.org/schema/Person">
      <PersonID>54321</PersonID>
    </PersonReq>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
|#]
```

Figure 5.14.12 *WS-Addressing headers added to the SOAP Request*

Consult WS-Addressing standards document for what these tags and their values mean.

Look, in server.log, for literal text “com.sun.xml.ws.assembler.server.wsa.before:response”. The XML markup shown will look similar to that in Figure 5.14.13, and will contain message ids of the original request and the response to it. These were, in this case, generated by the underlying web services infrastructure.



```
[#|2009-10-03T18:49:58.120+1000|INFO|sun-appserver2.1|javax.enterprise.system.stream.out|_ThreadID=128;_ThreadName=HTTPBC-JAXWS-Engine-4;|
====[com.sun.xml.ws.assembler.server.wsa.after:response]====|#]

[#|2009-10-03T18:49:58.120+1000|INFO|sun-appserver2.1|javax.enterprise.system.stream.out|_ThreadID=128;_ThreadName=HTTPBC-JAXWS-Engine-4;|
<?xml version="1.0" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <To xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing/anonymous</To>
    <Action xmlns="http://www.w3.org/2005/08/addressing">http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc/PersonAbsSvcPortType/output1</Action>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:fd27e4df-47fe-45a5-92d3-c5f762163621</MessageID>
    <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:df7df62a-92d6-4e05-b2e2-13966d8b0dec</RelatesTo>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <PersonRes xmlns:msgns="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc" xmlns="http://xml.netbeans.org/schema/Person">
      <PersonID>54321</PersonID>
      <FamilyName>Kowalski</FamilyName>
      <GivenName>Jan</GivenName>
      <Gender>M</Gender>
      <AddressDetails>
        <StreetAddress>Ul Florianska 22, m 11</StreetAddress>
        <CityTown>Siedlce</CityTown>
        <PostCode>08-110</PostCode>
        <Country>PL</Country>
      </AddressDetails>
      <CreditCardDetails>
        <CardType>AmEx</CardType>
        <CardNumber>CN1234-5678-9012</CardNumber>
        <ExpiryDate>03/12</ExpiryDate>
        <SecurityCode>seccode</SecurityCode>
      </CreditCardDetails>
    </PersonRes>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
|#]
```

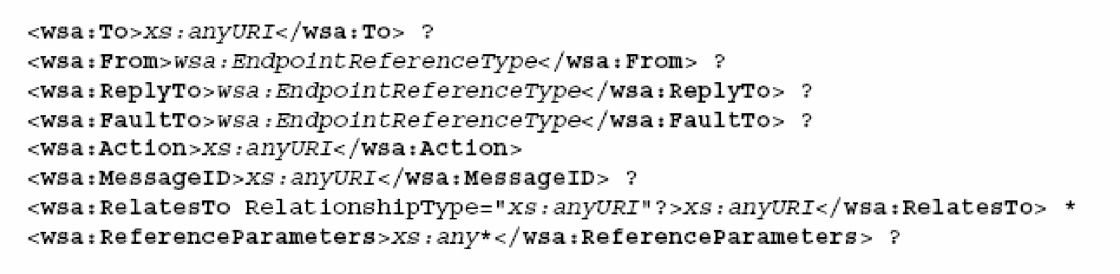
Figure 5.14.13 *Response with WS-Addressing markup*

5.14.2 Interacting with WS-Addressing Headers in BPEL

By itself, enabling WS-Addressing support is pretty useless. It is true that the infrastructure assigned message IDs to requests and responses, and that the response carries the message ID of the request but nothing is done with these values. To make WS-Addressing useful one must be able to use WS-Addressing headers in processing logic.

One uses NMProperties, the associated tooling and a bit of manual WSDL editing, to expose WS-Addressing headers in BPEL.

The WS-Addressing Core specification includes standard headers shown in Figure 5.14.14.

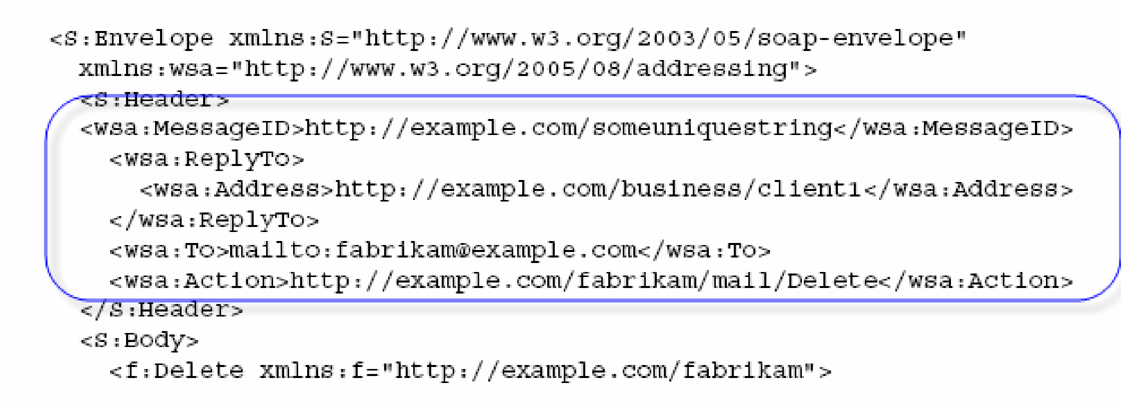


```
<wsa:To>xs:anyURI</wsa:To> ?
<wsa:From>wsa:EndpointReferenceType</wsa:From> ?
<wsa:ReplyTo>wsa:EndpointReferenceType</wsa:ReplyTo> ?
<wsa:FaultTo>wsa:EndpointReferenceType</wsa:FaultTo> ?
<wsa:Action>xs:anyURI</wsa:Action>
<wsa:MessageID>xs:anyURI</wsa:MessageID> ?
<wsa:RelatesTo RelationshipType="xs:anyURI"?>xs:anyURI</wsa:RelatesTo> *
<wsa:ReferenceParameters>xs:any*</wsa:ReferenceParameters> ?
```

Figure 5.14.14 *WS-Addressing standard headers*

Note the “?” denotes an optional header and “*” denotes an optional and repeating header.

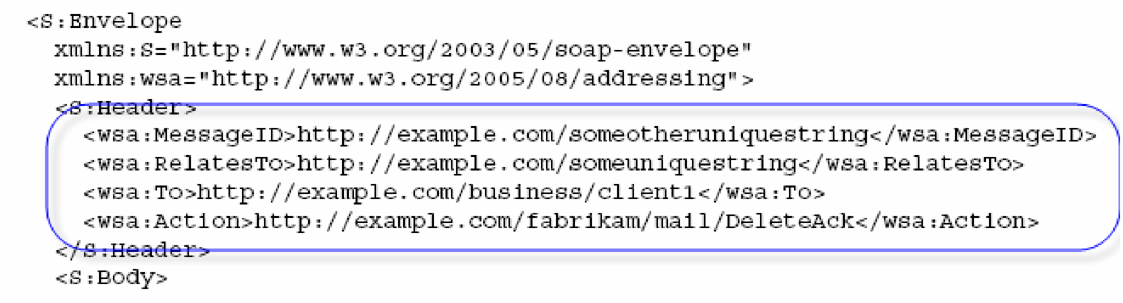
Figure 5.14.15, taken form the WS_Addressing Core specification, shows WS-Addressing headers in a sample request message.



```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
  <wsa:MessageID>http://example.com/someuniquestring</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://example.com/business/client1</wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>mailto:fabrikam@example.com</wsa:To>
    <wsa:Action>http://example.com/fabrikam/mail/Delete</wsa:Action>
  </S:Header>
  <S:Body>
    <f:Delete xmlns:f="http://example.com/fabrikam">
```

Figure 5.14.15 *Request WS-Addressing headers*

Figure 5.14.16, also taken from the WS-Addressing Core specification, shows the WS-Addressing headers in the response message.



```
<S:Envelope
  xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:MessageID>http://example.com/someotheruniquestring</wsa:MessageID>
    <wsa:RelatesTo>http://example.com/someuniquestring</wsa:RelatesTo>
    <wsa:To>http://example.com/business/client1</wsa:To>
    <wsa:Action>http://example.com/fabrikam/mail/DeleteAck</wsa:Action>
  </S:Header>
  <S:Body>
```

Figure 5.14.16 *Response WS-Addressing headers*

The value of the RelatesTo header is the response is the same as the value of the MessageID header in the request.

When using the HTTP BC for this the headers will look like these shown in Listings 5.14.1 and 5.14.2.

Listing 5.14.1 HTTP BC WS-Addressing headers in Request

```
<?xml version="1.0" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header>
      <To
xmlns="http://www.w3.org/2005/08/addressing">http://192.168.56.1:29080/casaService1/casaPo
rt1</To>
      <Action
xmlns="http://www.w3.org/2005/08/addressing">http://j2ee.netbeans.org/wsdl/CommonXML/Perso
nAbsSvc/PersonAbsSvcPortType/input1</Action>
      <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
         <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
      </ReplyTo>
      <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:a9293398-34be-419a-
9cde-b73810b524cd</MessageID>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <PersonReq xmlns:msgns="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc"
xmlns="http://xml.netbeans.org/schema/Person">
         <PersonID>54321</PersonID>
      </PersonReq>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 5.14.1 HTTP BC WS-Addressing headers in Response

```
<?xml version="1.0" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
      <Address>http://mcz.com.au/MyProperty</Address>
    </ReplyTo>
    <To
xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing/anonymou
s</To>
    <Action
xmlns="http://www.w3.org/2005/08/addressing">http://j2ee.netbeans.org/wsdl/CommonXML/Perso
nAbsSvc/PersonAbsSvcPortType/output1</Action>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:f9b19f55-46e7-470a-bd2e-
5ceb31269274</MessageID>
    <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:a9293398-34be-419a-9cde-
b73810b524cd</RelatesTo>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <PersonRes xmlns:msgns="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc"
xmlns="http://xml.netbeans.org/schema/Person">
      <PersonID>54321</PersonID>
      <FamilyName>Kowalski</FamilyName>
      <GivenName>Jan</GivenName>
      <Gender>M</Gender>
      <AddressDetails>
        <StreetAddress>Ul Florianska 22, m 11</StreetAddress>
        <CityTown>Siedlce</CityTown>
        <PostCode>08-110</PostCode>
        <Country>PL</Country>
      </AddressDetails>
      <CreditCardDetails>
        <CardType>AmEx</CardType>
        <CardNumber>CN1234-5678-9012</CardNumber>
        <ExpiryDate>03/12</ExpiryDate>
        <SecurityCode>seccode</SecurityCode>
      </CreditCardDetails>
    </PersonRes>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Compare the value of the MessageID in the request to the value of the RelatesTo in the response.

Note, also, the value of the ReplyTo header in the request and the To header in the response. Both are "…/anonymous". This is a synchronous request/response.

So far we have been reusing the same two BPEL Module projects, PersonSvc and PersonCli. This is because we did not need to make any changes to the business logic. Now that we would like to interact with the WS-Addressing headers we need to modify BPEL logic to get header values. Rather then modifying the original BPEL Modules we will clone the projects for this occasion and modify the cloned BPEL logic.

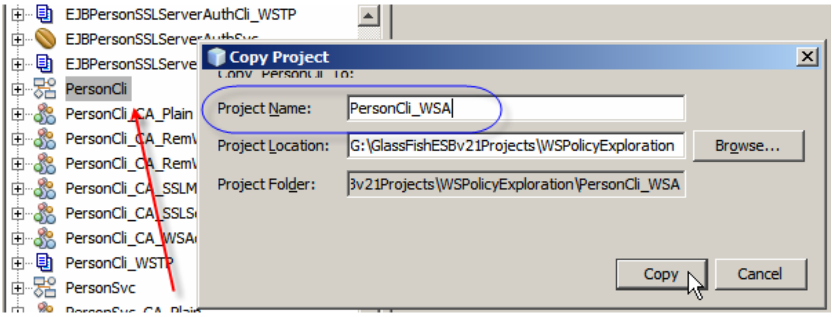
Since the WS-Addressing headers are not specified in the WSDL Interface Document, and are added by the Metro infrastructure, we can not access to them in BPEL through the request and response message structures. A document "Java CAPS 5 / 6, OpenESB, GlassFish ESB – Handling SOAP Headers in BPEL", available at http://blogs.sun.com/javacapsfieldtech/entry/java_caps_5_6_openesb, discusses how one could add SOAP Headers to a WSDL so that they are accessible in BPEL. To preserve the original WSDL we will not use this method. We will use the NMProperties (Normalized Message Properties) and the associated tooling. This technique can be used to access arbitrary SOAP Header values without having them declared in the WSDL.

The NMProperties technique requires us to define a NMProperty for each header we need to access. Tooling provides only partial support for what we need to do so certain amount of manual XML modification will be required. Both properties and logic to manipulate them will be added to the BPEL process, which will make it non-generic and dependent, in our case, on WS-Addressing policy being enforced for the service and the client.

The WS-addressing headers with which we wish to work are the "MessageID" in the Request, the "RelatesTo" in the Response.

Let's start by deploying the projects PersonSvc_CA_WSAddressing, if it is not deployed, and undeploying the project PersonCli_CA_WSAddressing, if it is deployed. We worked on these earlier in this section.

Let's clone the consumer BPEL Module project. Right-click on the name of the PersonCli project and choose Copy. Give the project the name of PersonCli_WSA. Figure 5.14.17 illustrates the copy dialogue box.



Copy Project

Project Name: PersonCli_WSA

Project Location: G:\GlassFishESBv21Projects\WSPolicyExploration

Project Folder: 3v21Projects\WSPolicyExploration\PersonCli_WSA

Figure 5.14.17 *Copy dialogue box*

Expand the new project. Open personCli.bpel business process in the editor, select the Assign1 activity and switch to the Mapper. Expand the vPersonRes structure, right-click on Properties node and choose "Add NMProperty…", as illustrated in Figure 5.14.18.

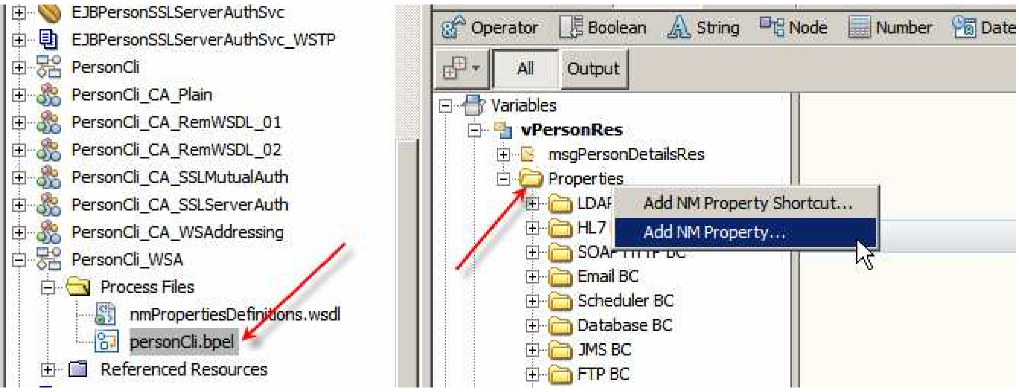
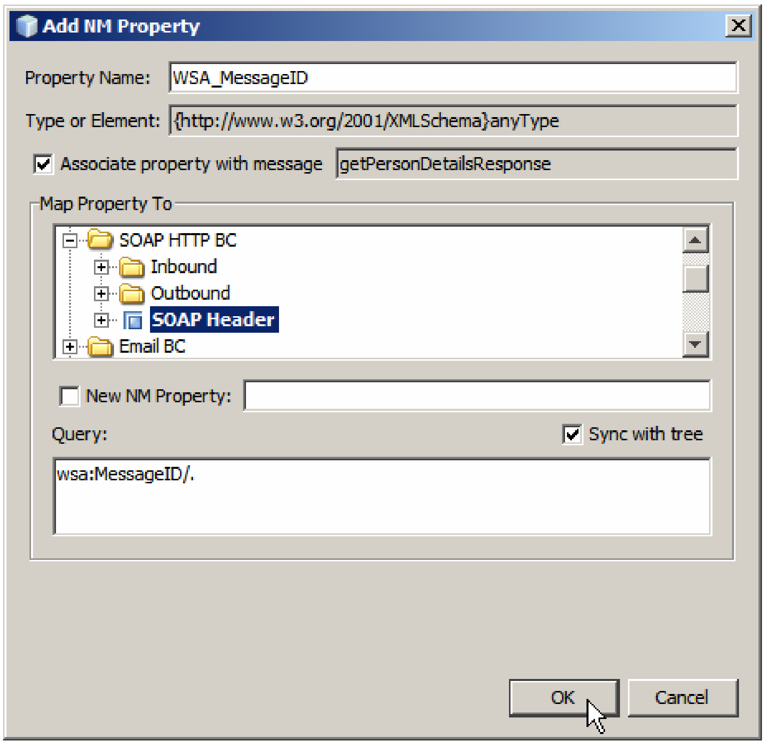


Figure 5.14.18 *Add NMProperty*

Provide the name for this property: WSA_MessageID, check the "Associate property with message" checkbox, expand "SOAP HTTP BC" and choose "SOAP Header" node then enter "wsa:MessageID/." as the Query value. Figure 5.14.19 illustrates these points.


Add NM Property

Property Name: WSA_MessageID

Type or Element: {http://www.w3.org/2001/XMLSchema}anyType

☑ Associate property with message getPersonDetailsResponse

Map Property To

□ New NM Property:

Query: ☑ Sync with tree

wsa:MessageID/.

Figure 5.14.19 *WSA_MessageID WS-Addressing Header NMProperty*

Click OK. Note a new WSDL document, nmPropertiesDefinitons.wsdl in the project tree, as illustrated in Figure 5.14.20.

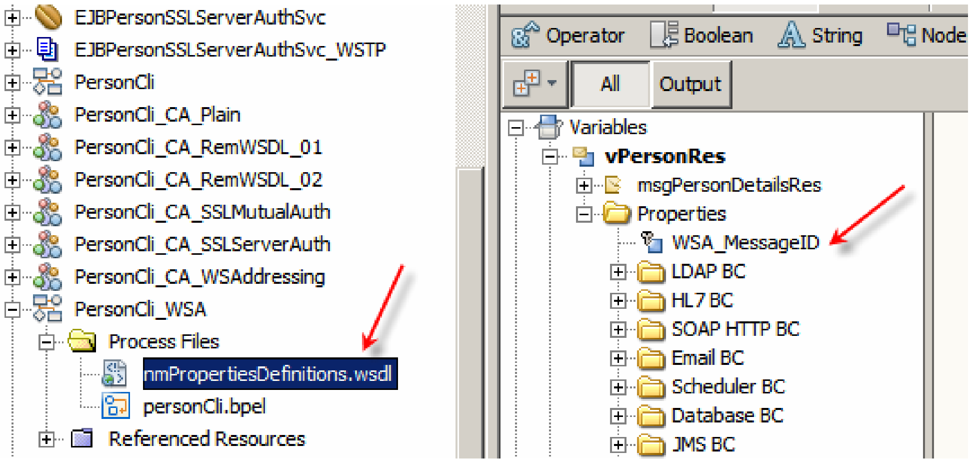
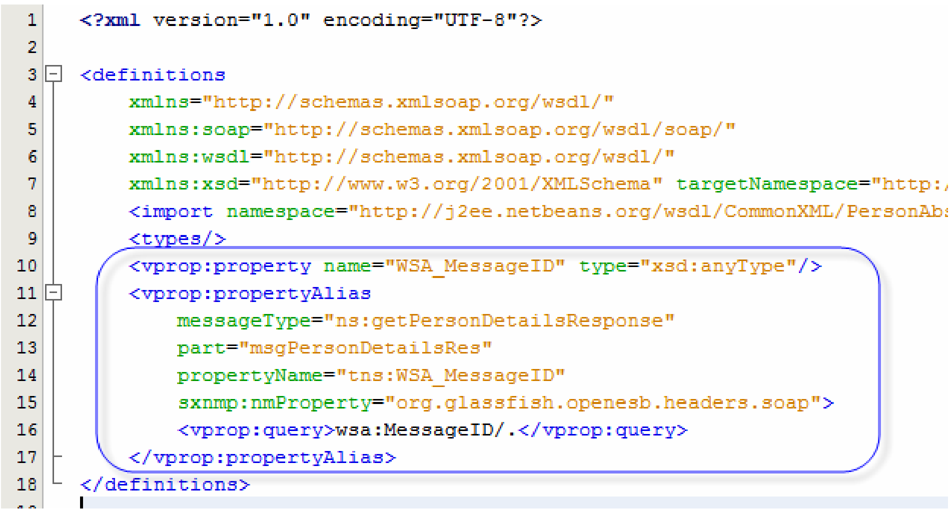


Figure 5.14.20 *New NMProperty and nmPropertiesDefinitions WSDL*

Open the nmPropertiesDefinitions WSDL and switch to Source mode. Figure 5.14.21 illustrates the WSDL which was generated for me, with long lines reformatted for better readability.



```
<?xml version="1.0" encoding="UTF-8"?>

<definitions
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http:/
    <import namespace="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAb
    <types/>
    <vprop:property name="WSA_MessageID" type="xsd:anyType"/>
    <vprop:propertyAlias
        messageType="ns:getPersonDetailsResponse"
        part="msgPersonDetailsRes"
        propertyName="tns:WSA_MessageID"
        sxnmp:nmProperty="org.glassfish.openesb.headers.soap">
        <vprop:query>wsa:MessageID/.</vprop:query>
    </vprop:propertyAlias>
</definitions>
```

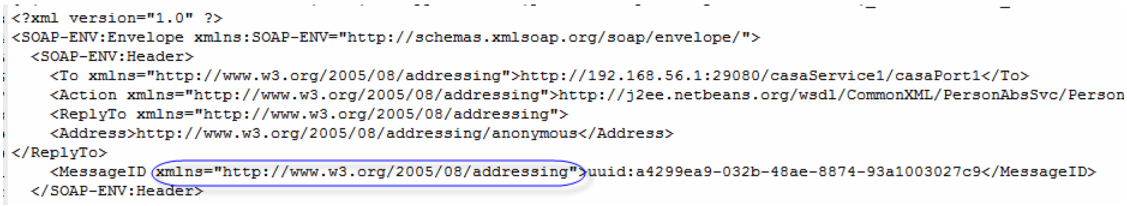
Figure 5.14.21 *nmPropertiesDefiniotions WSDL with WSA_MessageID property*

Note the stanza in Listing 5.14.3, which was generated by addition of the property through the wizard. This definition is not complete, as will transpire shortly.

Listing 5.14.3 Initial WSA_MessageID property definition

```
<vprop:property name="WSA_MessageID" type="xsd:anyType"/>
<vprop:propertyAlias
    messageType="ns:getPersonDetailsResponse"
    part="msgPersonDetailsRes"
    propertyName="tns:WSA_MessageID"
    sxnmp:nmProperty="org.glassfish.openesb.headers.soap">
    <vprop:query>wsa:MessageID/.</vprop:query>
</vprop:propertyAlias>
```

The XPath query, “wsa:MessageID/.”, will not work until we specify the namespace whose prefix is wsa and which “anchors” the relative path. Figure 5.14.22 shows the example WS-Addressing headers generated for the request. Note the namespace, to which the MessageID header belongs, and its relative position with respect to SOAP Header element.



```
<?xml version="1.0" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <To xmlns="http://www.w3.org/2005/08/addressing">http://192.168.56.1:29080/casaService1/casaPort1</To>
    <Action xmlns="http://www.w3.org/2005/08/addressing">http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc/Person
    <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
    <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
</ReplyTo>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:a4299ea9-032b-48ae-8874-93a1003027c9</MessageID>
  </SOAP-ENV:Header>
```

Figure 5.14.22 *MessageID header and the namespace to which it belongs*

The namespace, http://www.w3.org/2005/08/addressing, is the namespace defined in the WS-Addressing specification.

Manually edit the nmPropertiesDefinitions WSDL changing “<vprop:query>” tag to read “<vprop:query **xmlns:wsa=”http://www.w3.org/2005/08/addressing”**>”. This defines the wsa prefix used in the XPath expression and provides the context for this expression. Note, too, the literal “/.”, following “wsa:MessageID”. Empirical knowledge tells me that it is necessary to append this literal to the ‘top level” WS-Addressing nodes in order to obtain the actual value of the element. The final XML markup will look like that shown in Listing 5.14.4.

Listing 5.14.4 Modified WSA_MessageID property

```
<vprop:property name="WSA_MessageID" type="xsd:anyType"/>
<vprop:propertyAlias
    messageType="ns:getPersonDetailsResponse"
    part="msgPersonDetailsRes"
    propertyName="tns:WSA_MessageID"
    sxnmp:nmProperty="org.glassfish.openesb.headers.soap">
    <vprop:query
       xmlns:wsa=”http://www.w3.org/2005/08/addressing">wsa:MessageID/.</vprop:query>
</vprop:propertyAlias>
```

This property will allow us to read the value of the response’s MessageID property.

Now we need to define the NMProperty for the RelatesTo header so that we can read the original request MessageID property value.

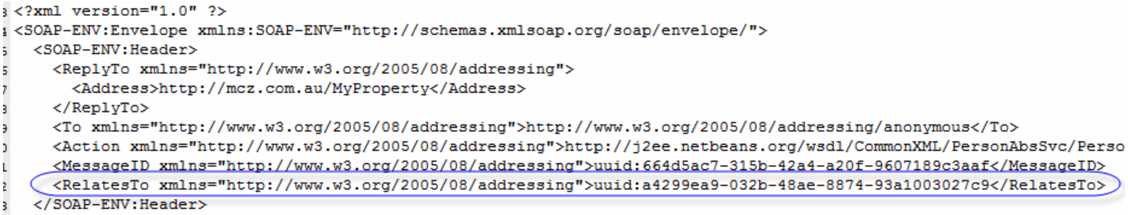
Replicate the XML structure shown in Listing 5.14.4 and change all occurrences of MessageID to RelatesTo. Listing 5.14.5 illustrates the fragment of nmPropertiesDefinitions WSDL after this is done.

Listing 5.14.4 nmPropertiesDefinitons WSDL with two NM Properties defined

```
<vprop:property name="WSA_MessageID" type="xsd:anyType"/>
<vprop:propertyAlias
    messageType="ns:getPersonDetailsResponse"
    part="msgPersonDetailsRes"
    propertyName="tns:WSA_MessageID"
    sxnmp:nmProperty="org.glassfish.openesb.headers.soap">
    <vprop:query
        xmlns:wsa="http://www.w3.org/2005/08/addressing">wsa:MessageID/.</vprop:query>
</vprop:propertyAlias>
```

```
<vprop:property name="WSA_RelatesTo" type="xsd:anyType"/>
<vprop:propertyAlias
    messageType="ns:getPersonDetailsResponse"
    part="msgPersonDetailsRes"
    propertyName="tns:WSA_RelatesTo"
    sxnmp:nmProperty="org.glassfish.openesb.headers.soap">
    <vprop:query
        xmlns:wsa="http://www.w3.org/2005/08/addressing">wsa:RelatesTo/.</vprop:query>
</vprop:propertyAlias>
```

Figure 5.14.23 shows a fragment of a response with WS-Addressing headers to allow us to place this property and the XPath query in the context of an actual XML Instance Document.



```
<?xml version="1.0" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
      <Address>http://mcz.com.au/MyProperty</Address>
    </ReplyTo>
    <To xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing/anonymous</To>
    <Action xmlns="http://www.w3.org/2005/08/addressing">http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc/Perso
    <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:664d5ac7-315b-42a4-a20f-9607189c3aaf</MessageID>
    <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:a4299ea9-032b-48ae-8874-93a1003027c9</RelatesTo>
  </SOAP-ENV:Header>
```

Figure 5.14.23 *WS-Addressing RelatesTo header in a response*

Now we need to modify the consumer BPEL process to echo to the log the response’s MessageID and RelatesTo header values.

Switch to the Design view, select the Assign2 activity and switch to Logging view. Add logging “mapping” as shown in Figure 5.14.24.

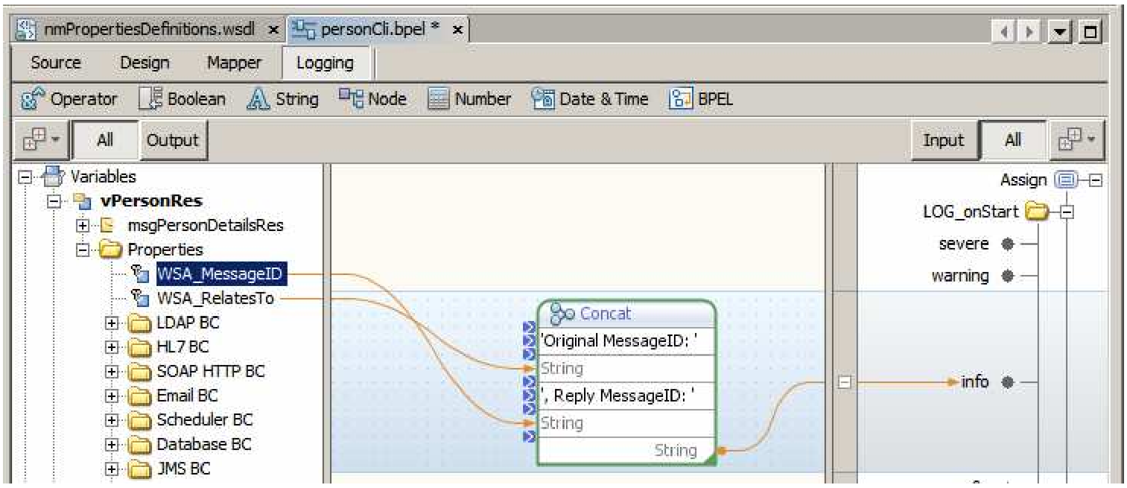


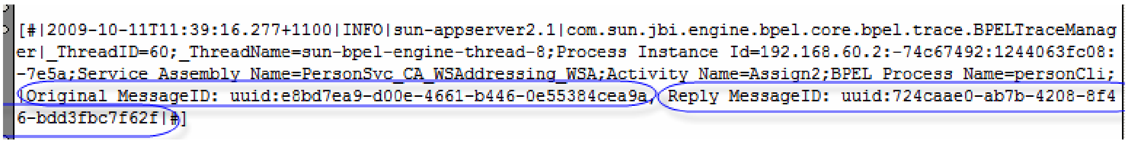
Figure 5.14.24 *Log MessageID values for the request and the response*

Build the BPEL Module project.

Create a new Composite Application Project, PersonCli_CA_WSAddressing_WSA, much as we have done in Section 5.14.1 with PersonCli_CA_WSAddressing, but use the PersonCli_WSA BPEL Module project instead of the PersonCli BPEL Module project. As we have done previously, create a new External WSDL Document using the service WSDL URL, “Load WSDL Port” into the CASA, connect, Build and Deploy.

Exercise the end-to-end solution using the PersonCli_WSTP Web Service Testing project.

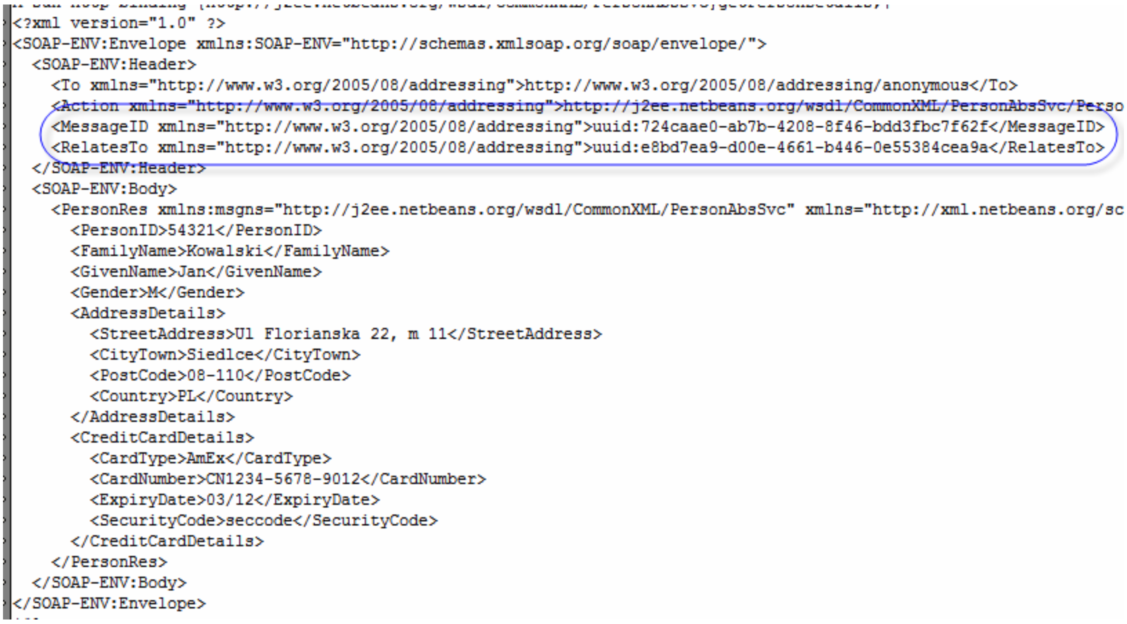
The server.log shows the trace of the request and response message IDs, as shown in Figure 5.14.25.



```
[#|2009-10-11T11:39:16.277+1100|INFO|sun-appserver2.1|com.sun.jbi.engine.bpel.core.bpel.trace.BPELTraceManag
er|_ThreadID=60;_ThreadName=sun-bpel-engine-thread-8;Process Instance Id=192.168.60.2:-74c67492:1244063fc08:
-7e5a;Service Assembly Name=PersonSvc_CA_WSAddressing_WSA;Activity Name=Assign2;BPEL Process Name=personCli;
Original MessageID: uuid:e8bd7ea9-d00e-4661-b446-0e55384cea9a, Reply MessageID: uuid:724caae0-ab7b-4208-8f4
6-bdd3fbc7f62f|#]
```

Figure 5.14.25 *Request with WS-Addressing MessageID header explicitly set*

The trace of the request message, in server.log, looks like that shown in Figure 5.14.26.



```
<?xml version="1.0" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <To xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing/anonymous</To>
    <Action xmlns="http://www.w3.org/2005/08/addressing">http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc/Perso
    <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:724caae0-ab7b-4208-8f46-bdd3fbc7f62f</MessageID>
    <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:e8bd7ea9-d00e-4661-b446-0e55384cea9a</RelatesTo>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <PersonRes xmlns:msgns="http://j2ee.netbeans.org/wsdl/CommonXML/PersonAbsSvc" xmlns="http://xml.netbeans.org/sc
      <PersonID>54321</PersonID>
      <FamilyName>Kowalski</FamilyName>
      <GivenName>Jan</GivenName>
      <Gender>M</Gender>
      <AddressDetails>
        <StreetAddress>Ul Florianska 22, m 11</StreetAddress>
        <CityTown>Siedlce</CityTown>
        <PostCode>08-110</PostCode>
        <Country>PL</Country>
      </AddressDetails>
      <CreditCardDetails>
        <CardType>AmEx</CardType>
        <CardNumber>CN1234-5678-9012</CardNumber>
        <ExpiryDate>03/12</ExpiryDate>
        <SecurityCode>seccode</SecurityCode>
      </CreditCardDetails>
    </PersonRes>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 5.14.26 SOAP Response with WS-Addressing headers

We demonstrated that the WS-Addressing headers can be accesses by the BPEL logic, if required.

For completeness' sake it is worth pointing out that WS-Addressing headers can also be set using NMProperties. Simply assign values to the properties. Dues to a bug in the infrastructure used by the GlassFish ESB v2.1 HTTP BC the headers will be duplicated, however, and will cause SOAP Faults to be thrown by the recipient of such malformed messages.

5.x Chapter Summary

This chapter explored selected methods of applying security to the channel over which SOAP messages are exchanged and the SOAP messages themselves, using a basic BPEL 2.0-based invoker and provider set.

A pair of projects, an invoker and a provider, were used to provide application logic.

Composite Applications were used to apply different variants of security policies.

The following security policies were explored:

- None
- HTP BC Channel Security - SSL / TLS with Server-side Authentication
- HTTP BC Channel Security - SSL / TLS with Mutual Authentication
- EJB Channel Security - SSL / TLS with Server-side Authentication
- EJB Channel Security - SSL / TLS with Mutual Authentication
- JBI-based Service – Exploring WS-Addressing
-
- Message Encryption
-

For each variant an end-to-end solution was built and exercised. Server.log traces from both sides were inspected and discussed as necessary to clarify what was happening during the process.