Java CAPS 5 / 6, OpenESB, GlassFish ESB
Handling SOAP Headers in BPEL

Michael.Czapski@sun.com, July 2009

Table of Contents

1 Introduction 1
2 Adding headers to an existing WSDL 1
3 Implement and test the service – JBI BPEL SE 16
4 Implement and test the service – eInsight 27
5 Summary 32

1 Introduction

Every now and then there arises a need to carry out-of-band information alongside the business payload but without disturbing or modifying it. Message Envelope Pattern is the Enterprise Integration Pattern label that is typically applied to solutions that address this issue. How the issue is addressed in practice varies depending on the technology in use. For JMS, for example, JMS Message Properties could be used to carry out-of-band information while the payload would be carried as the JMS payload. Web Services, typically using SOAP over HTTP, can address this requirement through the SOAP Header Extension Mechanism, whereby custom headers can be added to the SOAP Header while the payload is carried in the SOAP Body.

This document discusses construction of a WSDL that supports custom SOAP Header element and BPEL processes that are used to set and get custom header values in JBI and in eInsight. This mechanism is known to work in Java CAPS 5.x, Java CAPS 6 Classic and OpenESB / GlassFish ESB.

It is assumed that the reader is sufficiently familiar with the GlassFish ESB / OpenESB BPEL Service Engine and the SOAP/HTTP Binding Component, and / or Java CAPS Classic eInsight Business Process Manager and eDesigner IDE to be able to build projects without a step-by-step pictorial document.

2 Adding headers to an existing WSDL

It's all in the WSDL. Many WSLD definitions, one comes across, don't use SOAP Headers. When examples are shown, the examples usually show SOAP Headers generated as a result of application of WS-Security, and related technologies, to a "plain" SOAP messages. This accounts for why it seems hard to find information on how to use custom SOAP Headers or even that it is possible to add and process custom SOAP headers outside the realms of WS-*.

Let's take a simple WSDL that describes a Patient Details Lookup web service interface, PatientDetailsSvcFlat.wsdl.
The original WSDL is shown in Listing 2-1.

Listing 2-1 Original WSDL

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

```
name="PatientDetailsSvc"
targetNamespace="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsSvc"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tns="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsSvc"
xmlns:ns0="urn:Sun:Michael.Czapski:XSDs:/PatientDetails"
xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
>
<types>
    <xsd:schema
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="urn:Sun:Michael.Czapski:XSDs:/PatientDetails"
        xmlns:tns="urn:Sun:Michael.Czapski:XSDs:/PatientDetails"
        elementFormDefault="qualified"
        >
        <xsd:element name="elPatDetailsReq">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="Facility" type="xsd:string"/>
                    <xsd:element name="LocalID" type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="elPatIDListReq">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="Facility" type="xsd:string"/>
                    <xsd:element name="LocalID" type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="elPatDetailsRes">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="EUID" type="xsd:string"/>
                    <xsd:element name="Facility" type="xsd:string"/>
                    <xsd:element name="LocalID" type="xsd:string"/>
                    <xsd:element name="LID_SEQNUM" type="xsd:string"/>
                    <xsd:element name="LID_STATUS" type="xsd:string"/>
                    <xsd:element name="FamilyName" type="xsd:string"/>
                    <xsd:element name="GivenName" type="xsd:string"/>
                    <xsd:element name="MiddelInitialOrName"
                            type="xsd:string" minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="Suffix" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="Title" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="AddressLine1" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="AddressLine2" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="SuburbTown" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="State" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="PostCode" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="Country" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="DoB" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="Gender" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="MedicareNumber" type="xsd:string"
                            minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="elPatIDListRes">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="EUID" type="xsd:string"/>
                    <xsd:element name="PatientIDs">
                        <xsd:complexType>
                            <xsd:sequence>
```

```
                                <xsd:element name="LocalIDsList"
                                        minOccurs="1" maxOccurs="unbounded">
                                    <xsd:complexType>
                                        <xsd:sequence>
                                            <xsd:element name="Facility"
                                                    type="xsd:string"/>
                                            <xsd:element name="LocalID"
                                                    type="xsd:string"/>
                                        </xsd:sequence>
                                    </xsd:complexType>
                                </xsd:element>
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="elPatSvcFlt">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="LIDReq">
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:element name="Facility"
                                            type="xsd:string"/>
                                <xsd:element name="LocalID"
                                            type="xsd:string"/>
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                    <xsd:element name="DetailsType" type="xsd:string"/>
                    <xsd:element name="FaultDetails" type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:schema>
</types>
<message name="msgGetPatDetailsReq">
    <part name="sBriefLIDReq" element="ns0:elPatDetailsReq"/>
</message>
<message name="msgGetPatDetailsRes">
    <part name="sBriefRes" element="ns0:elPatDetailsRes"/>
</message>
<message name="msgGetPatIDListReq">
    <part name="sIDListLIDReq" element="ns0:elPatIDListReq"/>
</message>
<message name="msgGetPatIDListRes">
    <part name="selPatIDListRes" element="ns0:elPatIDListRes"/>
</message>
<message name="msgPatSvcFault">
    <part name="sFault" element="ns0:elPatSvcFlt"/>
</message>

<portType name="PatientDetailsSvcPortType">
    <operation name="opGetPatientDetailsBriefLID">
        <input
            name="inPatDetailsReq"
            message="tns:msgGetPatDetailsReq"/>
        <output
            name="outPatDetailsRes"
            message="tns:msgGetPatDetailsRes"/>
        <fault
            name="fltPatDetaislFlt"
            message="tns:msgPatSvcFault"/>
    </operation>
    <operation name="opGetPatientIDListLID">
        <input
            name="inPatIDListReq"
            message="tns:msgGetPatIDListReq"/>
        <output
            name="outPatIDListRes"
            message="tns:msgGetPatIDListRes"/>
        <fault
            name="fltPatIDListFlt"
            message="tns:msgPatSvcFault"/>
    </operation>
</portType>
```

```
    <binding name="PatientDetailsSvcPortTypeBinding"
            type="tns:PatientDetailsSvcPortType">
        <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="opGetPatientDetailsBriefLID">
            <soap:operation/>
            <input name="inPatDetailsReq">
                <soap:body use="literal"/>
            </input>
            <output name="outPatDetailsRes">
                <soap:body use="literal"/>
            </output>
            <fault name="fltPatDetaislFlt">
                <soap:fault name="fltPatDetaislFlt" use="literal"/>
            </fault>
        </operation>
        <operation name="opGetPatientIDListLID">
            <soap:operation/>
            <input name="inPatIDListReq">
                <soap:body use="literal"/>
            </input>
            <output name="outPatIDListRes">
                <soap:body use="literal"/>
            </output>
            <fault name="fltPatIDListFlt">
                <soap:fault name="fltPatIDListFlt" use="literal"/>
            </fault>
        </operation>
    </binding>
    <service name="PatientDetailsSvcFlatService">
        <port name="PatientDetailsSvcPortTypeBindingPort"
                binding="tns:PatientDetailsSvcPortTypeBinding">
            <soap:address
                location="http://localhost:${HttpDefaultPort}/service"/>
        </port>
    </service>
</definitions>
```

NetBeans IDE, part of Java CAPS, GlassFish ESB and OpenESB, has a fairly good WSDL editor. I find it convenient to work with it on WSDLs, switching to manual editing when that is more efficient. Let's create New Project -> SOA -> BPEL Module, named PatientWSDL, and create New -> External WSDL Document(s), Figure 2-1, pointing to the PatientDetailsSvcFlat.wsdl, shown in Listing 2-1.

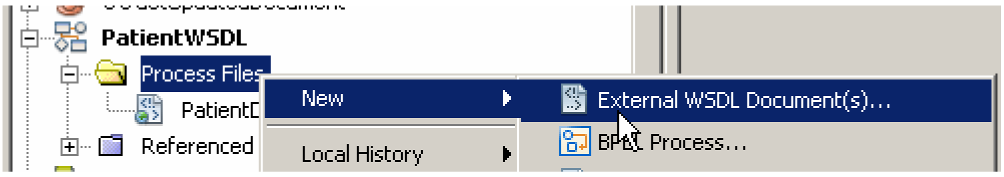


Figure 2-1 Trigger the Import external WSDL wizard

Once available, Edit the WSDL, switch to WSDL tab and click the Tree view, as shown in Figure 2-2.

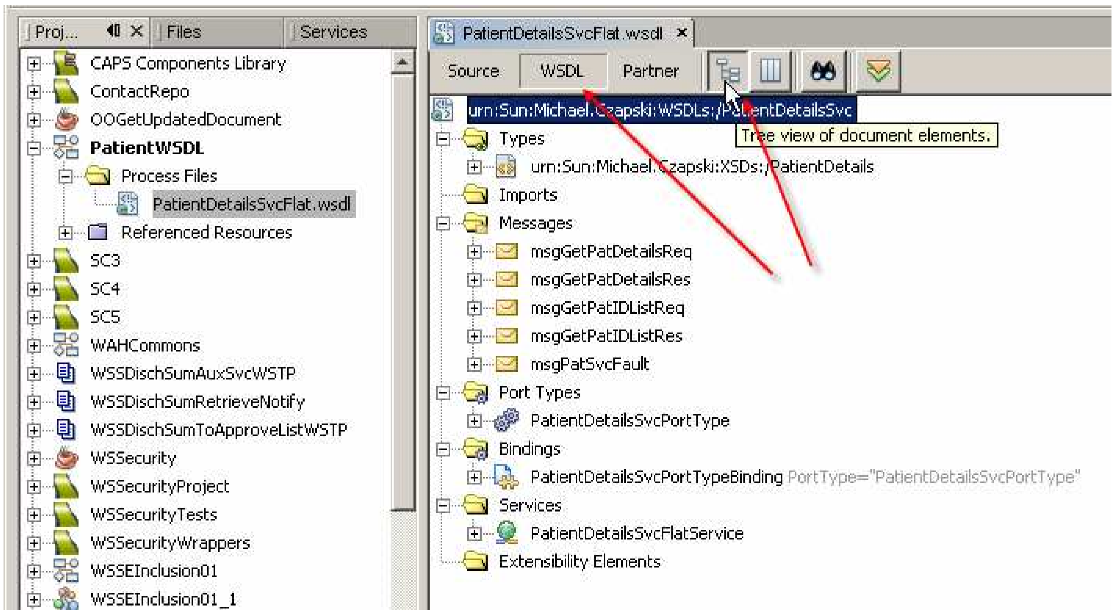


Figure 2-2 Edit WSDL in WSDL Mode, with Tree View

In this walkthrough we will add the xml structure, which will represent the custom SOAP Header, directly to the existing in-line xml schema. We could also have imported an external schema and used a definition from it instead.

Expand the Types structure, through Elements, right-click on the Elements node and choose Add Element, illustrated in figure 2-3.

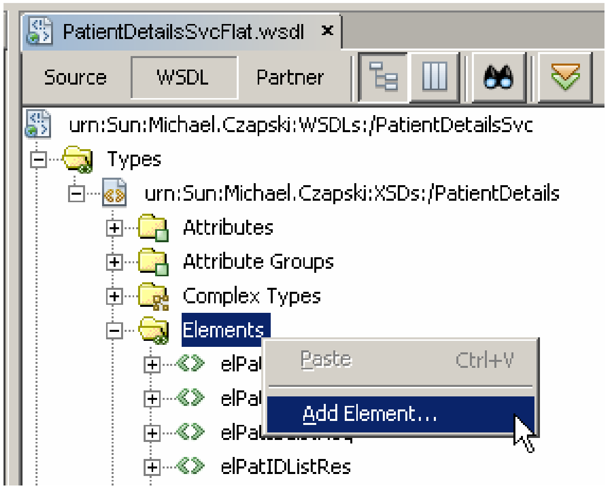
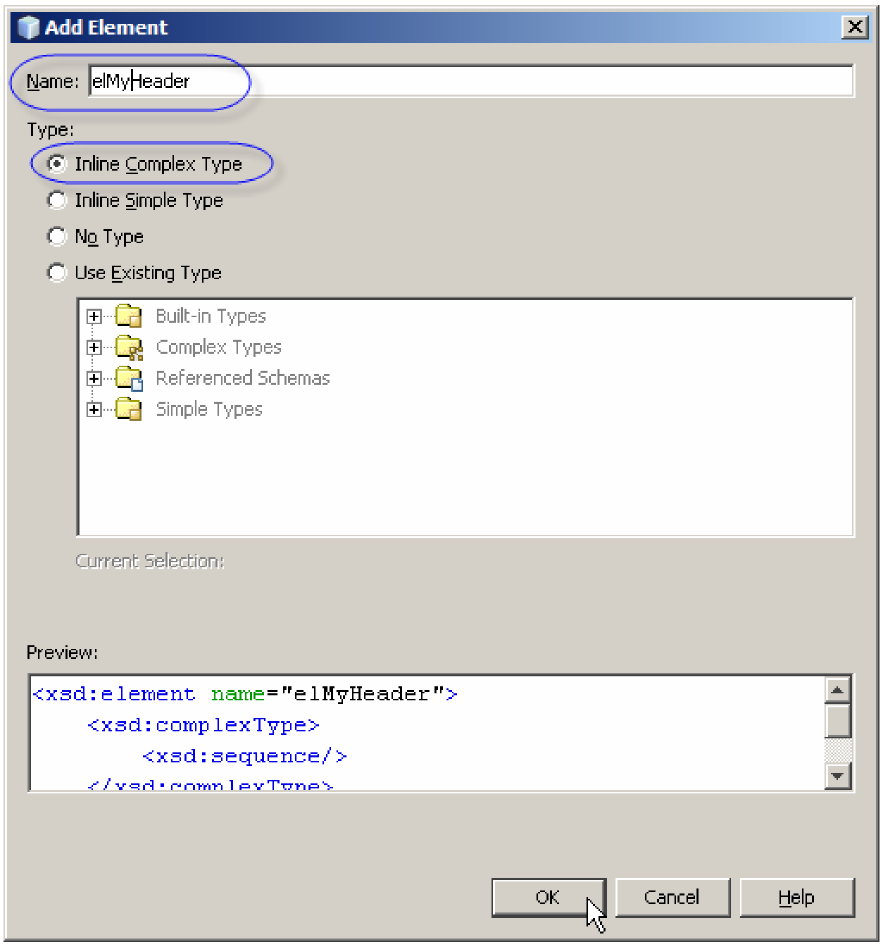


Figure 2-3 Trigger Add Element wizard

Name the new element elMyHeader, leaving the type at the default of Inline Complex Type. Figure 2-4 illustrates this. We will add further elements nested in elMyHeader.


Add Element

Name: elMyHeader

Type:

[✓ Inline Complex Type]

Inline Simple Type

No Type

Use Existing Type

Preview:

```
<xsd:element name="elMyHeader">
    <xsd:complexType>
        <xsd:sequence/>
```

Figure 2-4 Add new complex type, elMyHeader

Right-click elMyHeader and choose Add Element again, see Figure 2-5. Name the element SessionToken, click the “Use Existing Type” radio button and select “string” form the list. This is illustrated in Figure 2-6.

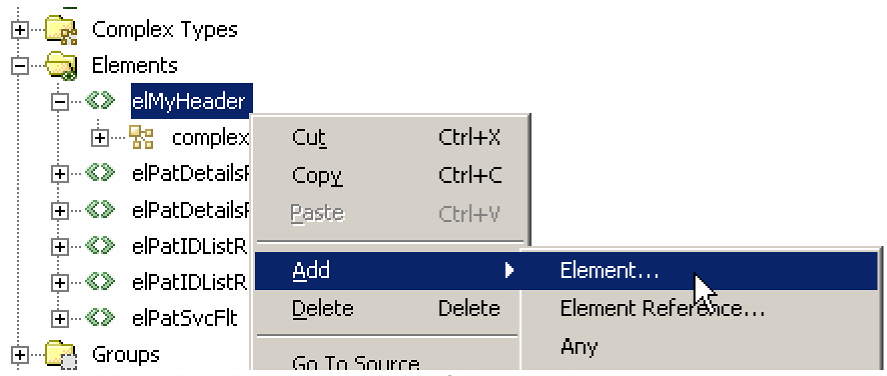
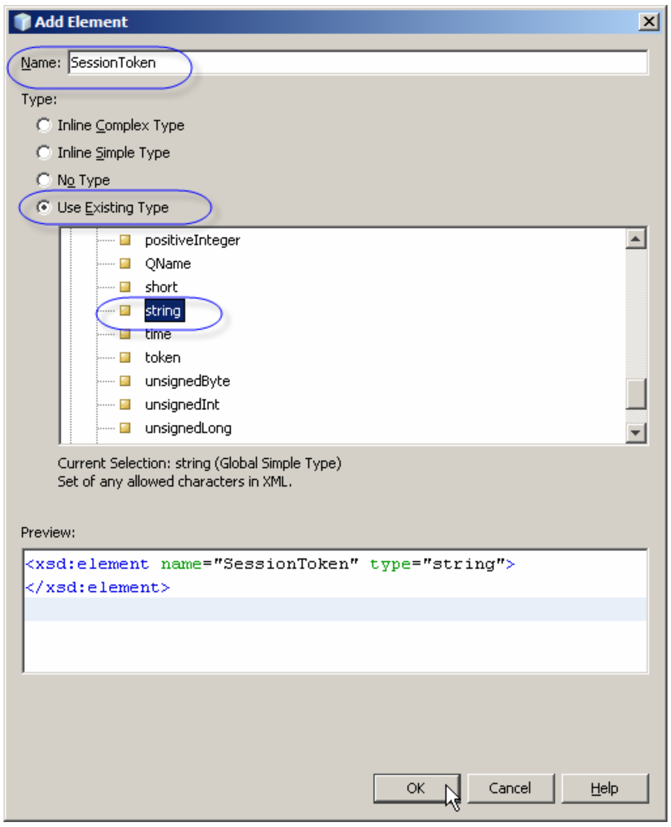


Figure 2-5 Add another element as a child of elMyHeader



Name: SessionToken

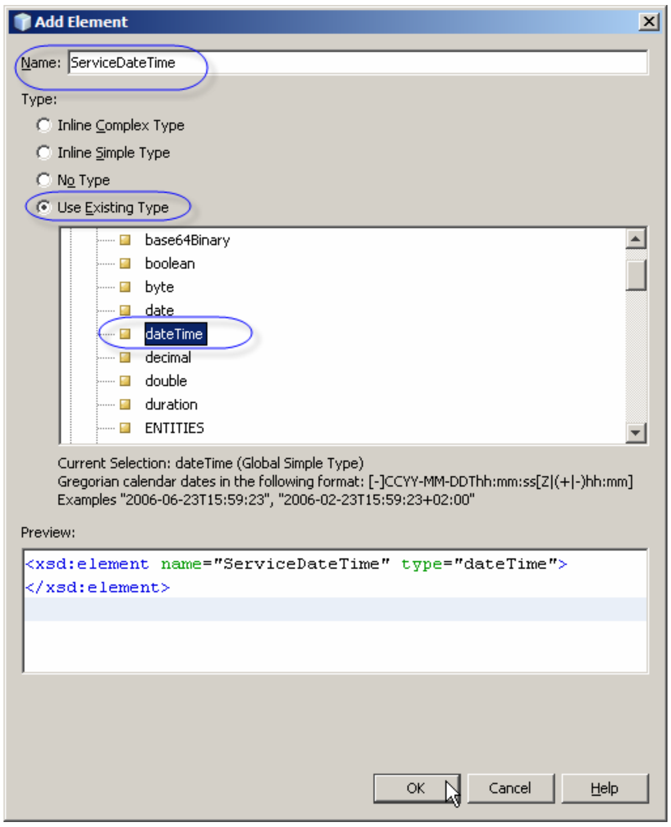
Use Existing Type: string

Current Selection: string (Global Simple Type)
Set of any allowed characters in XML.

```
<xsd:element name="SessionToken" type="string">
</xsd:element>
```

Figure 2-6 Name the new element and set its type.

Right-click elMyHeader again and add a child element named SessionDateTime of existing type dateTime, as shown in Figure 2-7.



Name: ServiceDateTime

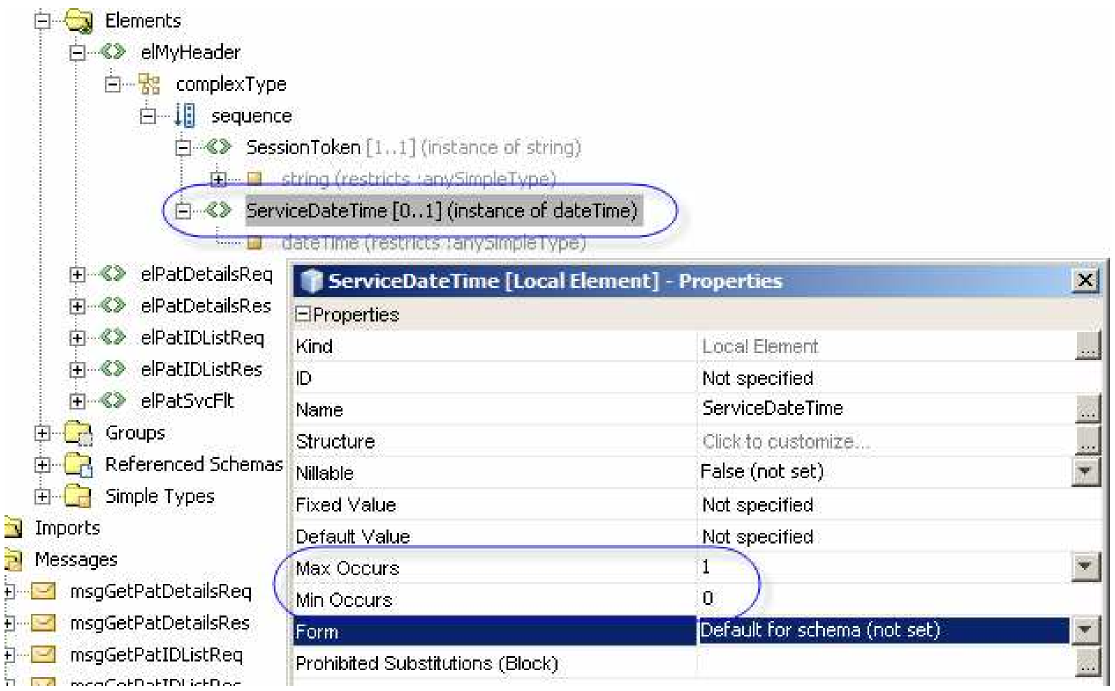
Use Existing Type: dateTime

Current Selection: dateTime (Global Simple Type)
Gregorian calendar dates in the following format: [-]CCYY-MM-DDThh:mm:ss[Z|(+|-)hh:mm]
Examples "2006-06-23T15:59:23", "2006-02-23T15:59:23+02:00"

```
<xsd:element name="ServiceDateTime" type="dateTime">
</xsd:element>
```

Figure 2-7 Add child element SessionDateTime of type dateTime

Right-click the new element SessionDateTime, choose Properties and change the MinOccurs property value to 0, to make the element optional, as shown in Figure 2-8.


Elements
elMyHeader
complexType
sequence
SessionToken [1..1] (instance of string)
string (restricts :anySimpleType)
ServiceDateTime [0..1] (instance of dateTime)
dateTime (restricts :anySimpleType)
elPatDetailsReq
elPatDetailsRes
elPatIDListReq
elPatIDListRes
elPatSvcFlt
Groups
Referenced Schemas
Simple Types
Imports
Messages
msgGetPatDetailsReq
msgGetPatDetailsRes
msgGetPatIDListReq

ServiceDateTime [Local Element] - Properties

Properties

Kind	Local Element
ID	Not specified
Name	ServiceDateTime
Structure	Click to customize...
Nillable	False (not set)
Fixed Value	Not specified
Default Value	Not specified
Max Occurs	1
Min Occurs	0
Form	Default for schema (not set)
Prohibited Substitutions (Block)	

Figure 2-8 Make SessionDateTime optional

We now have a data structure which the custom header will use. For service operation “opGetPatientDetailsBriefLID” we will require this header to be present in the request, with SessionToken carrying a value and SessionDateTime not present, and to be present in the response, with both SessionToken and SessionDateTime valued.

The messages for “opGetPatientDetailsBriefLID” are msgGetPatDetailsReq and msgGetPatDetailsRes. Each has a single part, sBriefLIDReq and sBriefRes respectively, as illustrated in Figure 2-9. The parts represent SOAP Body. We need to add a new part to each, to represent the SOAP Header.

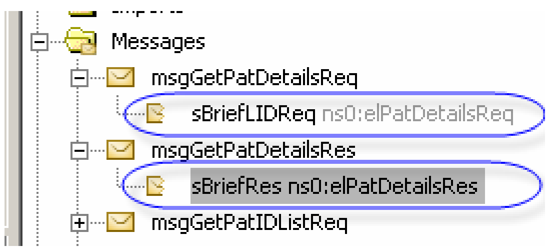

Messages
msgGetPatDetailsReq
sBriefLIDReq ns0:elPatDetailsReq
msgGetPatDetailsRes
sBriefRes ns0:elPatDetailsRes
msgGetPatIDListReq

Figure 2-9 Message parts representing structures carried in SOAP Body

Right-click the name of the request message, msgGetPatDetailsReq, choose Add and choose Part, as shown in Figure 2-10.

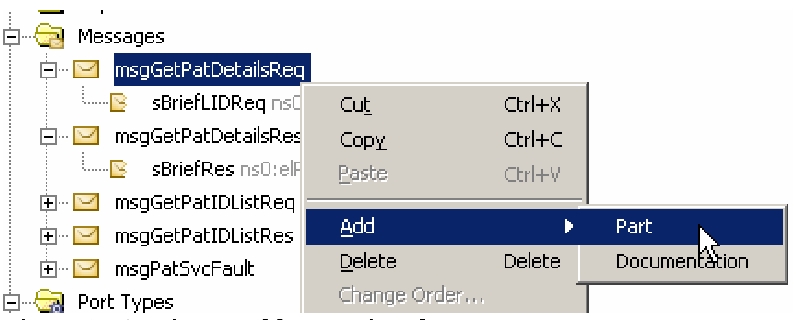


Figure 2-10 Trigger Add Part wizard

Refactor->Rename part to MyHeader, right-click MyHeader part and choose Properties, as illustrated in Figure 2-11.

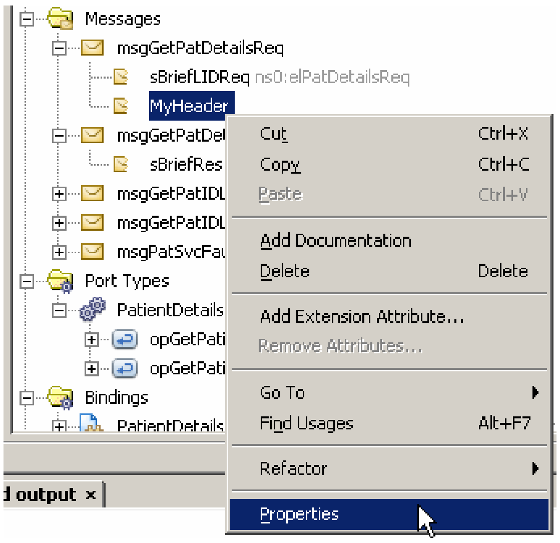


Figure 2-11 Open properties panel for the part

Click the ellipsis button to the right of the Element or Type property and choose MyHeader from the list of elements, as illustrated in Figure 2-12.

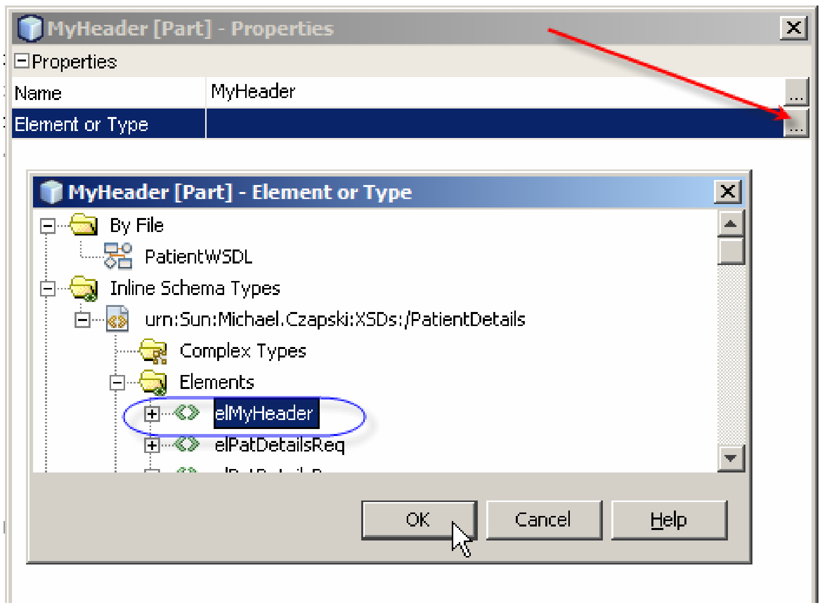


Figure 2-12 Set type for the new part

Repeat the process for the message msgGetPatDetailsRes.

Figure 2-13 illustrates modified messages.

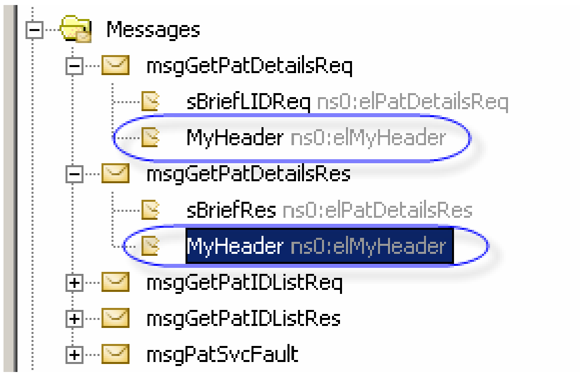


Figure 2-13 Messages with two parts

The abstract part of the WSDL now has two two-part message definitions. In the concrete part, in the binding stanza, we need to specify which parts of each message represent SOAP Body and which represent SOAP Headers.

Expand the Binding section through the inPatDetailsReq node, Right-click inPatDetailsReq, choose Add and choose SOAP Header, as shown in Figure 2-14.

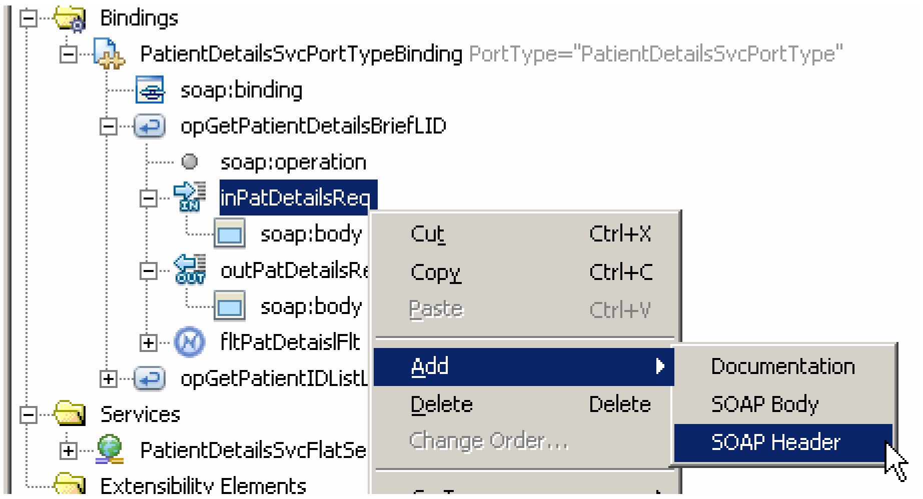


Figure 2-14 Trigger Add SOAP Header wizard

Right-click the new node soap:header and choose Properties, as shown in Figure 2-15.

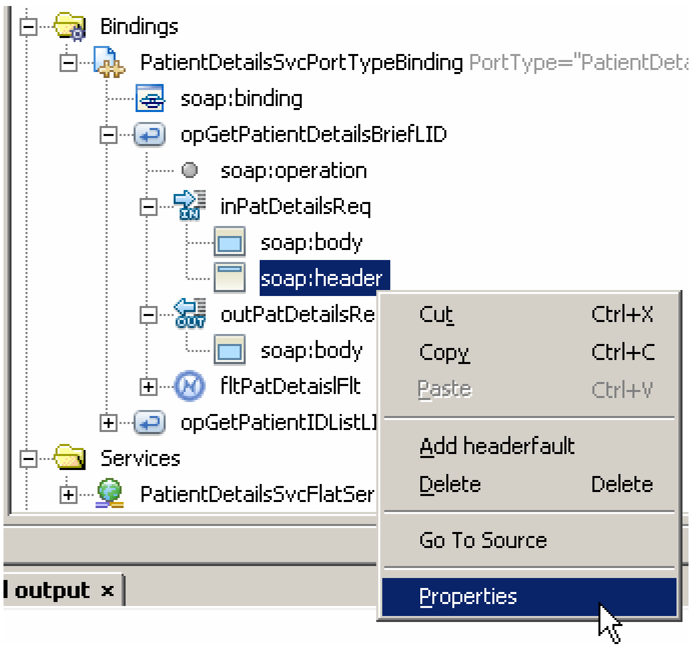


Figure 2-15 Open soap:header Proeprties panel

Pull down the drop-down menu, next to the message property and choose msgPatDetailsReq message, as illustrated in Figure 2-16.

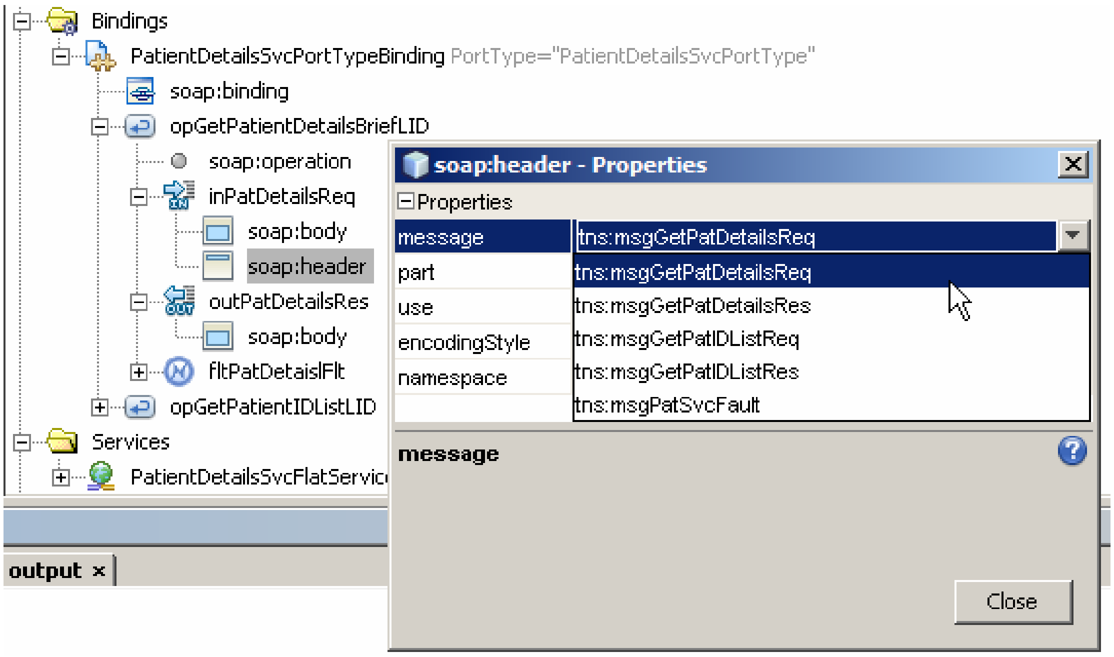
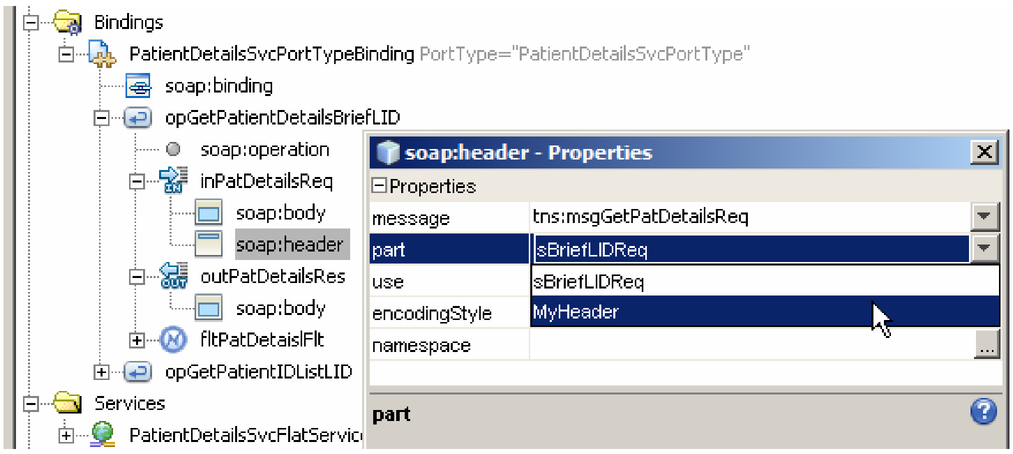


Figure 2-16 Choose message

Pull down the drop-down menu next to the part property, choose MyHeader and click Close. Figure 2-17 shows the snapshot of this.



message	tns:msgGetPatDetailsReq
part	sBriefLIDReq
use	sBriefLIDReq
encodingStyle	MyHeader
namespace	

Figure 2-17 Choose the type for the part

Right-click soap:body and choose properties, as shown in Figure 2-17.

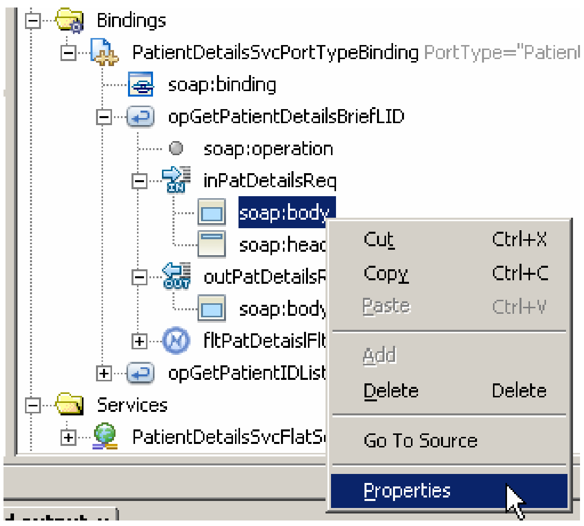
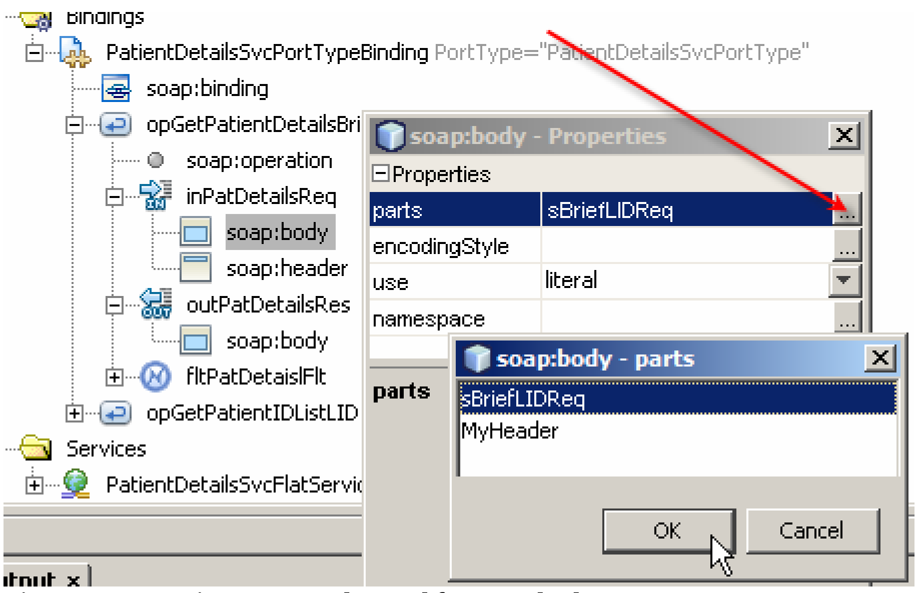


Figure 2-18 Open soap:body properties panel

Click the ellipsis button next to the parts property and choose the sBriefLIDReq part.



parts	sBriefLIDReq
encodingStyle	
use	literal
namespace	

Figure 2-19 Nominate part to be used for soap:body

Repeat the process for the outPatDetailsRes message, choosing msgPatDetailsRes and MyHeader for soap:header and sBriefRes for soap:body part.

Note that we did not add soap:header parts to the other operation, opGetPatientIDListLID, therefore soap headers will not be used for that operation's requests and responses.

The modified WSDL is shown in Listing 2-2 with new sections highlighted in bold typeface.

Listing 2-2 Modified WSDL

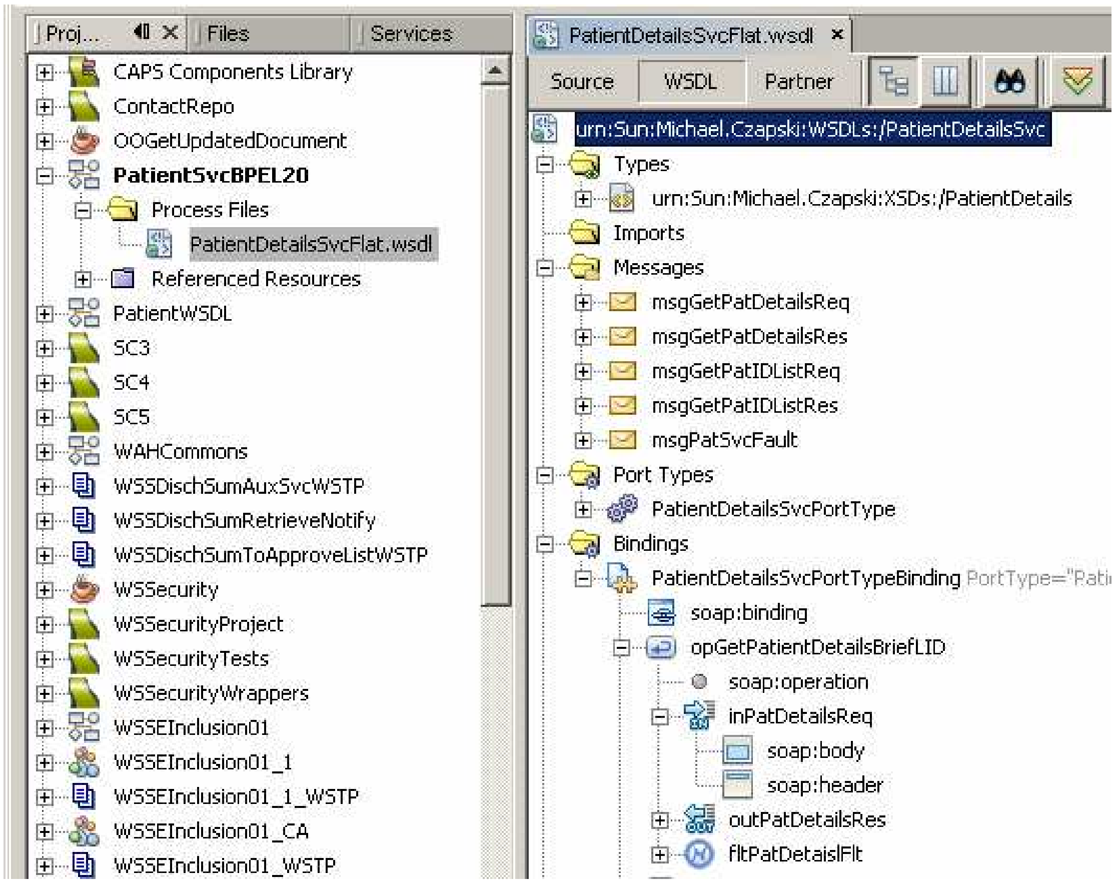
```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
    name="PatientDetailsSvc"
    targetNamespace="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsSvc"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsSvc"
    xmlns:ns0="urn:Sun:Michael.Czapski:XSDs:/PatientDetails"
    xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    >
    <types>
        <xsd:schema
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="urn:Sun:Michael.Czapski:XSDs:/PatientDetails"
            xmlns:tns="urn:Sun:Michael.Czapski:XSDs:/PatientDetails"
            elementFormDefault="qualified"
            >
            <xsd:element name="elPatDetailsReq">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="Facility" type="xsd:string"/>
                        <xsd:element name="LocalID" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="elPatIDListReq">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="Facility" type="xsd:string"/>
                        <xsd:element name="LocalID" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="elPatDetailsRes">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="EUID" type="xsd:string"/>
                        <xsd:element name="Facility" type="xsd:string"/>
                        <xsd:element name="LocalID" type="xsd:string"/>
                        <xsd:element name="LID_SEQNUM" type="xsd:string"/>
                        <xsd:element name="LID_STATUS" type="xsd:string"/>
                        <xsd:element name="FamilyName" type="xsd:string"/>
                        <xsd:element name="GivenName" type="xsd:string"/>
                        <xsd:element name="MiddelInitialOrName"
                                type="xsd:string" minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="Suffix" type="xsd:string"
                                        minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="Title" type="xsd:string"
                                        minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="AddressLine1" type="xsd:string"
                                        minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="AddressLine2" type="xsd:string"
                                        minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="SuburbTown" type="xsd:string"
                                        minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="State" type="xsd:string"
```

```
                                    minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="PostCode" type="xsd:string"
                                    minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="Country" type="xsd:string"
                                    minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="DoB" type="xsd:string"
                                    minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="Gender" type="xsd:string"
                                    minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="MedicareNumber" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="elPatIDListRes">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="EUID" type="xsd:string"/>
                    <xsd:element name="PatientIDs">
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:element name="LocalIDsList"
                                        minOccurs="1" maxOccurs="unbounded">
                                    <xsd:complexType>
                                        <xsd:sequence>
                                            <xsd:element name="Facility"
                                                    type="xsd:string"/>
                                            <xsd:element name="LocalID"
                                                    type="xsd:string"/>
                                        </xsd:sequence>
                                    </xsd:complexType>
                                </xsd:element>
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="elPatSvcFlt">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="LIDReq">
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:element name="Facility"
                                            type="xsd:string"/>
                                <xsd:element name="LocalID"
                                            type="xsd:string"/>
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                    <xsd:element name="DetailsType" type="xsd:string"/>
                    <xsd:element name="FaultDetails" type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="elMyHeader">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="SessionToken" type="xsd:string"/>
                    <xsd:element name="ServiceDateTime" type="xsd:dateTime"
                                minOccurs="0"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:schema>
</types>
<message name="msgGetPatDetailsReq">
    <part name="sBriefLIDReq" element="ns0:elPatDetailsReq"/>
    <part name="MyHeader" element="ns0:elMyHeader"/>
</message>
<message name="msgGetPatDetailsRes">
    <part name="sBriefRes" element="ns0:elPatDetailsRes"/>
    <part name="MyHeader" element="ns0:elMyHeader"/>
</message>
<message name="msgGetPatIDListReq">
    <part name="sIDListLIDReq" element="ns0:elPatIDListReq"/>
    </message>
    <message name="msgGetPatIDListRes">
        <part name="selPatIDListRes" element="ns0:elPatIDListRes"/>
    </message>
    <message name="msgPatSvcFault">
        <part name="sFault" element="ns0:elPatSvcFlt"/>
    </message>

    <portType name="PatientDetailsSvcPortType">
        <operation name="opGetPatientDetailsBriefLID">
            <input
                name="inPatDetailsReq"
                message="tns:msgGetPatDetailsReq"/>
            <output
                name="outPatDetailsRes"
                message="tns:msgGetPatDetailsRes"/>
            <fault
                name="fltPatDetaislFlt"
                message="tns:msgPatSvcFault"/>
        </operation>
        <operation name="opGetPatientIDListLID">
            <input
                name="inPatIDListReq"
                message="tns:msgGetPatIDListReq"/>
            <output
                name="outPatIDListRes"
                message="tns:msgGetPatIDListRes"/>
            <fault
                name="fltPatIDListFlt"
                message="tns:msgPatSvcFault"/>
        </operation>
    </portType>
    <binding name="PatientDetailsSvcPortTypeBinding"
            type="tns:PatientDetailsSvcPortType">
        <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="opGetPatientDetailsBriefLID">
            <soap:operation/>
            <input name="inPatDetailsReq">
                <soap:body use="literal" parts="sBriefLIDReq"/>
                <soap:header message="tns:msgGetPatDetailsReq"
                            part="MyHeader" use="literal"/>
            </input>
            <output name="outPatDetailsRes">
                <soap:body use="literal" parts="sBriefRes"/>
                <soap:header message="tns:msgGetPatDetailsRes"
                            part="MyHeader" use="literal"/>
            </output>
            <fault name="fltPatDetaislFlt">
                <soap:fault name="fltPatDetaislFlt" use="literal"/>
            </fault>
        </operation>
        <operation name="opGetPatientIDListLID">
            <soap:operation/>
            <input name="inPatIDListReq">
                <soap:body use="literal"/>
            </input>
            <output name="outPatIDListRes">
                <soap:body use="literal"/>
            </output>
            <fault name="fltPatIDListFlt">
                <soap:fault name="fltPatIDListFlt" use="literal"/>
            </fault>
        </operation>
    </binding>
    <service name="PatientDetailsSvcFlatService">
        <port name="PatientDetailsSvcPortTypeBindingPort"
                binding="tns:PatientDetailsSvcPortTypeBinding">
            <soap:address
                location="http://localhost:${HttpDefaultPort}/service"/>
        </port>
    </service>
</definitions>
```

3 Implement and test the service – JBI BPEL SE

Create a New -> SOA -> BPEL Module named PatientSvcBPEL20. In that project create a New -> External WSDL Document(s), using the PatientDetailsSvcFlat.wsdl which we modified in the previous section. Figure 3-1 shows a snapshot of the project at the end of this process.



```
CAPS Components Library
ContactRepo
OOGetUpdatedDocument
PatientSvcBPEL20
  Process Files
    PatientDetailsSvcFlat.wsdl
  Referenced Resources
PatientWSDL
SC3
SC4
SC5
WAHCommons
WSSDischSumAuxSvcWSTP
WSSDischSumRetrieveNotify
WSSDischSumToApproveListWSTP
WSSecurity
WSSecurityProject
WSSecurityTests
WSSecurityWrappers
WSSEInclusion01
WSSEInclusion01_1
WSSEInclusion01_1_WSTP
WSSEInclusion01_CA
WSSEInclusion01_WSTP
```

```
urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsSvc
  Types
    urn:Sun:Michael.Czapski:XSDs:/PatientDetails
  Imports
  Messages
    msgGetPatDetailsReq
    msgGetPatDetailsRes
    msgGetPatIDListReq
    msgGetPatIDListRes
    msgPatSvcFault
  Port Types
    PatientDetailsSvcPortType
  Bindings
    PatientDetailsSvcPortTypeBinding PortType="Pati
      soap:binding
      opGetPatientDetailsBriefLID
        soap:operation
        inPatDetailsReq
          soap:body
          soap:header
        outPatDetailsRes
        fltPatDetaislFlt
```

Figure 3-1 PatientSvcBPEL project with WSDL added

Create New -> BPEL Process, named bpPatientSvcBPEL, drag the WSDL onto the canvas naming it PatientRR, and add pick-assign-reply activities, adding variables, vPatDetailsReq and vPatDetailsRes, to onMessage and Reply activities respectively. Figure 3-2 shows a snapshot of the process. Make sure to check the "Create Instance" property's checkbox on the Pick activity.

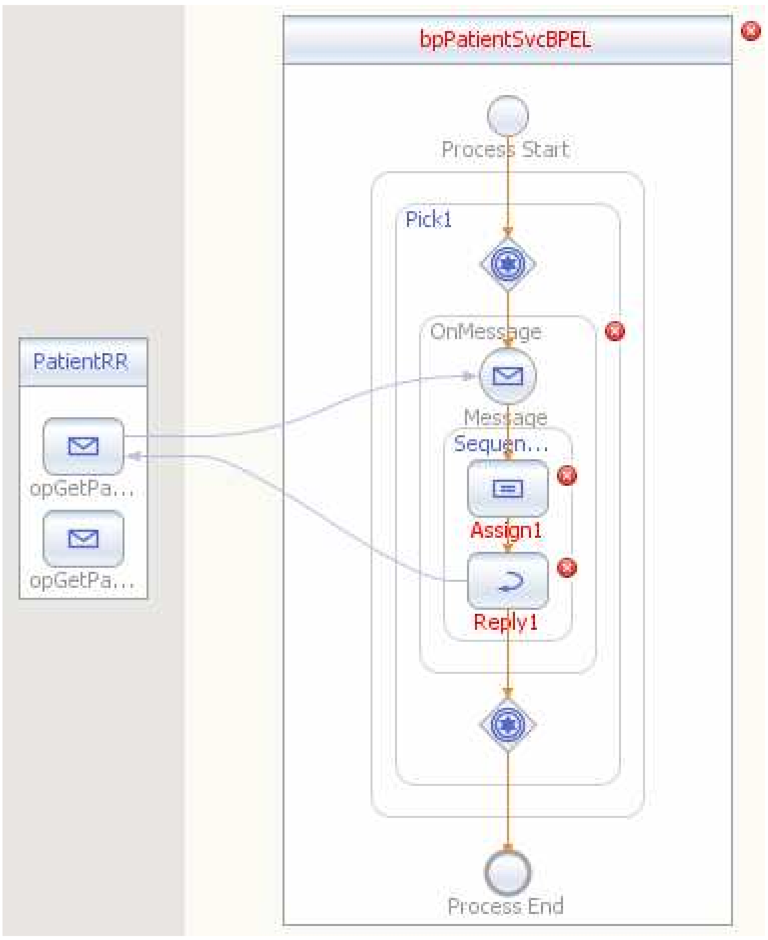


Figure 3-2 Process snapshot

Select the Assign activity and switch to the mapper view. Note the header nodes in the request and reply structures, shown in Figure 3-3.

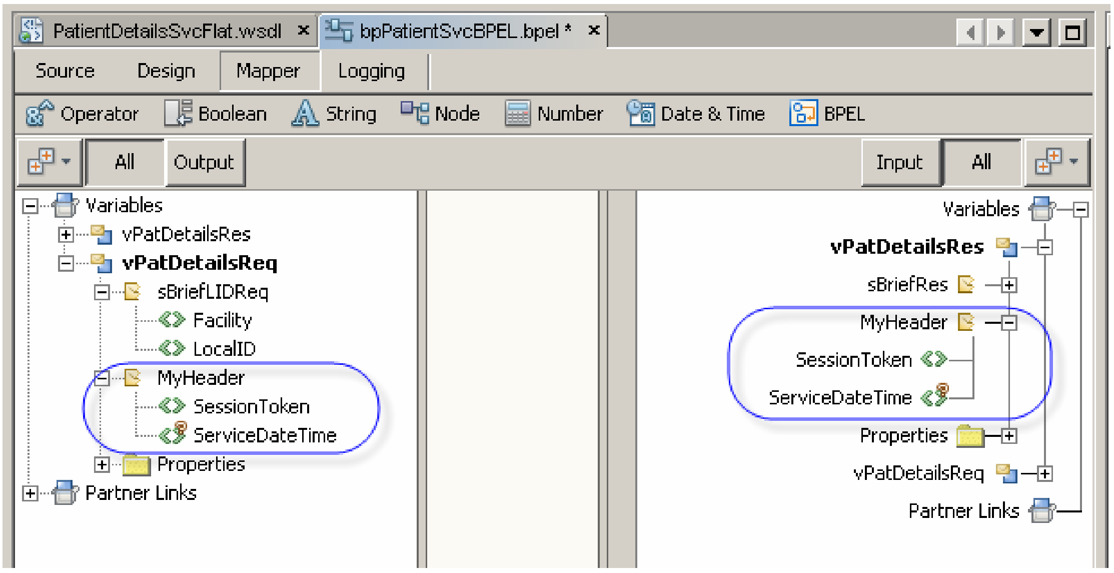


Figure 3-3 header nodes in the request and the reply structures

Normally, we would invoke a Database BC-based service to obtain data from the database and would use this data to populate the response message. In this case will we will merely assign literals values to the required nodes in the patient details response message, leaving optional nodes unmapped. It does not matter what values we assign to patient details so long as the data types are correct. Figure 3-4 illustrates mapping of request body to response body and assignment of literal values to required nodes.

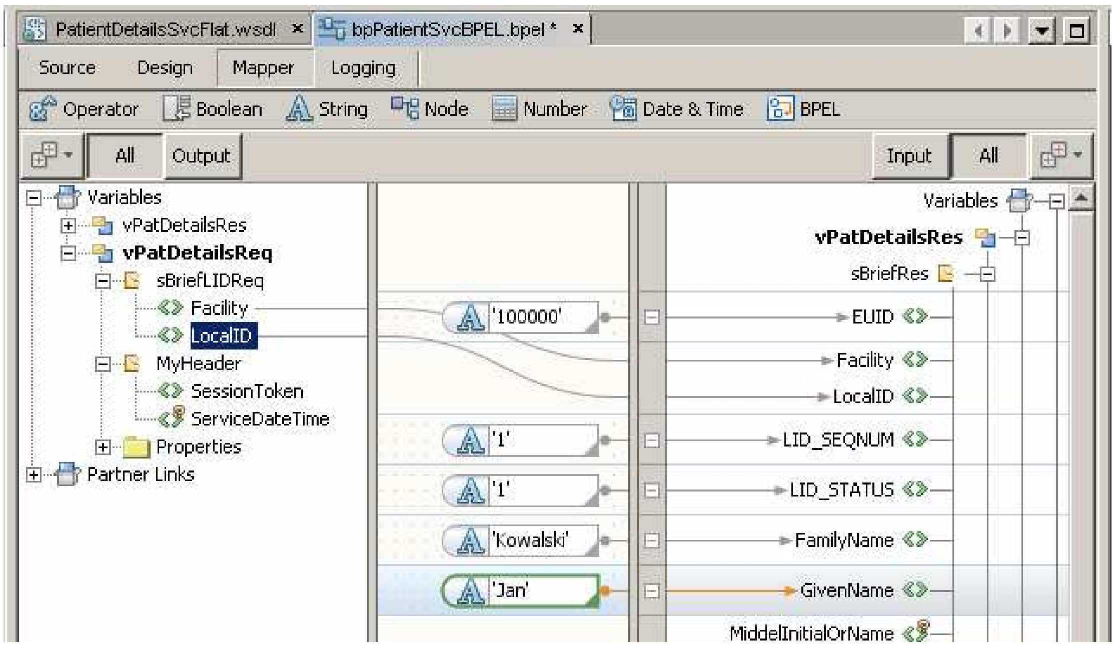


Figure 3-4 Mapping request body to response body and populating required body nodes

Let's now turn to the header nodes. The SessionToken will be passed as is from the request to the response header nodes. SessionDateTime will be assigned using a data/time function. Figure 3-5 illustrates the mapping.

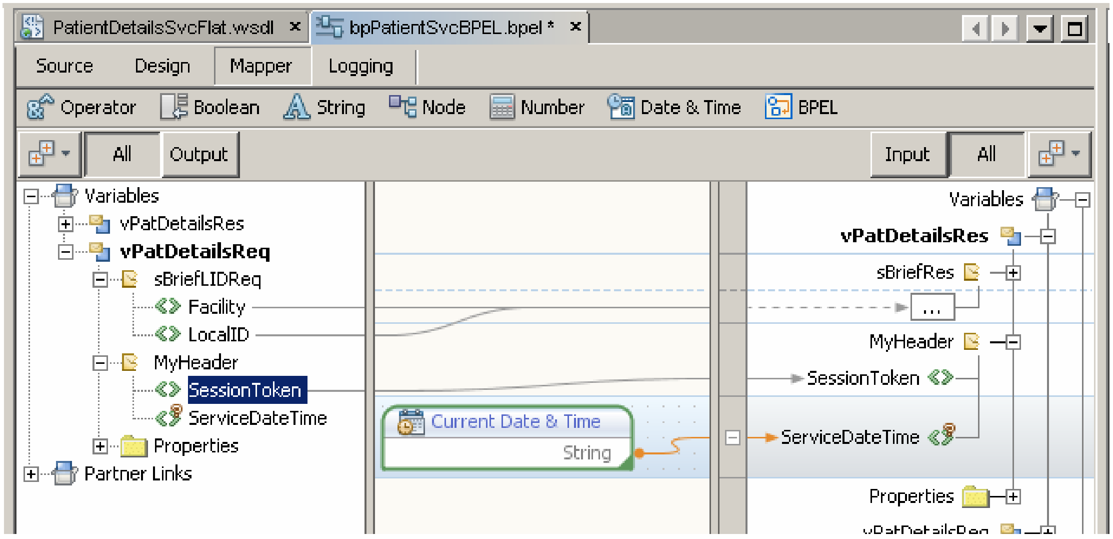


Figure 3-5 Mapping header nodes

Build the project.

Create a New -> SOA -> Composite Application project, PatientSvcBPEL20_CA, drag the PatientSvcBPEL20 project onto the canvas, build and deploy. Figure 3-6 shows the Composite Application Service Assembly editor with the CASA map of the application.

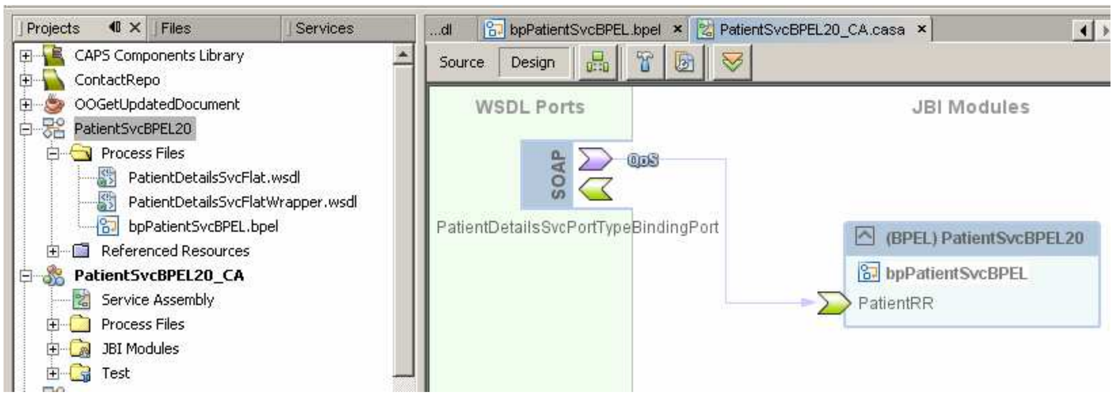


Figure 3-6 CASA map of the composite application

Take a look at the WSDL to determine the soap:address->location attribute value. For me this is:

```
http://localhost:${HttpDefaultPort}/service
```

Replacing ${HttpDefaultPort} with the runtime value from the sun-http-binding component's properties, I have the resulting address as:

```
http://localhost:39080/service
```

Use the web browser to confirm that the service is deployed by asking for its WSDL with the URL:

```
http://localhost:39080/service?WSDL
```

Let's now create a New -> Enterprise -> Web Services Testing Project, called PatientSvcBPEL20_WSTP, using the WSDL URL shown above. This assumes that the SoapUI PlugIn has been installed. The project is shown in Figure 3-7.

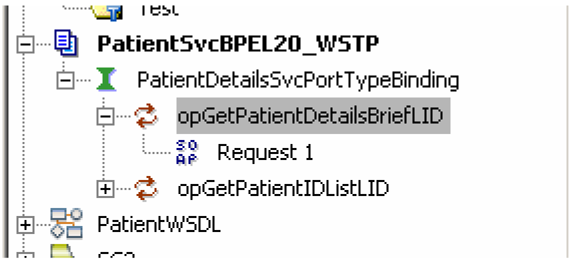
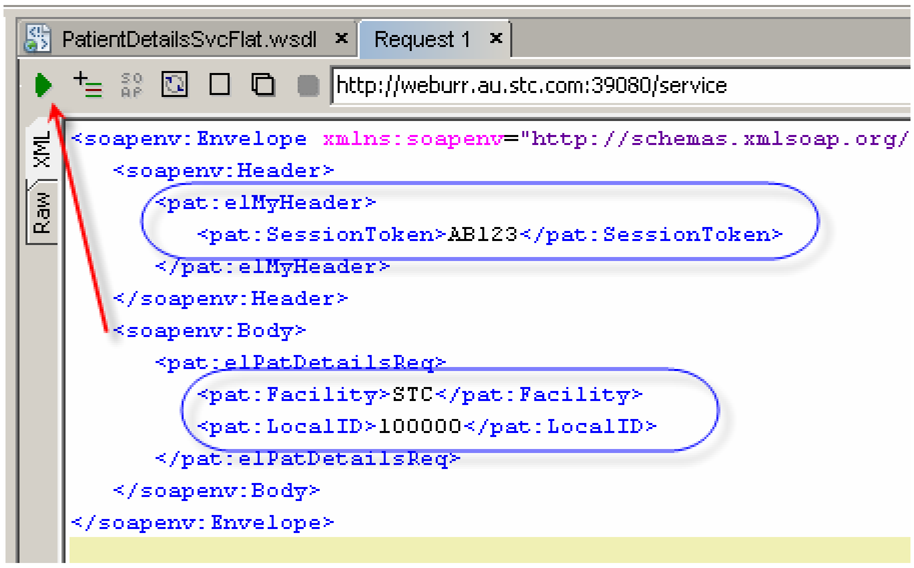


Figure 3-7 Web Service Testing Project for the Composite Application service

Expand the opGetPatientDetailsBriefLID operation, right-click the "Request 1" and choose Show Request Editor.

Remove the SessionDateTime optional header node, supply values for the remaining header and body nodes and click the Submit Request button. Figure 3-8 illustrates the request and points out the Submit Request button.

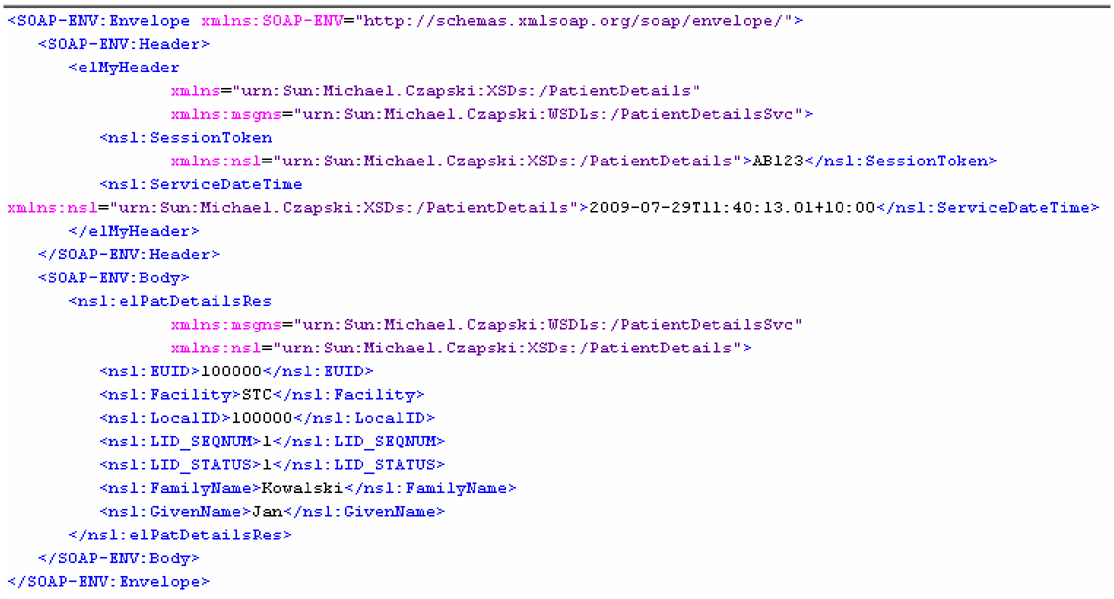


```
http://weburr.au.stc.com:39080/service
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
   <soapenv:Header>
      <pat:elMyHeader>
         <pat:SessionToken>AB123</pat:SessionToken>
      </pat:elMyHeader>
   </soapenv:Header>
   <soapenv:Body>
      <pat:elPatDetailsReq>
         <pat:Facility>STC</pat:Facility>
         <pat:LocalID>100000</pat:LocalID>
      </pat:elPatDetailsReq>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 3-8 SOAP Request

The response, reformatted to somewhat improve readability, is shown in Figure 3-9.



```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header>
      <elMyHeader
                xmlns="urn:Sun:Michael.Czapski:XSDs:/PatientDetails"
                xmlns:msgns="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsSvc">
         <ns1:SessionToken
                xmlns:ns1="urn:Sun:Michael.Czapski:XSDs:/PatientDetails">AB123</ns1:SessionToken>
         <ns1:ServiceDateTime
xmlns:ns1="urn:Sun:Michael.Czapski:XSDs:/PatientDetails">2009-07-29T11:40:13.01+10:00</ns1:ServiceDateTime>
      </elMyHeader>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <ns1:elPatDetailsRes
                xmlns:msgns="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsSvc"
                xmlns:ns1="urn:Sun:Michael.Czapski:XSDs:/PatientDetails">
         <ns1:EUID>100000</ns1:EUID>
         <ns1:Facility>STC</ns1:Facility>
         <ns1:LocalID>100000</ns1:LocalID>
         <ns1:LID_SEQNUM>1</ns1:LID_SEQNUM>
         <ns1:LID_STATUS>1</ns1:LID_STATUS>
         <ns1:FamilyName>Kowalski</ns1:FamilyName>
         <ns1:GivenName>Jan</ns1:GivenName>
      </ns1:elPatDetailsRes>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 3-9 Web Service response

Note the custom SOAP Header nodes in the request and the response.

Let’s now build a BPEL 2.0-based service consumer to exercise the solution end-to-end.

Create a New -> SOA -> BPEL Module, named PatientCliBPEL20.

Create a New -> External WSDL Document(s), providing the URL of the deployed service. For me this will be http://localhost:39080/service?WSDL. This will represent the interface to the service.

Create New -> WSDL Document, named PatientDetailsSvcNoHeaders.wsdl, accepting all defaults. Replace the content of the new WSDL document with the WSDL text shown in Listing 3-1.

Listing 3-1 WSDL with no headers

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
    name="PatientDetailsNoHeadersSvc"
    targetNamespace="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsNoHeadersSvc"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsNoHeadersSvc"
    xmlns:ns0="urn:Sun:Michael.Czapski:XSDs:/PatientDetailsNoHeaders"
    xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    >
    <types>
        <xsd:schema
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="urn:Sun:Michael.Czapski:XSDs:/PatientDetailsNoHeaders"
            xmlns:tns="urn:Sun:Michael.Czapski:XSDs:/PatientDetailsNoHeaders"
            elementFormDefault="qualified"
            >
            <xsd:element name="elPatDetailsReq">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="Facility" type="xsd:string"/>
                        <xsd:element name="LocalID" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="elPatIDListReq">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="Facility" type="xsd:string"/>
                        <xsd:element name="LocalID" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="elPatDetailsRes">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="EUID" type="xsd:string"/>
                        <xsd:element name="Facility" type="xsd:string"/>
                        <xsd:element name="LocalID" type="xsd:string"/>
                        <xsd:element name="LID_SEQNUM" type="xsd:string"/>
                        <xsd:element name="LID_STATUS" type="xsd:string"/>
                        <xsd:element name="FamilyName" type="xsd:string"/>
                        <xsd:element name="GivenName" type="xsd:string"/>
                        <xsd:element name="MiddelInitialOrName"
                                type="xsd:string" minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="Suffix" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="Title" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="AddressLine1" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="AddressLine2" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="SuburbTown" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="State" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="PostCode" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="Country" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="DoB" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="Gender" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="MedicareNumber" type="xsd:string"
                            minOccurs="0" maxOccurs="1"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="elPatIDListRes">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="EUID" type="xsd:string"/>
                        <xsd:element name="PatientIDs">
                            <xsd:complexType>
                                <xsd:sequence>
                            <xsd:element name="LocalIDsList"
                                    minOccurs="1" maxOccurs="unbounded">
                                <xsd:complexType>
                                    <xsd:sequence>
                                        <xsd:element name="Facility"
                                                type="xsd:string"/>
                                        <xsd:element name="LocalID"
                                                type="xsd:string"/>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="elPatSvcFlt">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="LIDReq">
                            <xsd:complexType>
                                <xsd:sequence>
                            <xsd:element name="Facility"
                                        type="xsd:string"/>
                            <xsd:element name="LocalID"
                                        type="xsd:string"/>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="DetailsType" type="xsd:string"/>
                        <xsd:element name="FaultDetails" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:schema>
    </types>
    <message name="msgGetPatDetailsReq">
        <part name="sBriefLIDReq" element="ns0:elPatDetailsReq"/>
    </message>
    <message name="msgGetPatDetailsRes">
        <part name="sBriefRes" element="ns0:elPatDetailsRes"/>
    </message>
    <message name="msgGetPatIDListReq">
        <part name="sIDListLIDReq" element="ns0:elPatIDListReq"/>
    </message>
    <message name="msgGetPatIDListRes">
        <part name="selPatIDListRes" element="ns0:elPatIDListRes"/>
    </message>
    <message name="msgPatSvcFault">
        <part name="sFault" element="ns0:elPatSvcFlt"/>
    </message>

    <portType name="PatientDetailsNoHeadersSvcPortType">
        <operation name="opGetPatientDetailsNoHeadersBriefLID">
            <input
                name="inPatDetailsReq"
                message="tns:msgGetPatDetailsReq"/>
            <output
                name="outPatDetailsRes"
                message="tns:msgGetPatDetailsRes"/>
            <fault
                name="fltPatDetaislFlt"
                message="tns:msgPatSvcFault"/>
        </operation>
        <operation name="opGetPatientIDListLID">
            <input
```

```
                name="inPatIDListReq"
                message="tns:msgGetPatIDListReq"/>
            <output
                name="outPatIDListRes"
                message="tns:msgGetPatIDListRes"/>
            <fault
                name="fltPatIDListFlt"
                message="tns:msgPatSvcFault"/>
        </operation>
    </portType>
    <binding name="PatientDetailsNoHeadersSvcPortTypeBinding"
            type="tns:PatientDetailsNoHeadersSvcPortType">
        <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="opGetPatientDetailsNoHeadersBriefLID">
            <soap:operation/>
            <input name="inPatDetailsReq">
                <soap:body use="literal"/>
            </input>
            <output name="outPatDetailsRes">
                <soap:body use="literal"/>
            </output>
            <fault name="fltPatDetaislFlt">
                <soap:fault name="fltPatDetaislFlt" use="literal"/>
            </fault>
        </operation>
        <operation name="opGetPatientIDListLID">
            <soap:operation/>
            <input name="inPatIDListReq">
                <soap:body use="literal"/>
            </input>
            <output name="outPatIDListRes">
                <soap:body use="literal"/>
            </output>
            <fault name="fltPatIDListFlt">
                <soap:fault name="fltPatIDListFlt" use="literal"/>
            </fault>
        </operation>
    </binding>
    <service name="PatientDetailsNoHeadersSvcFlatService">
        <port name="PatientDetailsNoHeadersSvcPortTypeBindingPort"
                binding="tns:PatientDetailsNoHeadersSvcPortTypeBinding">
            <soap:address
                location="http://localhost:${HttpDefaultPort}/client"/>
        </port>
    </service>
</definitions>
```

Notice that this WSDL is very much like the WSDL we started with, except namespaces are somewhat different. We will not be adding soap headers to this WSDL. It will be used to trigger the client, which invokes our web service provider, then return service response.

Create a BPEL Process, bpPatientCliBPEL20. Use the PatientDetailsSvcNoHeaders.wsdl at the client side, that is on the left hand side of the process – call the partner link PatSvcRR - and service.wsdl a the service side, that is at the right hand side of the process model – call the partner link PatSvcWS.
Add Receive, Assign, Invoke, Assign and Reply activities. Create input and output variables for the Receive, Invoke and Reply activities. Connect Receive and Reply activities to the opGetPatientDetailsNoHeadersBriefLID operation of the PatSvcRR partner and the Invoke operation to the opGetPatientDetailsBriefLid of the PatSvcWS partner. Figure 3-10 shows a snapshot of the process at this time.

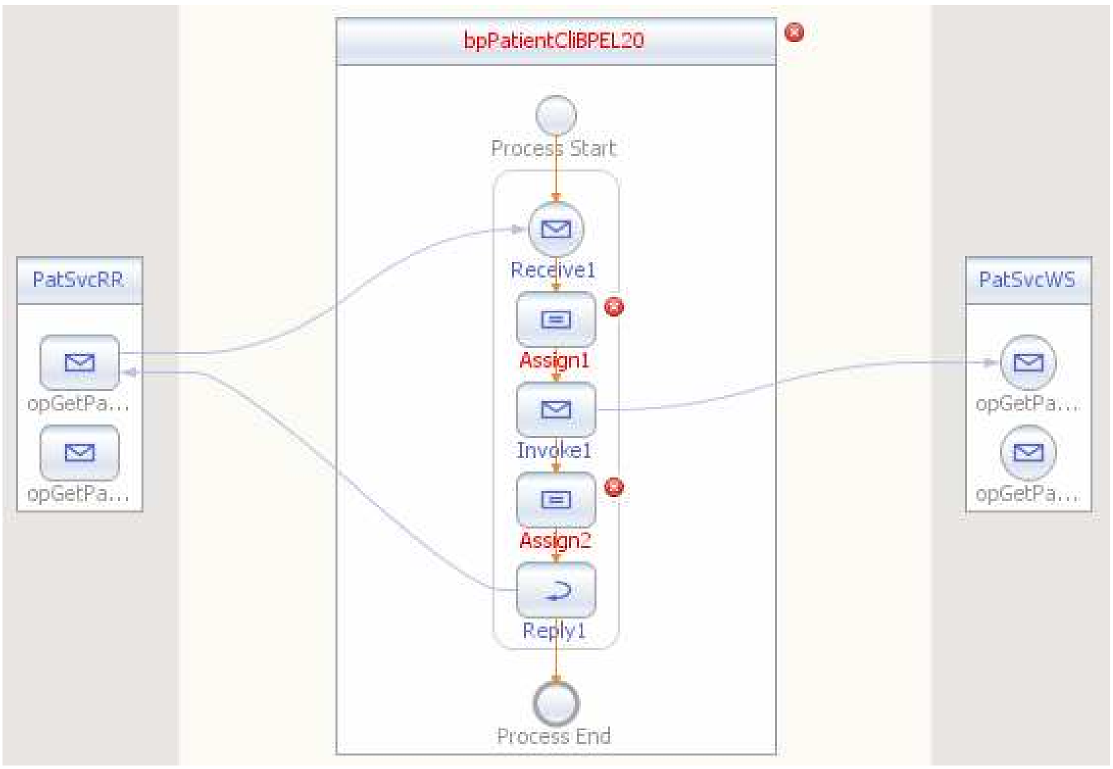


Figure 3-10 Process snapshot with activities and partners connected

In Assign1 activity map request values from the PatDetRR partner's input variable to the PatDetWS partner's input variable and use the BPEL -> GUID function to map a GUID to the SessionToken node. Figure 3-11 illustrates this mapping.

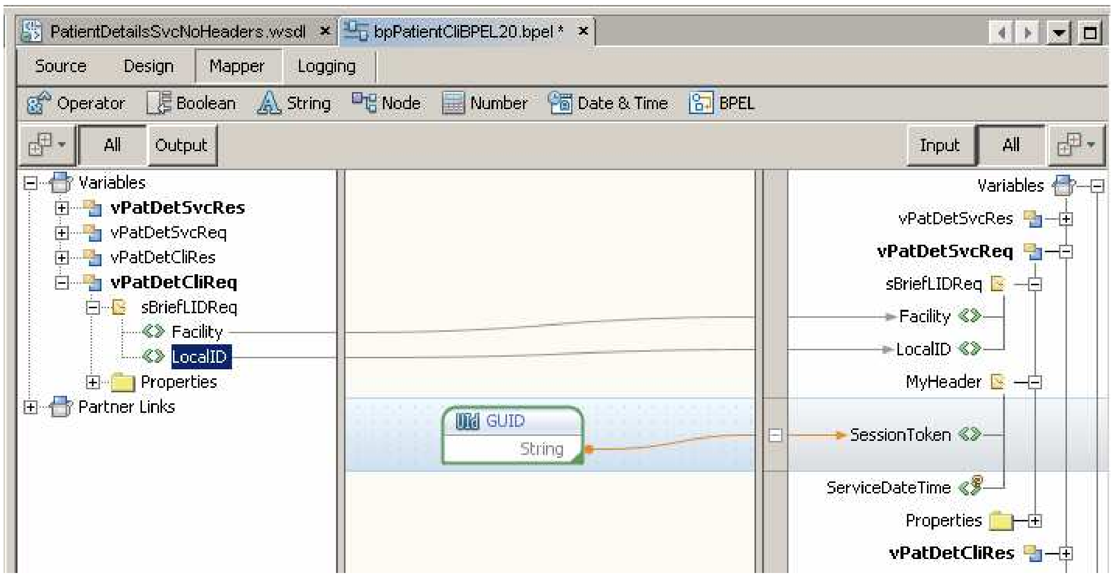


Figure 3-11 Request mapping

In Assign2 activity map service response to the client side and map SessionToken and SessionDateTime to MiddleInitialOrName and Suffix of the client response respectively. Figure 3-12 illustrates this mapping.

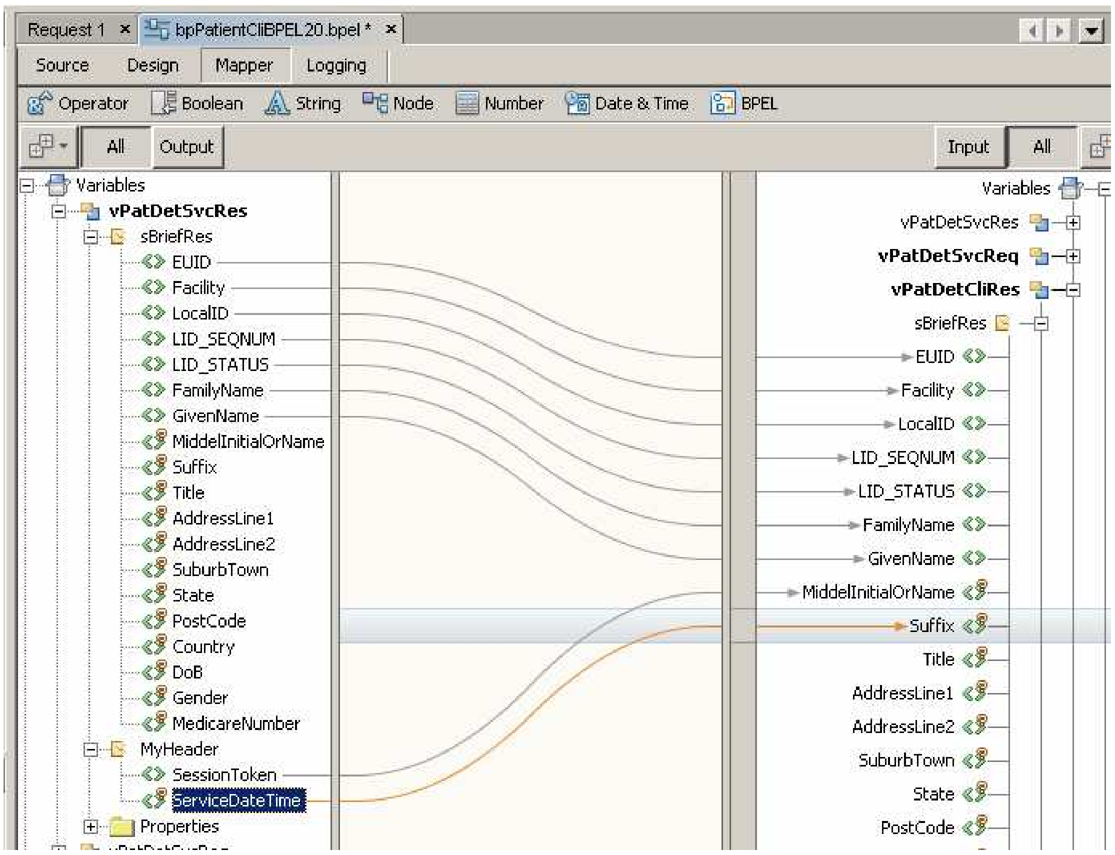
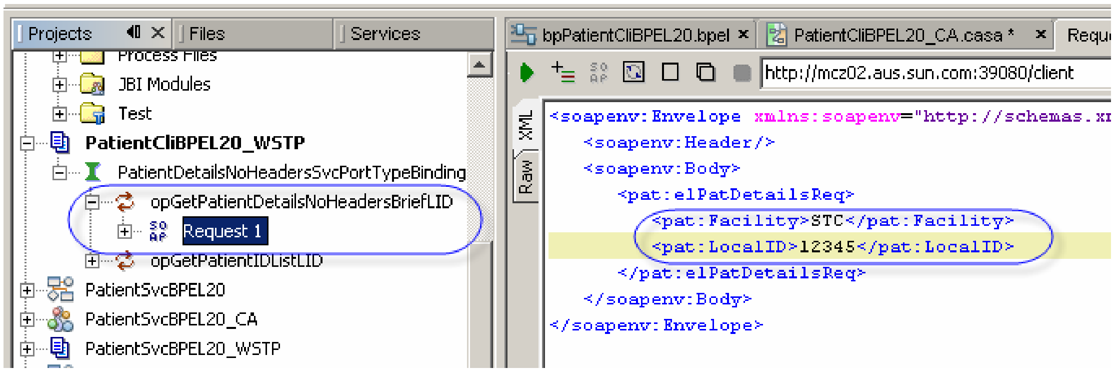


Figure 3-12 Response mapping

Build the project. Create a New -> SOA -> Composite Application project, PatientCliBPEL20_CA, drag the PatientCliBPEL20 project onto the casa map canvas, build and deploy.

Create a New -> Enterprise -> Web Service Testing Project, PatientCliBPEL20_WSTP, using the client-side address, http://localhost:39080/client - the WSDL has that as http://localhost:${HttpDefaultPort}/client?WSDL - replace ${HttpDefaultPort} with the port which which the sun-http-binding is configured.

Open the request under the …PatientDetails… operation, fill in the values and submit. Figure 3-13 illustrates the request.

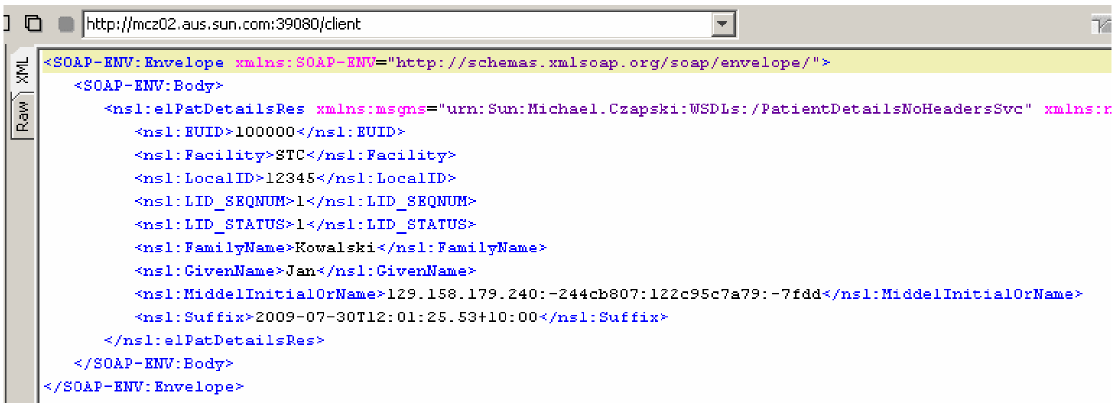


http://mcz02.aus.sun.com:39080/client

```
<soapenv:Envelope xmlns:soapenv="http://schemas.x
   <soapenv:Header/>
   <soapenv:Body>
      <pat:elPatDetailsReq>
         <pat:Facility>STC</pat:Facility>
         <pat:LocalID>12345</pat:LocalID>
      </pat:elPatDetailsReq>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 3-13 Client request

The response comes back as expected, with SessionToken and SessionDateTime in MiddleInitialOrName and Suffix respectively. Figure 3-14 illustrates this response.



```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <nsl:elPatDetailsRes xmlns:msgns="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsNoHeadersSvc" xmlns:r
         <nsl:EUID>100000</nsl:EUID>
         <nsl:Facility>STC</nsl:Facility>
         <nsl:LocalID>12345</nsl:LocalID>
         <nsl:LID_SEQNUM>1</nsl:LID_SEQNUM>
         <nsl:LID_STATUS>1</nsl:LID_STATUS>
         <nsl:FamilyName>Kowalski</nsl:FamilyName>
         <nsl:GivenName>Jan</nsl:GivenName>
         <nsl:MiddelInitialOrName>129.158.179.240:-244cb807:122c95c7a79:-7fdd</nsl:MiddelInitialOrName>
         <nsl:Suffix>2009-07-30T12:01:25.53+10:00</nsl:Suffix>
      </nsl:elPatDetailsRes>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 3-14 Response

With the correct logging settings you will be able to see message exchange in the server.log. To do this, add the following to the GlassFish Application Server's JVM Properties, restart the server and exercise the solution again.

```
-Dcom.sun.xml.ws.transport.http.HttpAdapter.dump=true
-Dcom.sun.xml.ws.transport.http.client.HttpTransportPipe.dump=true
```

Server.log should show messages like:

```
[#|2009-07-30T12:01:25.484+1000|INFO|sun-
appserver9.1|javax.enterprise.system.stream.out|_ThreadID=45;_ThreadName=httpWorkerThr
ead-39080-3;|
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:pat="urn:Sun:Michael.Czapski:XSDs:/PatientDetailsNoHeaders">
   <soapenv:Header/>
   <soapenv:Body>
      <pat:elPatDetailsReq>
         <pat:Facility>STC</pat:Facility>
         <pat:LocalID>12345</pat:LocalID>
      </pat:elPatDetailsReq>
   </soapenv:Body>
</soapenv:Envelope>|#]


[#|2009-07-30T12:01:25.531+1000|INFO|sun-
appserver9.1|javax.enterprise.system.stream.out|_ThreadID=57;_ThreadName=httpWorkerThr
ead-39080-0;|
<?xml version="1.0" ?><SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Header><elMyHeader
xmlns="urn:Sun:Michael.Czapski:XSDs:/PatientDetails"
xmlns:msgns="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsSvc"><ns1:SessionToken
xmlns:ns1="urn:Sun:Michael.Czapski:XSDs:/PatientDetails">129.158.179.240:-
244cb807:122c95c7a79:-7fdd</ns1:SessionToken></elMyHeader></SOAP-ENV:Header><SOAP-
ENV:Body><ns1:elPatDetailsReq
xmlns:msgns="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsSvc"
xmlns:ns1="urn:Sun:Michael.Czapski:XSDs:/PatientDetails"><ns1:Facility>STC</ns1:Facili
ty><ns1:LocalID>12345</ns1:LocalID></ns1:elPatDetailsReq></SOAP-ENV:Body></SOAP-
ENV:Envelope>|#]


[#|2009-07-30T12:01:25.546+1000|INFO|sun-
appserver9.1|javax.enterprise.system.stream.out|_ThreadID=56;_ThreadName=HTTPBC-
OutboundReceiver-5;Context=PatientCliBPEL20_CA-sun-http-binding-
{urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsSvc}opGetPatientDetailsBriefLID;|
<?xml version="1.0" ?><SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Header><elMyHeader
xmlns="urn:Sun:Michael.Czapski:XSDs:/PatientDetails"
xmlns:msgns="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsSvc"><ns1:SessionToken
xmlns:ns1="urn:Sun:Michael.Czapski:XSDs:/PatientDetails">129.158.179.240:-
244cb807:122c95c7a79:-7fdd</ns1:SessionToken><ns1:ServiceDateTime
xmlns:ns1="urn:Sun:Michael.Czapski:XSDs:/PatientDetails">2009-07-
30T12:01:25.53+10:00</ns1:ServiceDateTime></elMyHeader></SOAP-ENV:Header><SOAP-
ENV:Body><ns1:elPatDetailsRes
xmlns:msgns="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsSvc"
xmlns:ns1="urn:Sun:Michael.Czapski:XSDs:/PatientDetails"><ns1:EUID>100000</ns1:EUID><n
s1:Facility>STC</ns1:Facility><ns1:LocalID>12345</ns1:LocalID><ns1:LID_SEQNUM>1</ns1:L
ID_SEQNUM><ns1:LID_STATUS>1</ns1:LID_STATUS><ns1:FamilyName>Kowalski</ns1:FamilyName><
ns1:GivenName>Jan</ns1:GivenName></ns1:elPatDetailsRes></SOAP-ENV:Body></SOAP-
ENV:Envelope>|#]


[#|2009-07-30T12:01:25.578+1000|INFO|sun-
appserver9.1|javax.enterprise.system.stream.out|_ThreadID=47;_ThreadName=HTTPBC-JAXWS-
Engine-3;|
<?xml version="1.0" ?><SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Body><ns1:elPatDetailsRes
xmlns:msgns="urn:Sun:Michael.Czapski:WSDLs:/PatientDetailsNoHeadersSvc"
xmlns:ns1="urn:Sun:Michael.Czapski:XSDs:/PatientDetailsNoHeaders"><ns1:EUID>100000</ns
1:EUID><ns1:Facility>STC</ns1:Facility><ns1:LocalID>12345</ns1:LocalID><ns1:LID_SEQNUM
>1</ns1:LID_SEQNUM><ns1:LID_STATUS>1</ns1:LID_STATUS><ns1:FamilyName>Kowalski</ns1:Fam
ilyName><ns1:GivenName>Jan</ns1:GivenName><ns1:MiddelInitialOrName>129.158.179.240:-
244cb807:122c95c7a79:-7fdd</ns1:MiddelInitialOrName><ns1:Suffix>2009-07-
30T12:01:25.53+10:00</ns1:Suffix></ns1:elPatDetailsRes></SOAP-ENV:Body></SOAP-
ENV:Envelope>|#]
```

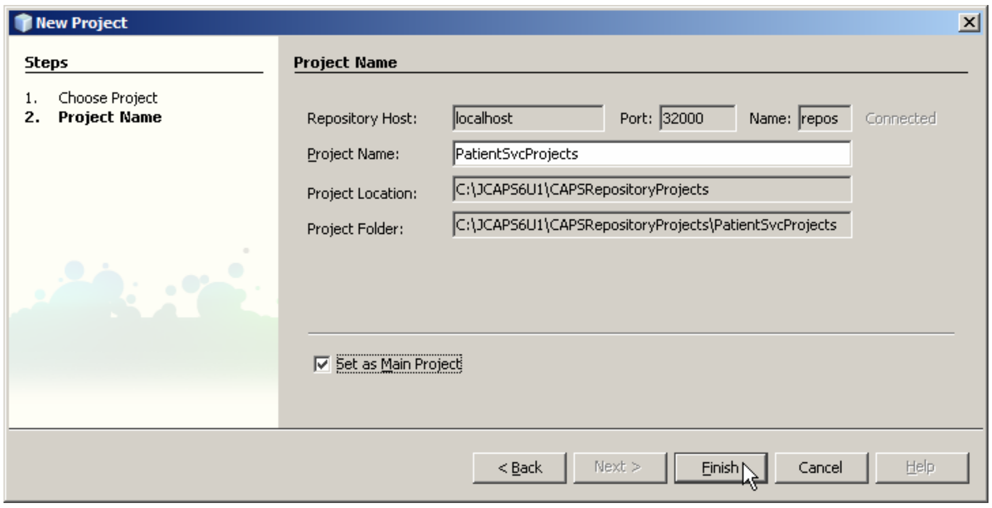
The service and the client work. The headers are passed back and forth and their node values are gettable and settable from BPEL 2.0.

4 Implement and test the service – eInsight

This section assumes that you have the Java CAPS 6 Repository environment configured. In particular you have a Java CAPS Environment consisting of at least 1 logical host with at least 1 Sun Java System Application Server, and the UDDI Registry container configured.

Make sure you are connected to the Repository.

Create a top-level New Project -> CAPS -> ESB -> CAPS Repository-based Project, named PatientSvcProjects. Figure 4-1 illustrates a step in the process.


New Project

Steps
1. Choose Project
2. **Project Name**

Project Name

Repository Host: localhost Port: 32000 Name: repos Connected
Project Name: PatientSvcProjects
Project Location: C:\JCAPS6U1\CAPSRepositoryProjects
Project Folder: C:\JCAPS6U1\CAPSRepositoryProjects\PatientSvcProjects

☑ Set as Main Project

< Back | Next > | Finish | Cancel | Help

Figure 4-1 Name the CAPS Repository project

This project will contain the service implementation project. It was intended to also contains the client implementation but I run out of motivation and did not develop it.

Right-click the name of the new CAPS project and choose New -> Project, as shown in Figure 4-2. Name the subproject PatientSvc.

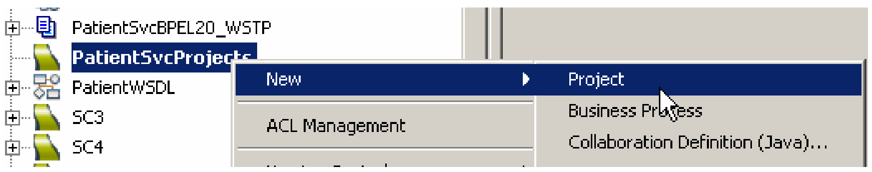
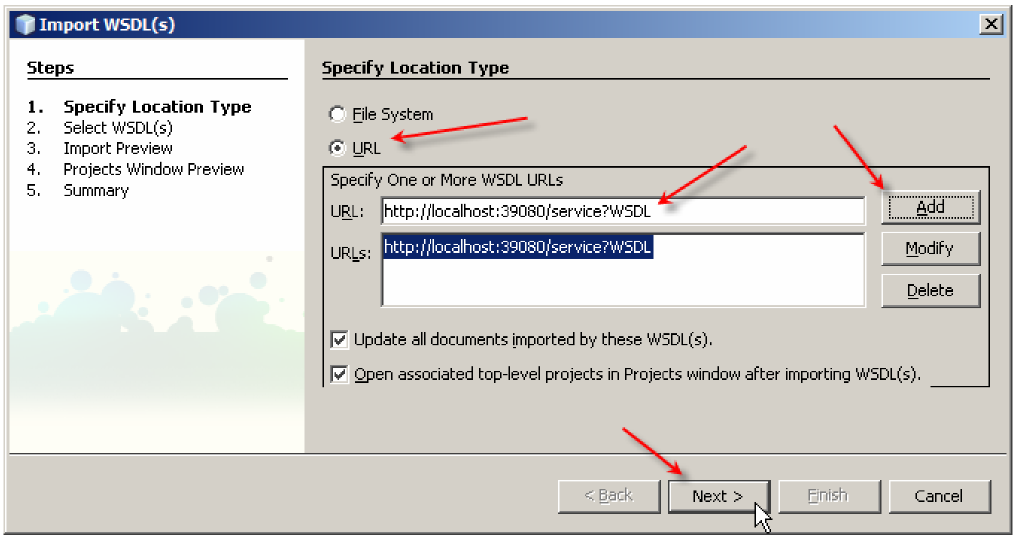


Figure 4-2 Creating a subproject

There are a couple of ways in which to add a WSDL to a Java CAPS Repository project. We will use the WSDL from a URL method. Assuming the PatientSvc JBI project is still deployed and the WSDL URL is http://localhost:39080/service?WSDL, right-click the name of the subproject, choose Import -> Web Service Definition …, select the URL radio button, enter/paste the WSDL URL into the URL: data entry box, click Add and click Next. Figure 4-3 illustrates this.



URL: http://localhost:39080/service?WSDL

URLs: http://localhost:39080/service?WSDL

☑ Update all documents imported by these WSDL(s).

☑ Open associated top-level projects in Projects window after importing WSDL(s).

Figure 4-3 Providing WSDL URL

Click Next, Next, Finish, ignoring a warning about invalid "?" character.

The project should now look like that shown in Figure 4-4.

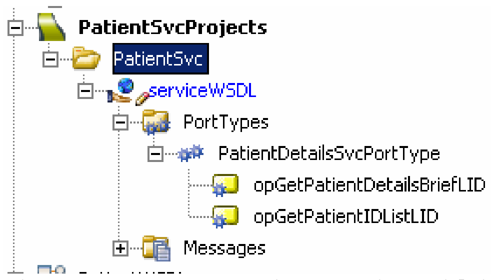


Figure 4-4 CAPS Repository project with imported WSDL

Create a New -> Business Process. Name the process bpPatientSvc. Drag the opGetPatientDetaislBriefLID operation onto the BP canvas and choose Implement Business Operation (Server Mode), as shown in Figure 4-5.

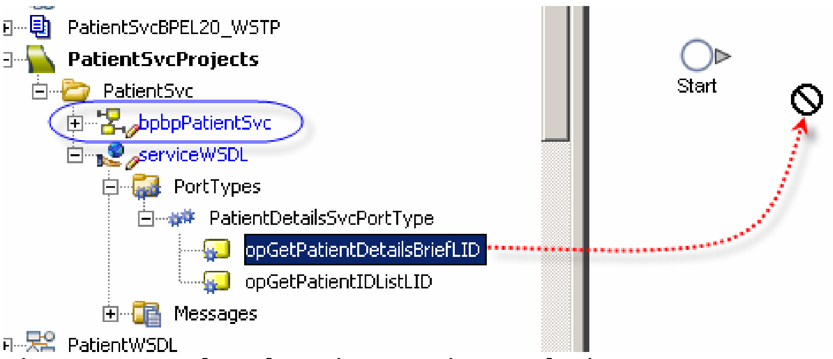


Figure 4-5 Drag the web service operation onto business process canvas

Connect activities, add business rules and map. Example is shown in Figure 4-6.

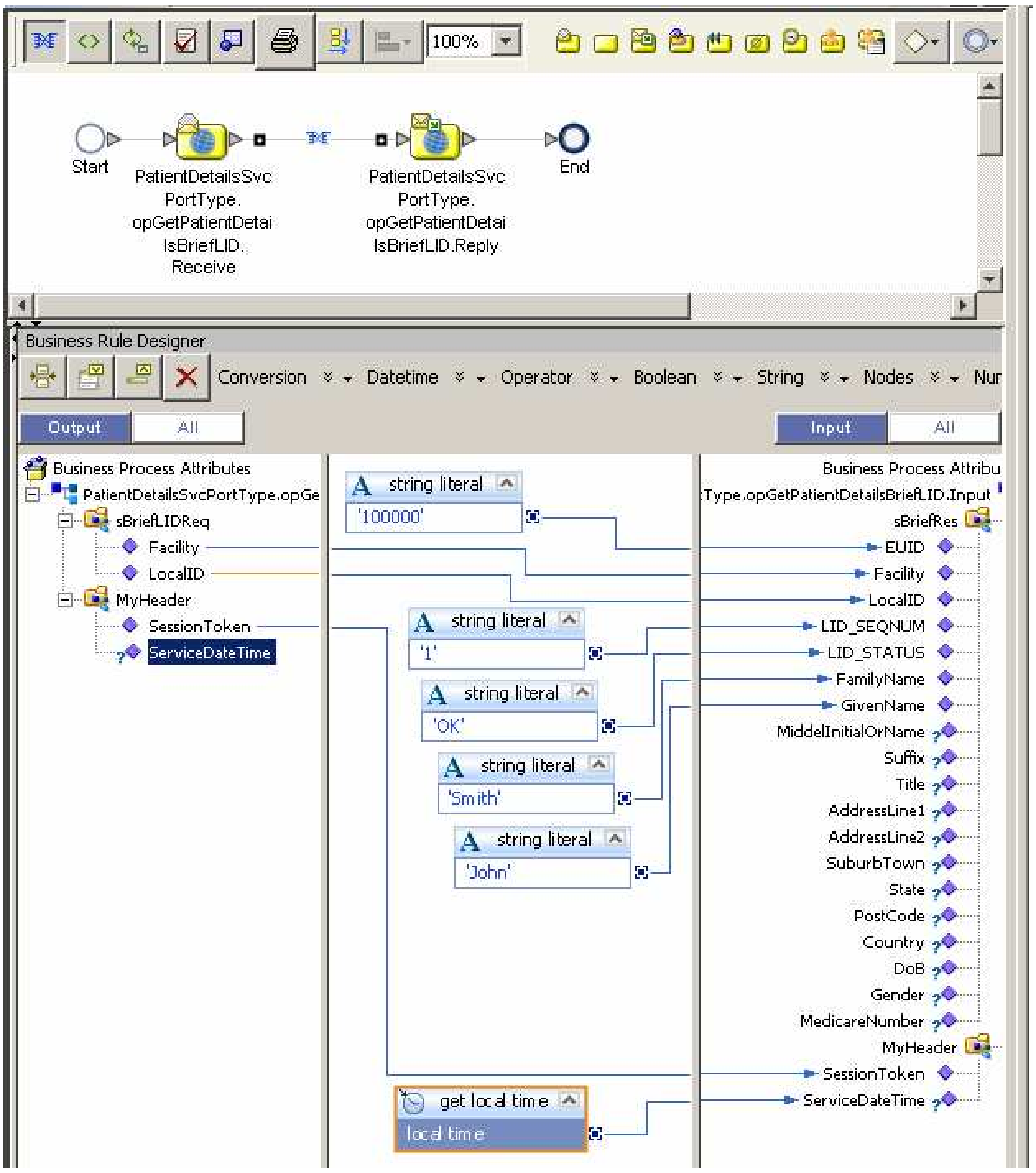


Figure 4-6 Map Request to Response and populate mandatory nodes

Create a connectivity map, cmPatientSvc, and configure. Figure 4-7 shows a completed connectivity map.

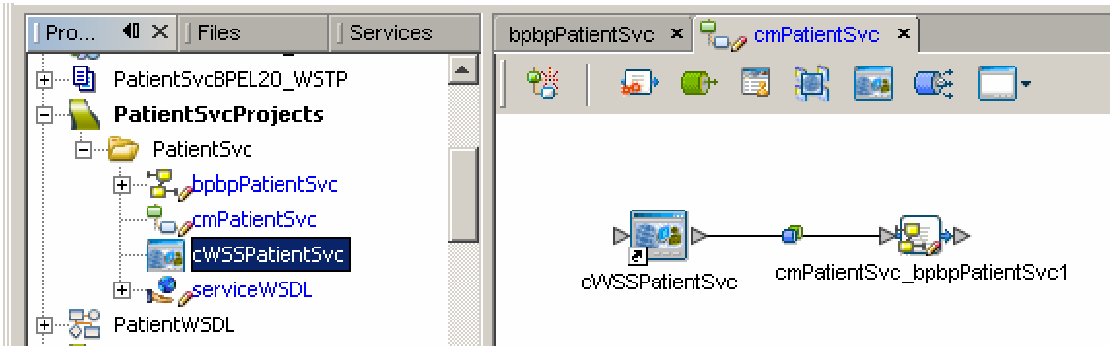


Figure 4-7 Connectivity map

Switch to Services tab, expand CAPS Environment, expand an environment (you have one, don't you), add a New -> SOAP/HTTP External System …, call it WSSPatientSvc. Set the Servlet Context to PatientSvc and leave all other properties at default.

Switch back to the Projects tab, right-click on the name of the subproject and create a deployment profile, dpPatientSvc. Map external systems to their corresponding containers, build and deploy. Figure 4-8 shown a snapshot of the process.

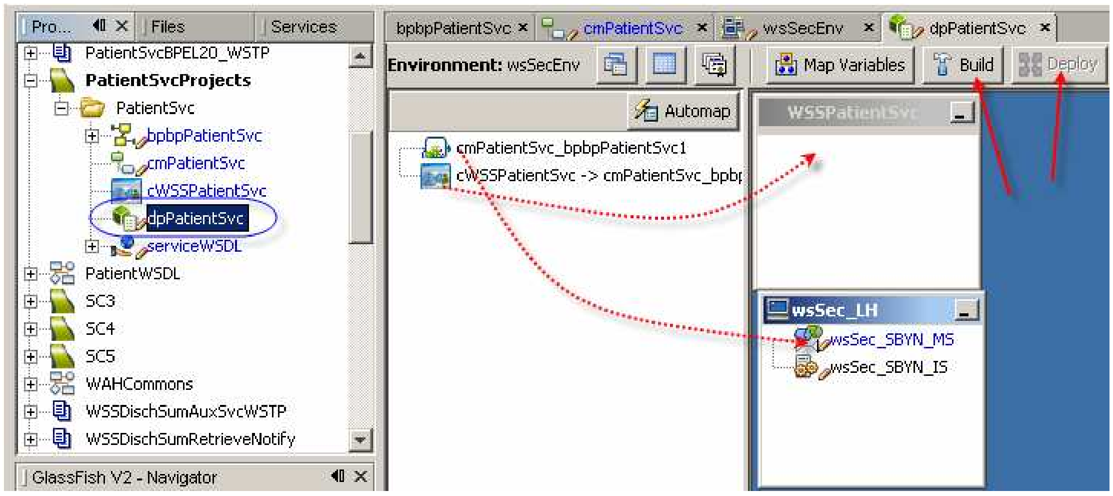
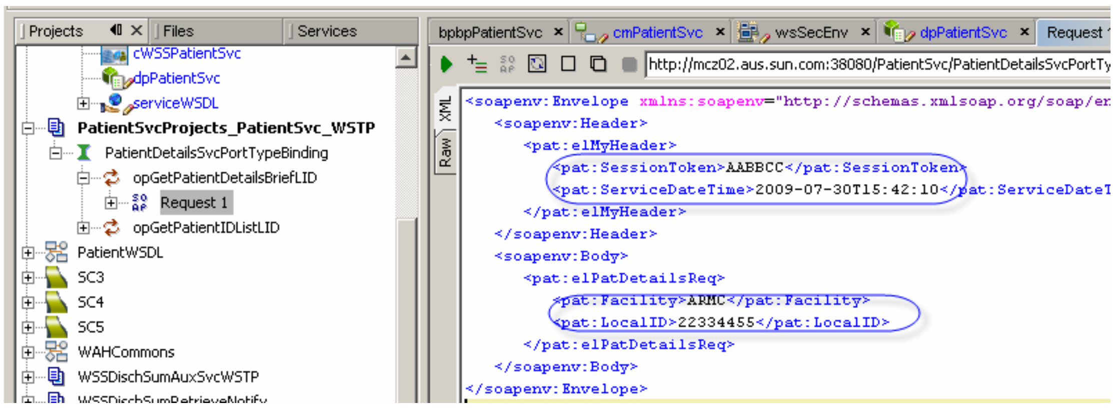


Figure 4-8 Assigning connectors to containers

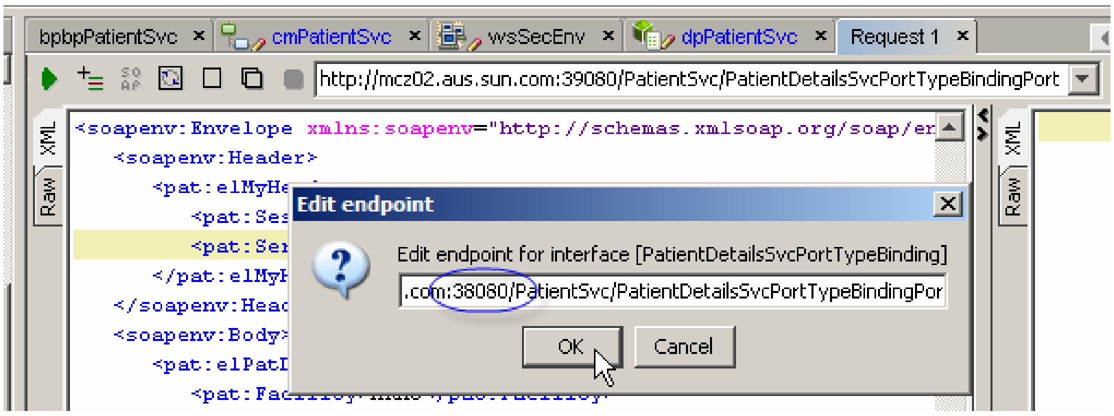
The project is deployed. Use the Java CAPS UDDI Browser to determine the WSDL URL for this service. For me this is:
http://localhost:34848/CAPSUDDI/displayWSDL?wsdlname=/wsSecEnv/PatientSvc/bpbpPatientSvc/serviceWSDL741303108.wsdl

Construct and run a New Project -> Enterprise -> Web Service Testing Project. Name the project PatientSvcProjects_PatientSvc_WSTP and provide the URL obtained from the UDDI registry. Modify the request for the opGetPatientDetaislBriefLID operation by providing reasonable values, modify the service URL to provide the port appropriate for your installation and submit the request. Figure 4-9 shows the project and the request. Figure 4-10 shows a dialogue in which URL can be modified.



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/en
   <soapenv:Header>
      <pat:elMyHeader>
         <pat:SessionToken>AABBCC</pat:SessionToken>
         <pat:ServiceDateTime>2009-07-30T15:42:10</pat:ServiceDateT
      </pat:elMyHeader>
   </soapenv:Header>
   <soapenv:Body>
      <pat:elPatDetailsReq>
         <pat:Facility>ARMC</pat:Facility>
         <pat:LocalID>22334455</pat:LocalID>
      </pat:elPatDetailsReq>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 4-9 SOAP Request



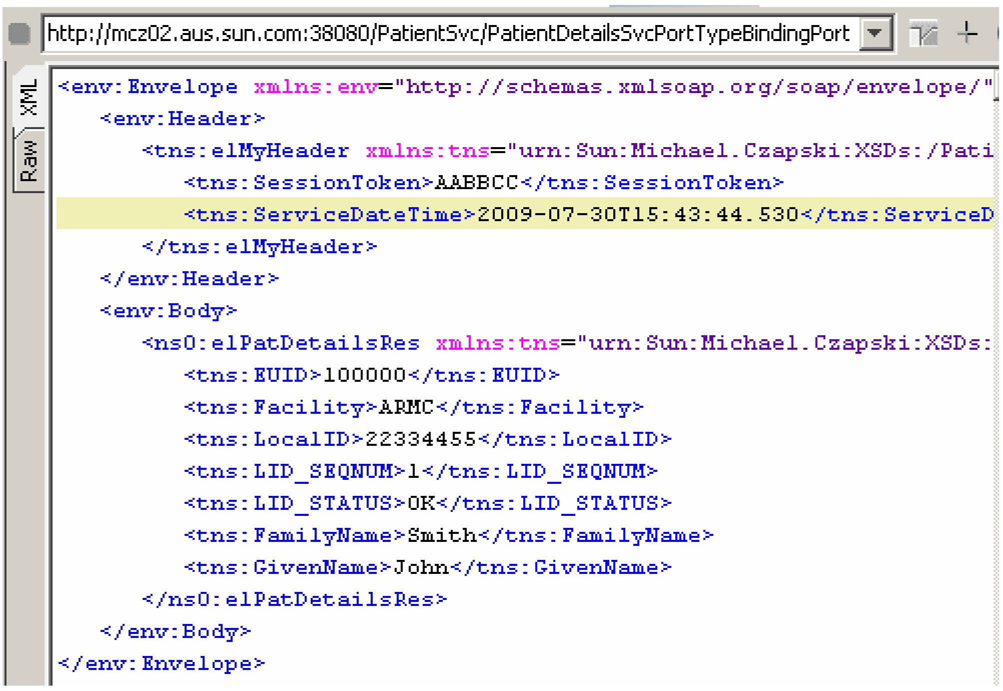
Edit endpoint

Edit endpoint for interface [PatientDetailsSvcPortTypeBinding]

.com:38080/PatientSvc/PatientDetailsSvcPortTypeBindingPor

Figure 4-10 Ensure the port is correct

The response is as expected. Figure 4-11 shows the response.



http://mcz02.aus.sun.com:38080/PatientSvc/PatientDetailsSvcPortTypeBindingPort

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
   <env:Header>
      <tns:elMyHeader xmlns:tns="urn:Sun:Michael.Czapski:XSDs:/Pati
         <tns:SessionToken>AABBCC</tns:SessionToken>
         <tns:ServiceDateTime>2009-07-30T15:43:44.530</tns:ServiceD
      </tns:elMyHeader>
   </env:Header>
   <env:Body>
      <ns0:elPatDetailsRes xmlns:tns="urn:Sun:Michael.Czapski:XSDs:
         <tns:EUID>100000</tns:EUID>
         <tns:Facility>ARMC</tns:Facility>
         <tns:LocalID>22334455</tns:LocalID>
         <tns:LID_SEQNUM>1</tns:LID_SEQNUM>
         <tns:LID_STATUS>OK</tns:LID_STATUS>
         <tns:FamilyName>Smith</tns:FamilyName>
         <tns:GivenName>John</tns:GivenName>
      </ns0:elPatDetailsRes>
   </env:Body>
</env:Envelope>
```

Figure 4-11 Service Response

Note the custom SOA Header in the request and response.

Building a client to exercise this service is as easy as building the service so I will not do this. Feel free to correct this omission.

5 Summary

This document discussed process of creating modifying a “regular” WSDL to support custom SOAP Headers, and building, deploying and exercising a BPEL SE solution and a partial eInsight solution that used these headers for the conveyance of out-of-bound information in web services based messaging.