



PEtALS SOAP Use Cases User's Guide

This document explain how to use the SOAP usecases in the PEtALS environment.

EBM WebSourcing
Christophe HAMERLING <christophe.hamerling@ebmwebsourcing.com>
- August 2008 -



(CC) EBM WebSourcing - This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/>



Table of Contents

PETALS-SOAP-USECASES	4
1. Getting started	5
1.1. Prerequisites	5
1.2. Generating usecases from sources	5
1.3. Getting usecases	5
1.4. Launching usecases	6
2. Use cases	7
2.1. Expose a JBI service as Web Service	7
2.1.1. Setting up the platform	7
2.1.2. Usage	8
2.2. Use PEtALS as Web Service Proxy	8
2.2.1. Setting up the platform	9
2.2.2. Usage	9
2.3. Handling Web Service notifications	11
2.3.1. Setting up the platform	12
2.3.2. Usage	12
2.4. Handling SOAP attachments	15
2.4.1. Setting up the platform	16
2.4.2. Usage	16
2.5. REST Services	17
2.5.1. Setting up the platform	17
2.5.2. Usage	17
2.6. Securing access to an exposed Web Service	19
2.6.1. Setting up the platform	19
2.6.2. Usage	19

List of Figures

2.1. Exposing a JBI service as Web Service	7
2.2. SOAP Hello JBI service call with SOAPUI	8
2.3. Proxying Web Service with PEtALS	9
2.4. Create the SOAP request in SOAPUI	10
2.5. SOAP message response in SOAPUI	11
2.6. Producing Web Service notification with PEtALS	12
2.7. Publish a message in the topic with the sample client component	14
2.8. Handling SOAP attachments with PEtALS	16
2.9. Calling the REST Yahoo Service with SOAPUI	17
2.10. Calling the Helloworld JBI Service in REST mode	19
2.11. SOAP security unsecure call with SOAPUI	20
2.12. SOAP security secure call with SOAPUI	20

PETALS-SOAP-USECASES

This document explains how to use the SOAP usecases provided with PEtALS. It describes simple scenarii which can be used to better understand the JBI and PEtALS ESB philosophy...



Note

These usecases have been tested on PEtALS Standalone and Quickstart Distribution v2.2.

Chapter 1. Getting started

1.1. Prerequisites

To use these usecases, you must at least get the following tools :

- PEtALS : The OW2 open source ESB (<http://petals.ow2.org>).
- Apache ant: Used to install components, deploy service assemblies and lauch sample clients (<http://ant.apache.org>).
- Apache Tomcat: Used to install Web Services which will be called by PEtALS services (<http://tomcat.apache.org>).
- SOAPUI : A SOAP application to test Web Services (<http://www.soapui.org>)

If you want to build usecases from sources, you also need to use Maven (<http://maven.apache.org>).

The current document is Linux user oriented, but it can be easily understood by Windows users.

1.2. Generating usecases from sources

The usecases are located under the <PETALS_SRC>/petals-demos/petals-usecases directory, where <PETALS_SRC> is the root directory where you have checked out the sources of the project.

You can generate the usecases with maven as usual :

```
cd <PETALS_SRC>/trunk/petals-demos/petals-usecases
```

```
mvn clean; mvn
```

To build an assembly of a usecase (a packaged directory with all the required artifacts), you must go into the usecase directory and generate it :

```
cd petals-<usecase>
```

```
mvn clean install package assembly:assembly
```

The usecase with all the needed components, service assemblies, clients and scripts are available under the target directory of the usecase.

The usecases are configured to work with local engines (PEtALS, tomcat), if you want to test the usecases in a distributed environment, you must change some parameters on the services units in order to have a proper configuration.

1.3. Getting usecases

Usecases are available for download as archives from the PEtALS forge download page (http://forge.objectweb.org/project/showfiles.php?group_id=213 under the petals-usecases section).

The usecase structure is generally:

```
build.xml
taskdef.properties

client/
  petals-*-client.jar
  *.jar

deployables/
  petals-bc-*.zip
  sa-*.zip
```

1.4. Launching usecases

Each usecase comes with its own Ant script.

You can edit the build.xml file to set the \$host, \$port \$username and \$password properties of your PEtALS JMX-RMI connector (*localhost*, *7700*, *petals*, *petals* are default ones).

To install/deploy artifacts, generally you just have to call the install task (or just call Ant with no argument):

ant install

If a usecase comes with a client application, you will have to call the client task to run the program:

ant client

To clean the platform (stop and uninstall all components and service assemblies), you can call the clean task:

ant clean



Note

Special calls are detailed for each usecase.

Chapter 2. Use cases

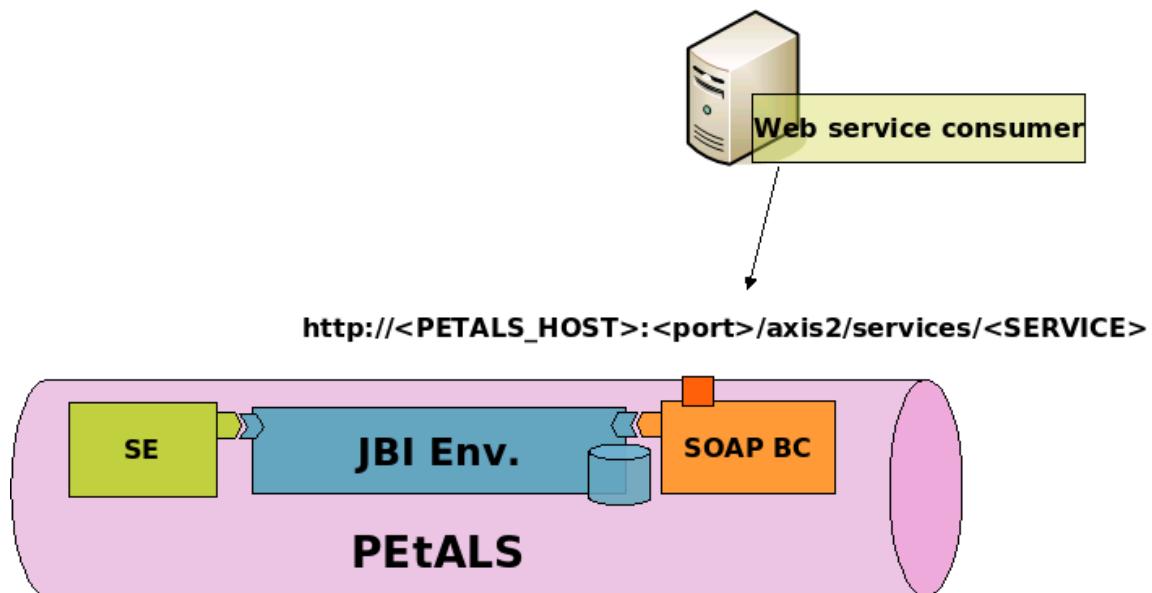
2.1. Expose a JBI service as Web Service

This use case demonstrates how to expose a JBI service as Web Service which can be called from external web service consumers. The current usecase expose an helloworld service provided by the petals-sample-helloworld component. This usecase can be extended to other JBI services.

To be able to consume to handle incoming Web Services calls, we need to use the SOAP binding component (petals-bc-soap) and deploy service assembly (SA) to configure it.

The JBI oriented representation of this use case is :

Figure 2.1. Exposing a JBI service as Web Service



2.1.1. Setting up the platform

1. Start PEtALS (if not already started)
2. Install components (petals-bc-soap, petals-sample-helloworld), deploy service assembly and start these artifacts. This can be done with the Ant script:

ant

The deployed Service Assembly contains :

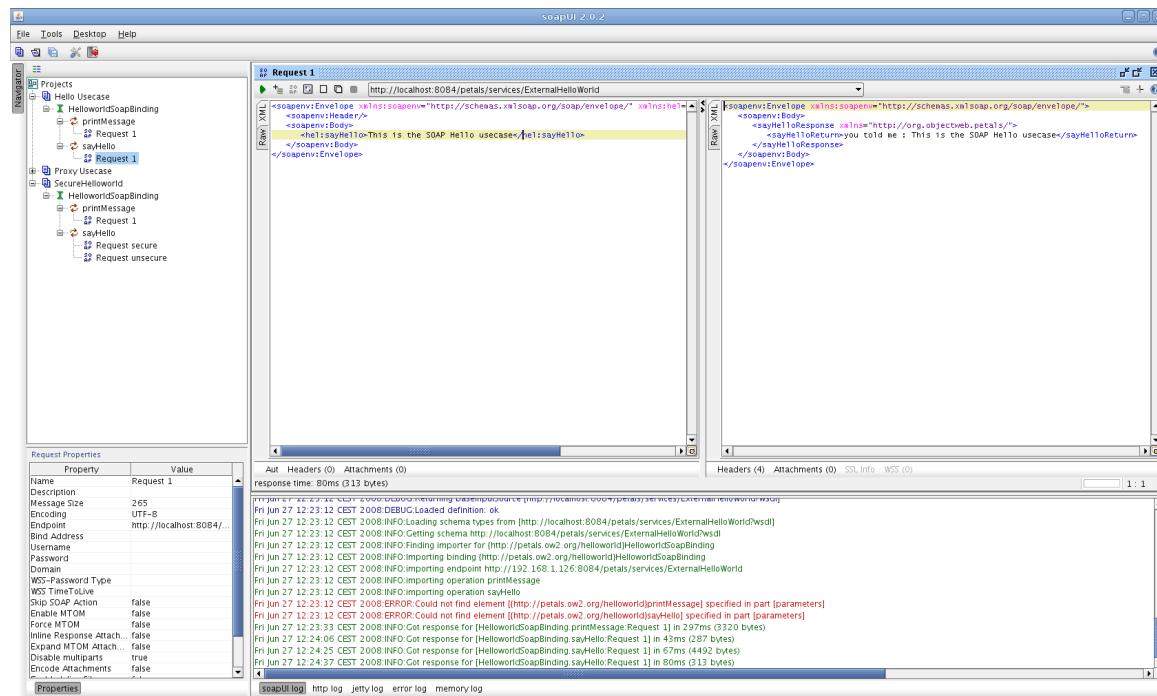
1. A service unit which will be deployed on the petals-sample-helloworld to provide fixed endpoint/service/interface values.
2. A service unit which will be deployed on the petals-bc-soap to consume the provided helloworld service. When consuming a JBI service with the SOAP binding component, a Web Service is created and is exposed to the outside (please refer to the petals-bc-soap documentation for more details).

A sample client is available to send a SOAP message to the exposed Web Service, it calls the *sayHello* operation on the exposed Web Service. The SOAP binding component will transform and forward the incoming SOAP message to the JBI service (helloworld).

2.1.2. Usage

A SOAPUI project is provided in the soapui folder of the usecase. Import it into SOAPUI and invoke the sayHello operation like in the following screenshot.

Figure 2.2. SOAP Hello JBI service call with SOAPUI



You should see a response on the right side like the following one :

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <sayHelloResponse xmlns="http://org.objectweb.petals/">
      <sayHelloReturn>you told me : This is the SOAP Hello usecase</sayHelloReturn>
    </sayHelloResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

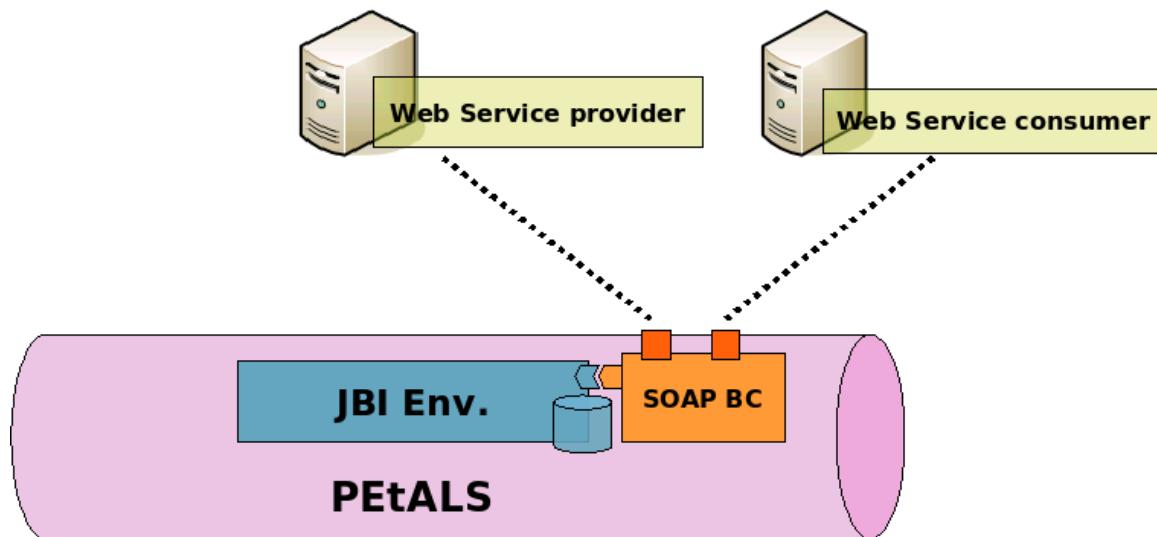
2.2. Use PEtALS as Web Service Proxy

This use case demonstrate how you can PEtALS to proxying Web Services.

The current usecase consumes an external Web Service and expose it as a JBI service in the bus. This provided service is consumed by an JBI service and exposed to external Web Services clients.

The JBI oriented representation of this usecase is :

Figure 2.3. Proxying Web Service with PEtALS



2.2.1. Setting up the platform

1. Install the Web Archive providing the echo Web Service (web-service-sample.war) into Tomcat.

Depending on your Tomcat container, the Web Service should be available on `http://<HOST>:<PORT>/web-service-sample/services/SimpleServiceImpl`

In the current usecase, we suppose that Tomcat is launched on the 8080 port of the localhost. Change these values if needed.

2. Start PEtALS (if not already started...)
3. Install components (petals-bc-soap), deploy service assembly and start these artifacts. This can be done with the Ant script:

ant

The deployed service assembly contains:

1. A service unit which will be deployed on the petals-bc-soap to consume an external Web Service and then provide it as a JBI service.
2. A service unit which will be deployed on the petals-bc-soap to consume the JBI service provided by the previous deployed service unit.

2.2.2. Usage

We use SOAPUI as Web Service client. The SOAPUI project is available under the soapui folder. Import it in SOAPUI with 'File>Import project'.

Note that the port specified is 8084, which is the port used by the PEtALS SOAP component, not the tomcat one; SOAPUI deals with the WSDL from the proxy web service, not the end targeted one.

This will create the SOAP payload, you just have to put a message if you want. For example :

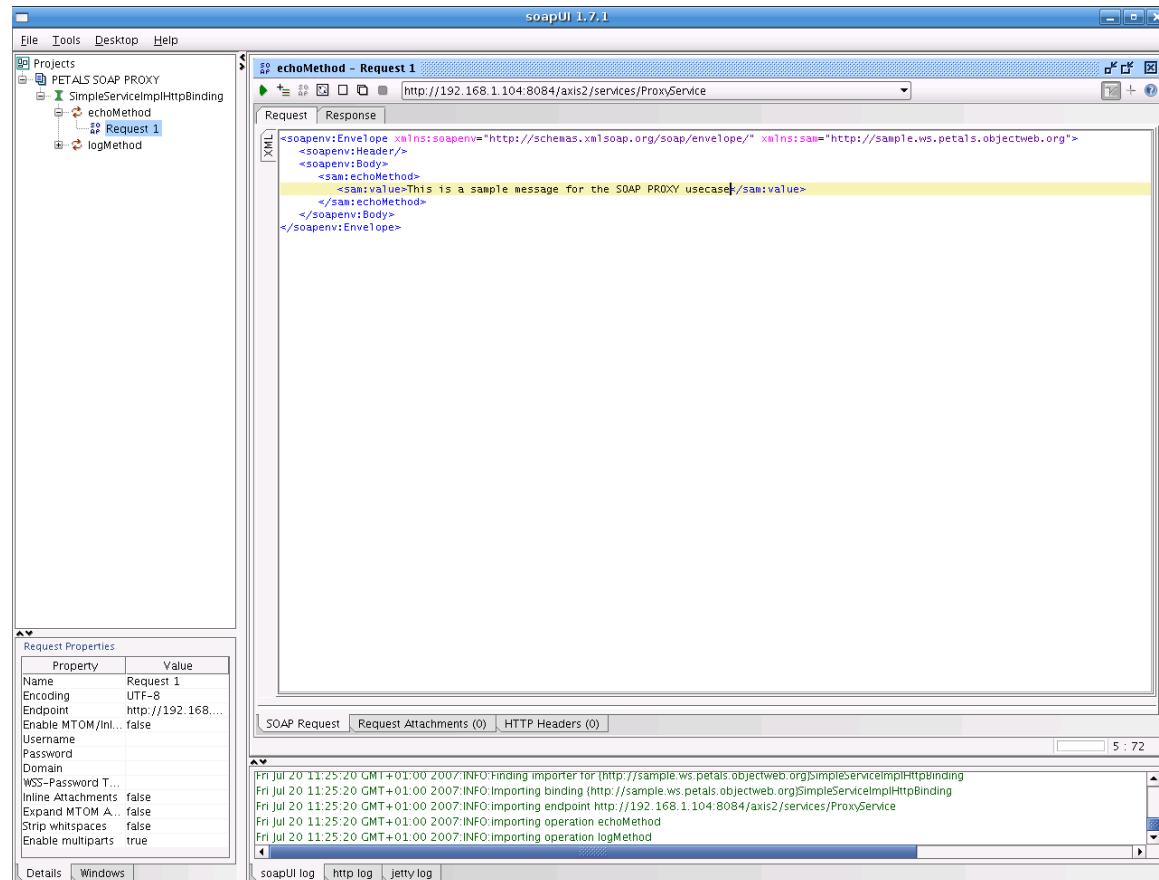
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:sam="http://sample.ws.petals.objectweb.org">
```

```

<soapenv:Header/>
<soapenv:Body>
  <sam:echoMethod>
    <sam:value>This is a sample message for the SOAP PROXY usecase</sam:value>
  </sam:echoMethod>
</soapenv:Body>
</soapenv:Envelope>

```

Figure 2.4. Create the SOAP request in SOAPUI



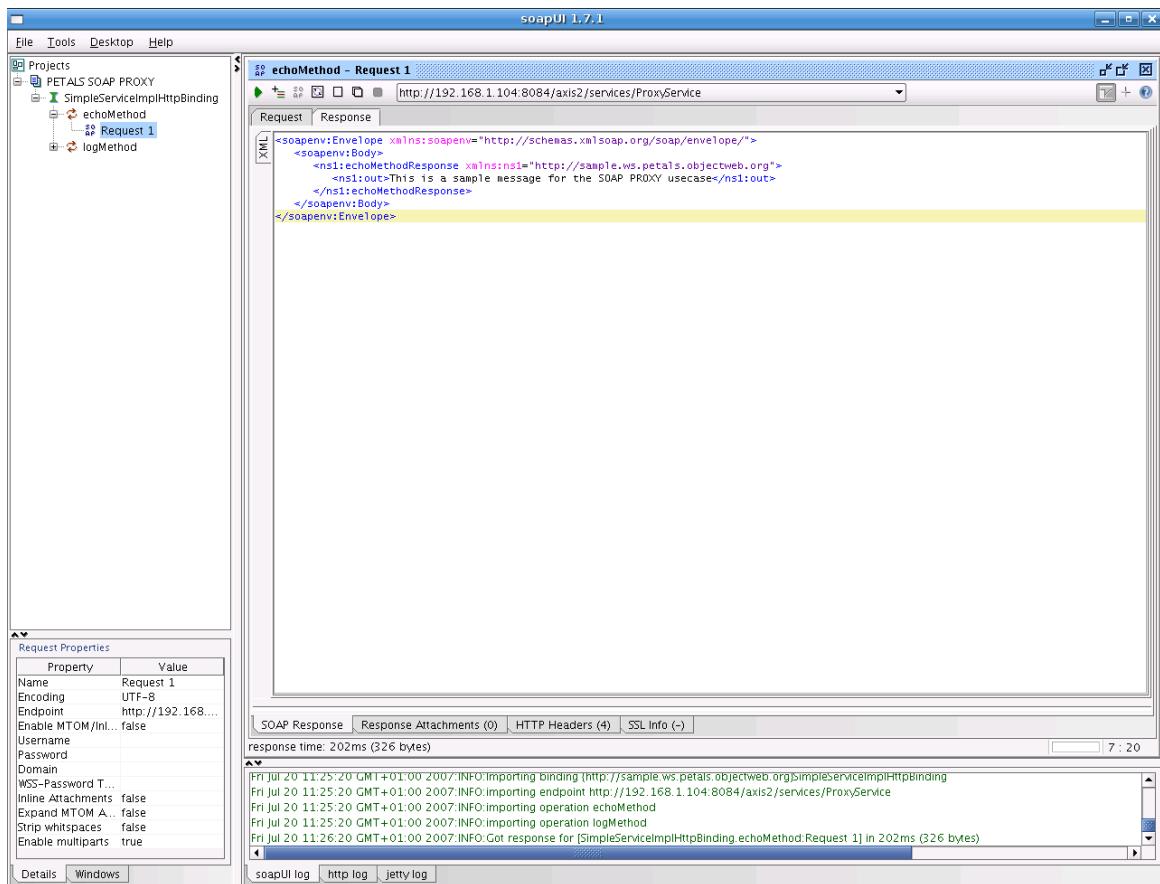
The SOAP request message is :

Send the SOAP message, the SOAP response should be like :

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns1:echoMethodResponse xmlns:ns1="http://sample.ws.petals.objectweb.org">
      <ns1:out>This is a sample message for the SOAP PROXY usecase</ns1:out>
    </ns1:echoMethodResponse>
  </soap:Body>
</soap:Envelope>

```

Figure 2.5. SOAP message response in SOAPUI

The message transits like :

1. SOAP message is sent to PEtALS Web Service
2. The petals-bc-soap translates the SOAP message into a JBI one
3. The JBI message is sent to the Web Service provider endpoint
4. The Web Service provider endpoint translates the SOAP message into a JBI message
5. The SOAP message is sent to the external Web Service hosted on the J2EE container
6. The external Web Service handles the incoming message, process its service and sends a response back
7. The SOAP message transits into the JBI environment and response is sent to the client



Note

You can also use the basic java client provided in the client directory.

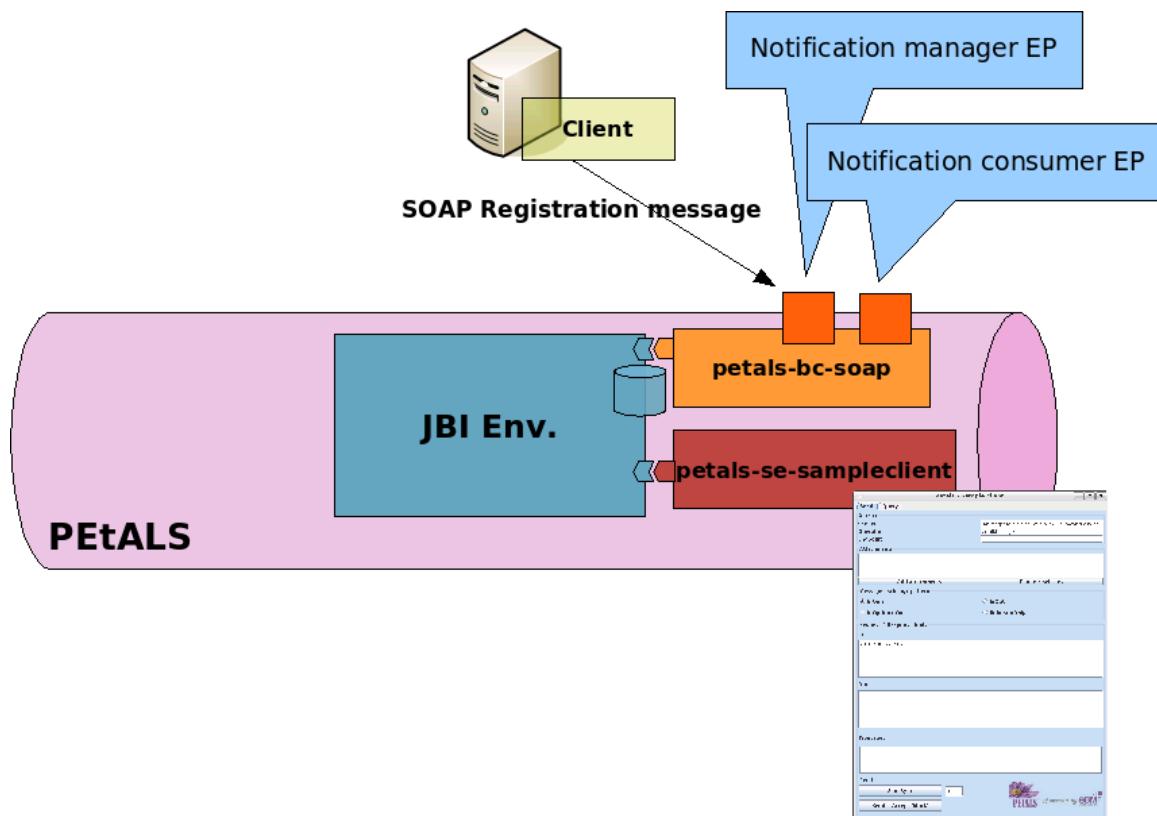
2.3. Handling Web Service notifications

This usecase explain how you can use PEtALS as Web Service notifications producer.

The current usecase provides a Web Service notification mechanisms so that external Web Service notification consumers can subscribe to notification. To get more details on this feature, please refer to the petals-bc-soap documentation.

The JBI oriented representation of this usecase is :

Figure 2.6. Producing Web Service notification with PEtALS



2.3.1. Setting up the platform

1. Start PEtALS (if not already started)
2. Install components (petals-bc-soap, petals-sample-client), deploy service assembly and start these artifacts. This can be done with the Ant script:

ant

The deployed Service Assembly contains :

1. A service unit which will be deployed on the petals-bc-soap to create a Web Service Topic (provides an endpoint).
2. A service unit which will be deployed on the petals-bc-soap to consume the file transfer activated endpoint : Expose a JBI service as external Web Service

A sample client is also provided and is used to create notification subscriptions.

2.3.2. Usage

On the user side, you must :

1. Launch the client which will register notification consumers into the Web Service Notification manager (hosted on the SOAP binding component). This client also launches a mini webserver which will receive Web Service Notifications on port 9999. The server will write the notifications it receives on the standard output. To launch these client, simply type:

ant client

The client registers an URL which is the one activated by a Service Unit on the petals-bc-soap. All the notification messages will be sent to this URL. You should see messages like this in your consoles :

On the client side :

```
[echo] Launching client...
[java] 2007-08-20 15:24:57.488::INFO: Logging to STDERR via org.mortbay.log.StderrLog
[java] 2007-08-20 15:24:57.582::INFO: jetty-6.1.4
[java] 2007-08-20 15:24:58.667::INFO: Started SocketConnector@0.0.0.0:9999
[java] Subscribing to topic TopicSample on WSN producer EPR :
http://localhost:8084/wsn/producer
[java] Notification will be sent to EPR : http://localhost:9999/notif/Consumer
[java] Waiting to complete, type 'q' to quit application...
```

On the PEtALS console, you should have somthinig like :

```
[petals.container.components.petals-bc-soap] Receiving a new request on WSN subscription service
```

2. As JBI service client, we use the sample client to send a JBI message to the topic activated service. Select the topic activated endpoint with the sample client component, and send an 'InOnly' message:

Figure 2.7. Publish a message in the topic with the sample client component

1. The message is received on the SOAP component side, the component analyzes it and detects that it is a notification message
2. The message is published to the linked Web Service topic
3. The notification manager sends the message to all the Web Service notification consumers
4. The notification consumer receives a notification message

You should see the following message on the client side :

```
[java] ### RECEIVING REQUEST :
[java] --MIMEBoundaryurn_uuid_8DB38D8DB2DEF5D9DF1187620267293
[java] Content-Type: application/xop+xml; charset=UTF-8; type="text/xml"
[java] Content-Transfer-Encoding: binary
[java] Content-ID: <0.urn:uuid:8DB38D8DB2DEF5D9DF1187620267294@apache.org>

[java] <?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header>
<wsa:To>http://localhost:7878/notif/Consumer</wsa:To>

<wsa:From><wsa:Address>http://192.168.1.104:8084/wsn/producer</wsa:Address></wsa:From>
<wsa:MessageID>urn:uuid:8DB38D8DB2DEF5D9DF1187620267291</wsa:MessageID>

<wsa:Action>http://docs.oasis-open.org/wsn/bw-2/NotificationConsumer/NotifyRequest</wsa:Action>
</soapenv:Header>
<soapenv:Body>
<wsnt:Notify xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2">
<wsnt:NotificationMessage>
<wsnt:SubscriptionReference>
<wsa:Address>http://localhost:7878/notif/Consumer</wsa:Address>
</wsnt:SubscriptionReference>
<wsnt:Topic
Dialect="http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete">TopicSample</wsnt:Topic>

<wsnt:ProducerReference><wsa:Address>http://192.168.1.104:8084/wsn/producer</wsa:Address></
wsnt:ProducerReference>
<wsnt:Message>
<text>This is a notification message from the sample client service</text>
</wsnt:Message>
</wsnt:NotificationMessage>
</wsnt:Notify>
</soapenv:Body>
</soapenv:Envelope>
[java] --MIMEBoundaryurn_uuid_8DB38D8DB2DEF5D9DF1187620267293--
[java] #####
```

The element under the `<wsnt:Message>` element is the content of the original JBI message sent from the sample client.

2.4. Handling SOAP attachments

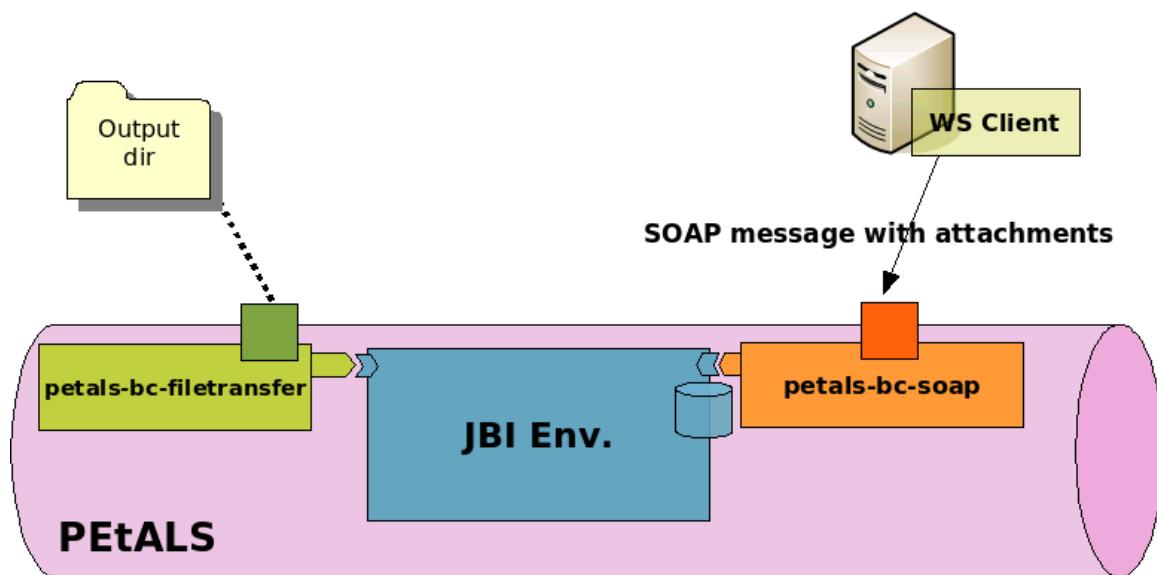
Warning

This usecase is under refactoring, it will be updated soon!

This usecase explains how the petals-bc-soap can handle SOAP attachments for incoming and outgoing SOAP messages. To get more details, please read the petals-bc-soap user's guide.

The JBI representation of the current usecase is :

Figure 2.8. Handling SOAP attachments with PEtALS



2.4.1. Setting up the platform

1. Start PEtALS (if not already started)
2. Install components (petals-bc-soap, petals-bc-filetransfer), deploy service assembly and start these artifacts. This can be done with the Ant script:

ant

The deployed service assembly contains:

1. A service unit which will be deployed on the petals-bc-filetransfer which provides a file transfer service. This service unit activates a *FileTransferMTOMEndpoint* endpoint.
2. A service unit which will be deployed on the petals-bc-soap which consumes *FileTransferMTOMEndpoint* endpoint activated by the previous service unit. This service unit also creates a new Axis2 service named *MTOMService*.

A Web Service sample client is also provided to send SOAP message with attachments to the exposed Web Service on <http://<HOST>:<PORT>/axis2/services/MTOMService>. This client uses the Axis2 client library to send the SOAP message with an attachment.

2.4.2. Usage

On the user side, after installation and deployment (see previous chapter), you just have to :

1. Launch the Web Service client which will send a SOAP message with attachments to the Web Service:

ant client

This invocation will send the mtom.txt file from the data directory as SOAP attachment (If you want to send another file, you can modify the Ant script).

The message transits in the different layers and components like this :

1. The SOAP message is handled by the petals-bc-soap activated Web Service
2. The SOAP message and its attachments are translated into a JBI message
3. The JBI message is sent to the file transfer activated endpoint

4. The JBI message and its attachments are written to files

To check that the usecase is valid you should have files in the directory you have chosen in the filetransfer service unit.

2.5. REST Services

This usecase explains how the petals-bc-soap can provide access to JBI services in REST mode and how REST services can be called. To get more specific details, please read the petals-bc-soap user's guide.

2.5.1. Setting up the platform

1. Start PEtALS (if not already started)
2. Install components (petals-bc-soap, petals-sample-helloworld, petals-sample-client), deploy service assemblies and start these artifacts. This can be done with the Ant script:

ant

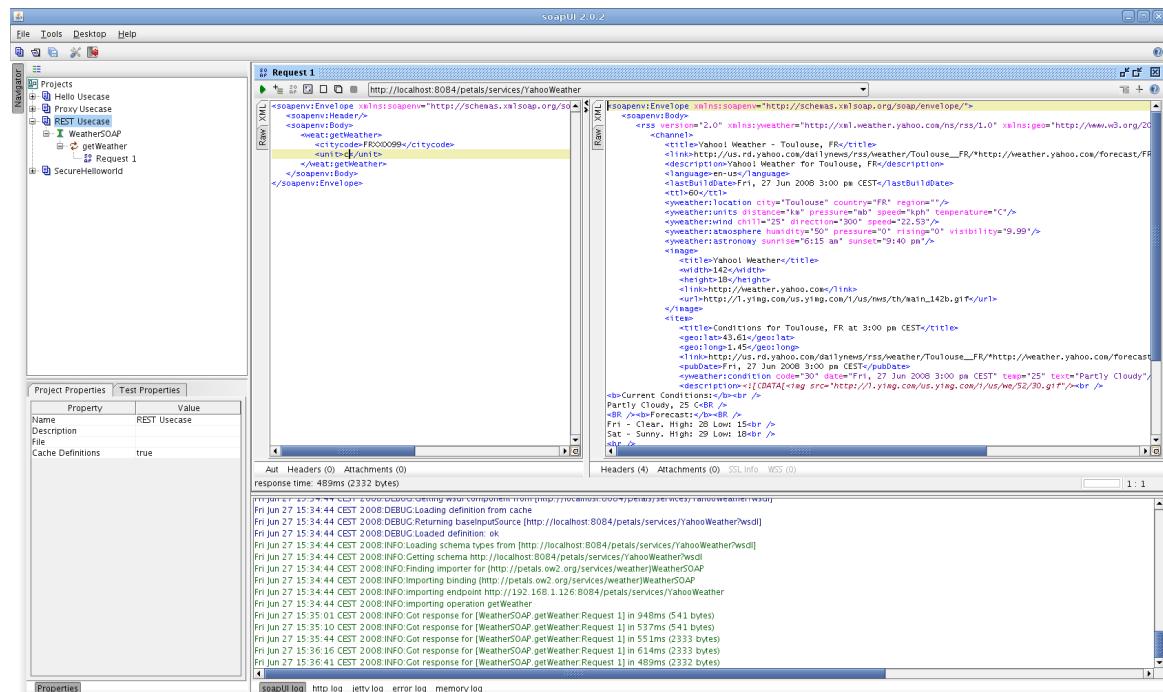
The deployed service assemblies contains:

1. A service unit which will be deployed on the petals-bc-soap which provides access to the Yahoo Weather REST service.
2. A service unit which will be deployed on the petals-bc-soap which consumes the previous activated service. It exposes the REST service as a SOAP one.
3. A service unit which will be deployed on the petals-sample-helloworld which provide the Helloworld service.
4. A service unit which will be deployed on the petals-bc-soap which will consume the previous Helloworld service and expose it as a REST service.

2.5.2. Usage

Import the SOAPUI project from the soapui folder of the usecase in order to call the Web Service which axposes the Yahoo Weather Service.

Figure 2.9. Calling the REST Yahoo Service with SOAPUI



You should have a response like:

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <rss version="2.0" xmlns:yweather="http://xml.weather.yahoo.com/ns/rss/1.0"
      xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
      <channel>
        <title>Yahoo! Weather - Toulouse, FR</title>

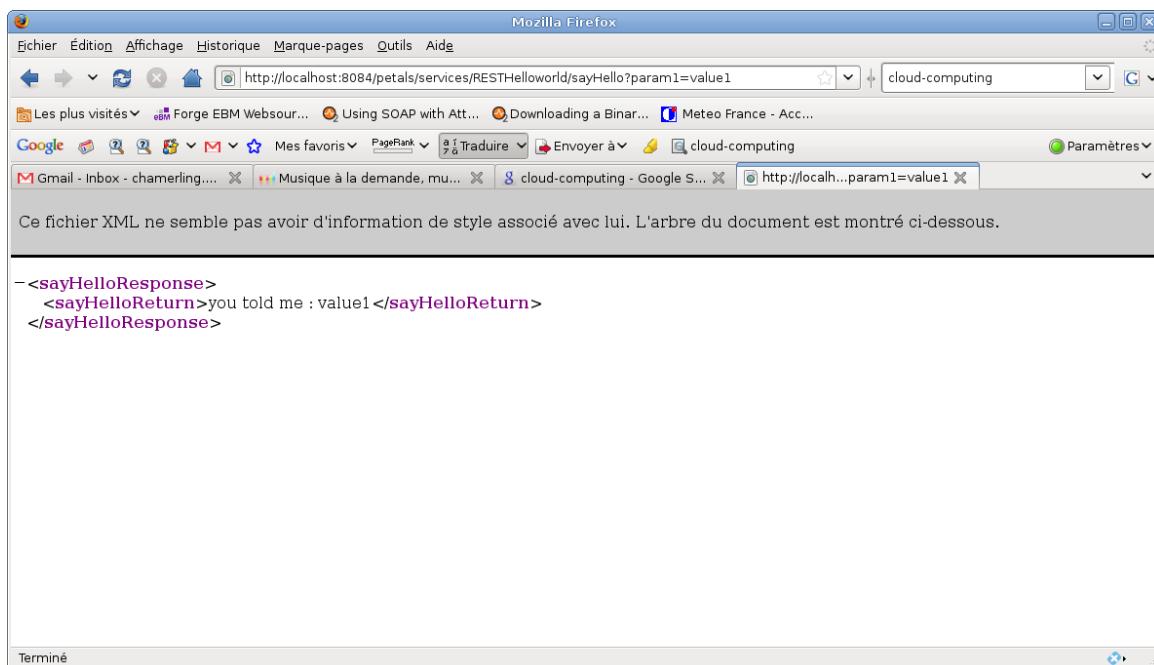
        <link>http://us.rd.yahoo.com/dailynews/rss/weather/Toulouse__FR/*http://weather.yahoo.com/forecast/
FRXX0099_c.html</link>
        <description>Yahoo! Weather for Toulouse, FR</description>
        <language>en-us</language>
        <lastBuildDate>Fri, 01 Aug 2008 4:30 pm CEST</lastBuildDate>
        <ttl>60</ttl>
        <yweather:location city="Toulouse" country="FR" region="" />
        <yweather:units distance="km" pressure="mb" speed="kph" temperature="C" />
        <yweather:wind chill="23" direction="350" speed="16.09" />
        <yweather:atmosphere humidity="53" pressure="0" rising="0" visibility="9.99" />
        <yweather:astronomy sunrise="6:44 am" sunset="9:17 pm" />
        <image>
          <title>Yahoo! Weather</title>
          <width>142</width>
          <height>18</height>
          <link>http://weather.yahoo.com</link>
          <url>http://l.yimg.com/us.yimg.com/i/us/nws/th/main_142b.gif</url>
        </image>
        <item>
          <title>Conditions for Toulouse, FR at 4:30 pm CEST</title>
          <geo:lat>43.61</geo:lat>
          <geo:long>1.45</geo:long>

          <link>http://us.rd.yahoo.com/dailynews/rss/weather/Toulouse__FR/*http://weather.yahoo.com/forecast/
FRXX0099_c.html</link>
          <pubDate>Fri, 01 Aug 2008 4:30 pm CEST</pubDate>
          <yweather:condition code="28" date="Fri, 01 Aug 2008 4:30 pm CEST" temp="23"
            text="Mostly Cloudy" />
          <description><![CDATA[<br
/>
<b>Current Conditions:</b><br />
Mostly Cloudy, 23 C<BR />
<BR /><b>Forecast:</b><BR />
Fri - Partly Cloudy. High: 23 Low: 17<br />
Sat - Partly Cloudy. High: 29 Low: 18<br />
<br />
<a
  href="http://us.rd.yahoo.com/dailynews/rss/weather/Toulouse__FR/*http://weather.yahoo.com/forecast/
FRXX0099_c.html">Full Forecast at Yahoo! Weather</a><BR />
(provided by The Weather Channel)<br />]]></description>
          <yweather:forecast code="29" date="1 Aug 2008" day="Fri" high="23" low="17"
            text="Partly Cloudy" />
          <yweather:forecast code="30" date="2 Aug 2008" day="Sat" high="29" low="18"
            text="Partly Cloudy" />
          <guid isPermaLink="false">FRXX0099_2008_08_01_16_30_CEST</guid>
        </item>
      </channel>
    </rss>
  </soapenv:Body>
</soapenv:Envelope>

```

The Helloworld service can be called in REST mode from your browser at the following URL <http://localhost:8084/petals/services/RESTHelloworld/sayHello?param1=value1> (see component documentation for parameters description).

Figure 2.10. Calling the Helloworld JBI Service in REST mode



2.6. Securing access to an exposed Web Service

This usecase explains how you can secure Web Service exposed by PEtALS through the SOAP binding component. This feature is available since the version 2.0 of the SOAP binding component. To get more information on this feature, read the component documentation.

The current usecase provide user/password security on an exposed Web Service.

2.6.1. Setting up the platform

To set this configuration, you must:

1. Start PEtALS (if not already started).

Install components (petals-bc-soap, petals-sample-helloworld), deploy service assembly and start these artifacts. This can be done with the Ant script:

ant

The deployed service assembly contains :

1. A service unit which will be deployed on the petals-se-helloworld which provides the helloworld service. This service unit activates a JBI endpoint.
2. A service unit which will be deployed on the petals-bc-soap which consumes the helloworld service. The deployment of this service unit will create a new Axis service in the component and will add a Web Service security context.

A Web Service client is also provided to send SOAP messages with and without security contexts to the exposed Web Service. This client uses the Axis2 client library to build and send the SOAP messages.

2.6.2. Usage

We use SOAPUI to invoke Web Services operations. Import the SOAPUI project located in the soapui folder of the usecase.

The sayHello operation contains two requests :

1. The first one (Request secure) is secured, it contains the required SOAP security headers that will be handled by the Rampart module on the SOAP binding component side. The request is then processed by the component and sent to the right service.
2. The second one (Request unsecure) is not secured. The Rampart module will return a SOAP fault.

Figure 2.11. SOAP security unsecure call with SOAPUI

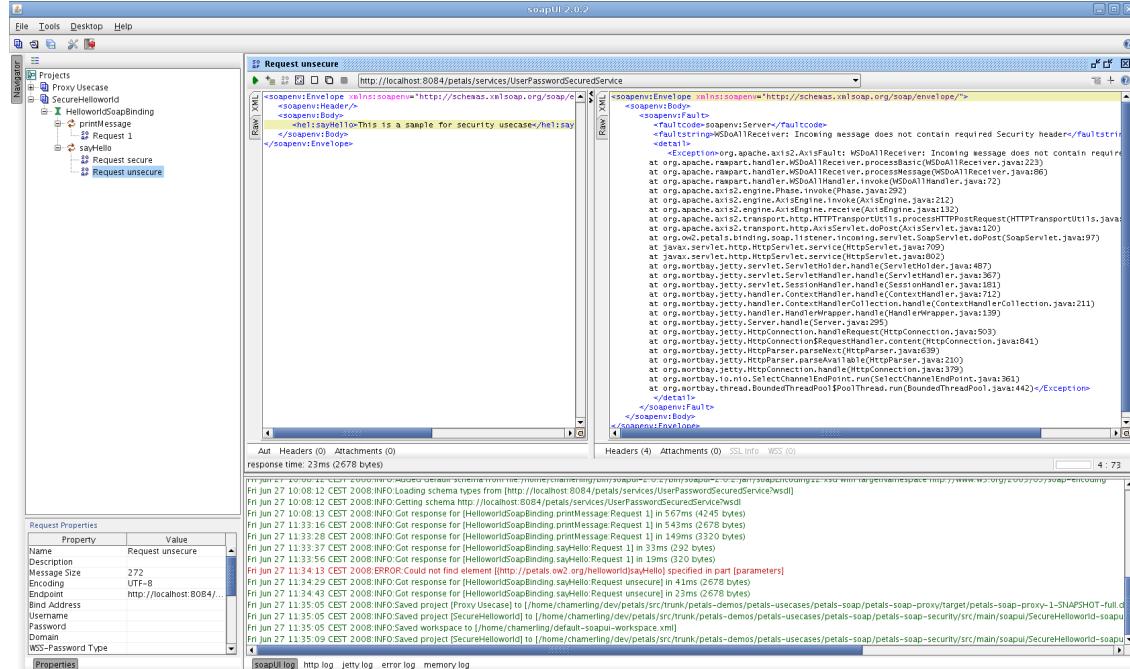


Figure 2.12. SOAP security secure call with SOAPUI

