Java CAPS 5.1.3

Creating a Simple Web Service from a JCD

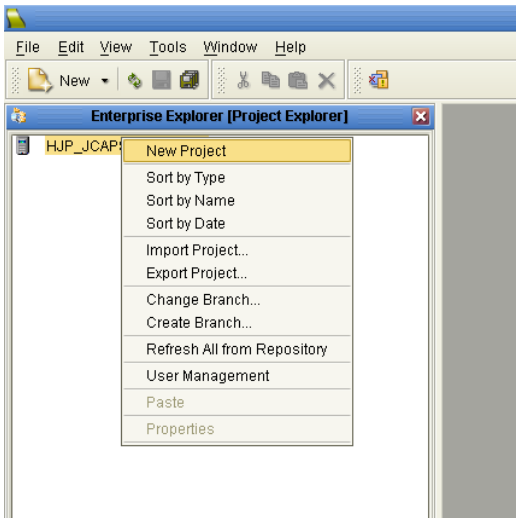
Holger Paffrath, August 2008

Introduction

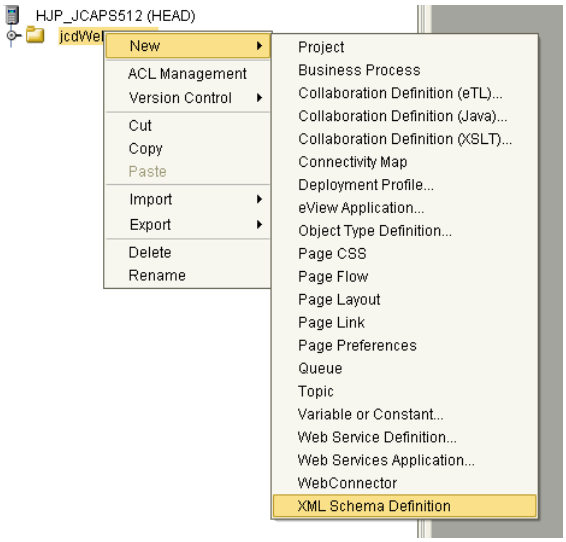
This tutorial shows you how to create an XML Schema definition to define the layout of your web service message, create an OTD based on this schema definition.
Build and deploy a Java Collaboration which exposes this OTD as a web service callable from external sources.
Finally goes through how to test your web service using a third party application (SOAPUI).

Tutorial

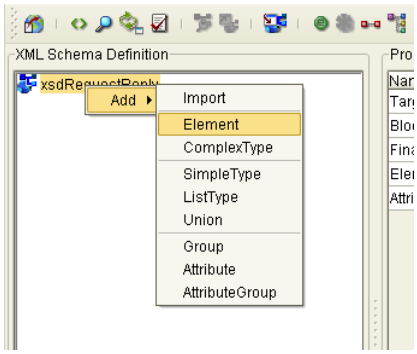
1. Create a new project called “jcdWebservice”.



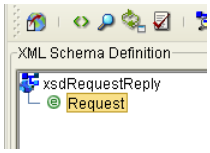
2. RightClick the project and select “New” then “XML Schema Definition”.
   This will create a new XSD that will hold the inbound and outbound message formats for the Web Service Request Reply.



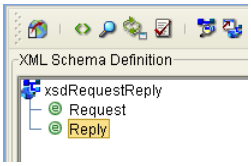
3. Rename the XSD to xsdRequestReply.
4. In the schema definition window, Right click the root node and select “Add” then “Element”. This element will be the root node of our XML message.



5. Give the element the name “Request”. It can be any name, its just that this is the one that I've chosen.

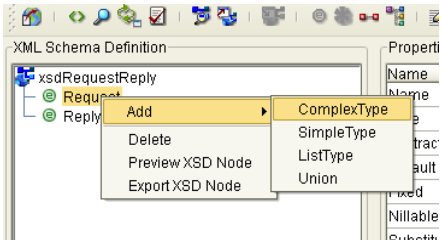


6. Create a second element called “Reply”

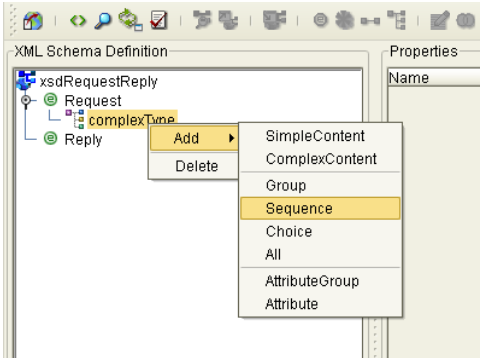


Now we have to create the underlying details for the XML message.

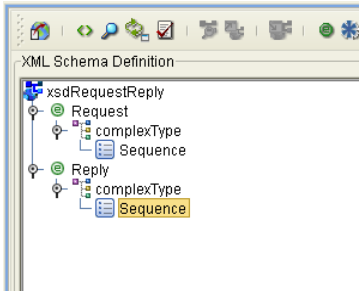
7. Under Request, Right click and select “Add” then “ComplexType”.



8. Under the newly created ComplexType, Right click and select “Add” then “Sequence”

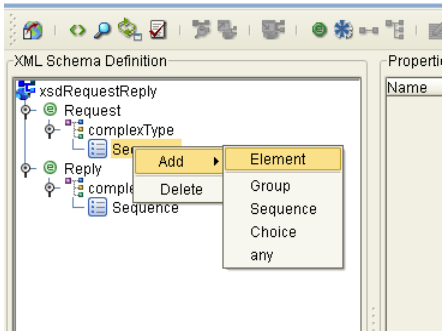


9. Repeat the process for Reply.



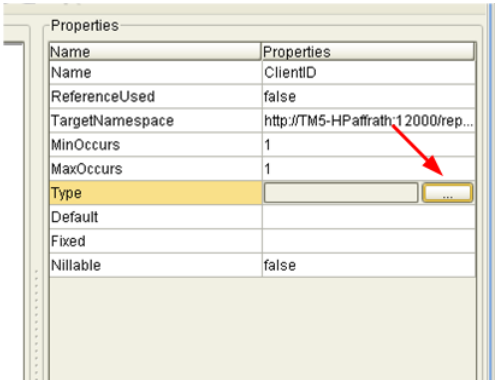
Now we are going to create the data fields.
For the Request, I'm going to add a field called “ClientID” and for the Reply, I'm going to add “FirstName” and “LastName”.

10. Right Click the Sequence under Request and select “Add” then “Element”.

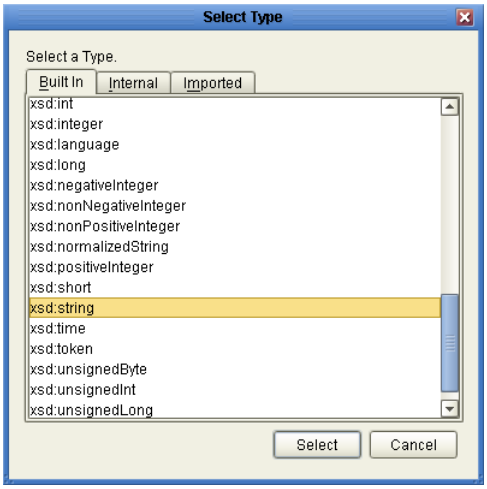


Give it the name “ClientID”.

11. To the Right of the Tree view, you should see a Properties view. Since our element is going to hold data, we need to define the type of data that it is going to contain. In this case it will be a String.
Select the “Type” field and click on the button with the 3 dots [...].



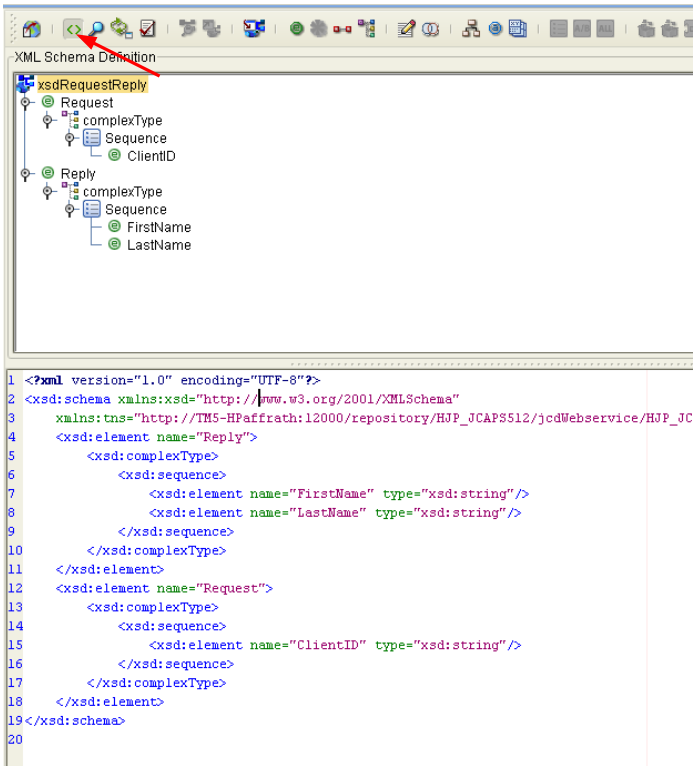
12. In the Dialog box that appears, scroll down till you see “xsd:string” then select this type by pressing the “Select” button.



13. Repeat the above steps for the Reply. Adding “FirstName” as type “xsd:string” and “LastName” as type “xsd:string”.

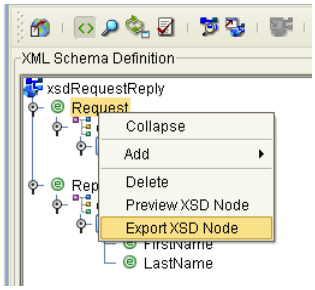
> Tip : If you need to rename an element, change the name in the properties window. You cannot right click and rename.

To view the source code for the XML Schema just created, select the “Source Code” button.

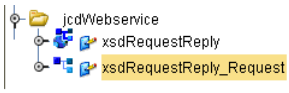


We now have our XML Schema defined.
The next step is to create an OTDs that will use this Xml schema.
This is relatively simple compared to building the XML Schema.

14. Right Click our Root node, in this case “Request” and select “Export XSD Node”.



A new OTD will be generated in our project view.

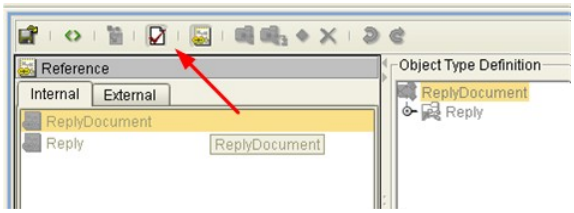


15. Repeat the process for Reply.

We now have our OTDs.

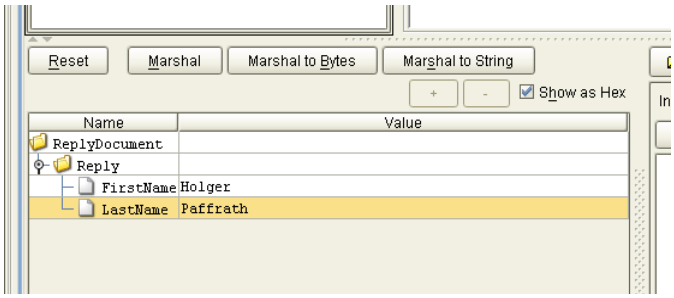
To Test the OTD's

16. Open the OTD and then Press the “Test” button.



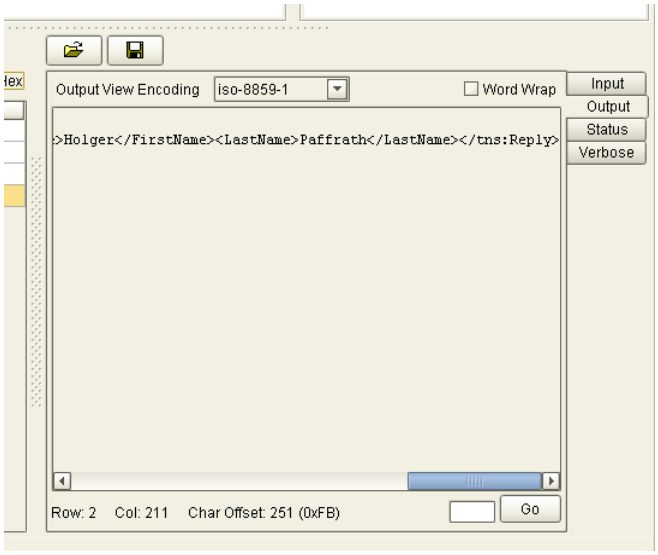
The test area will appear on the botton half of the screen.

17. Expand out the XML tree and start entering some data.



Then press the “Marshal to String” A result will appear in the output window.

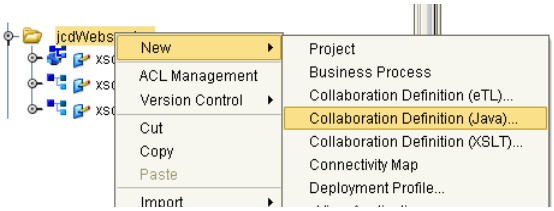
18. The Result can be seen by selecting the “Output” tab.



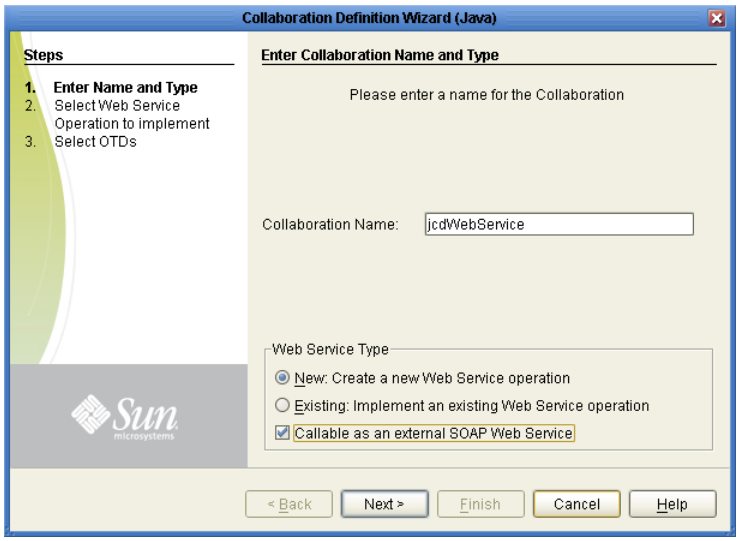
You will be able to cut and paste the XML message to verify its output.

Now we've come to the point where we need to create the Java Collaboration.

19. Right Click the project and select “New” then “Collaboration Definition (Java)”.



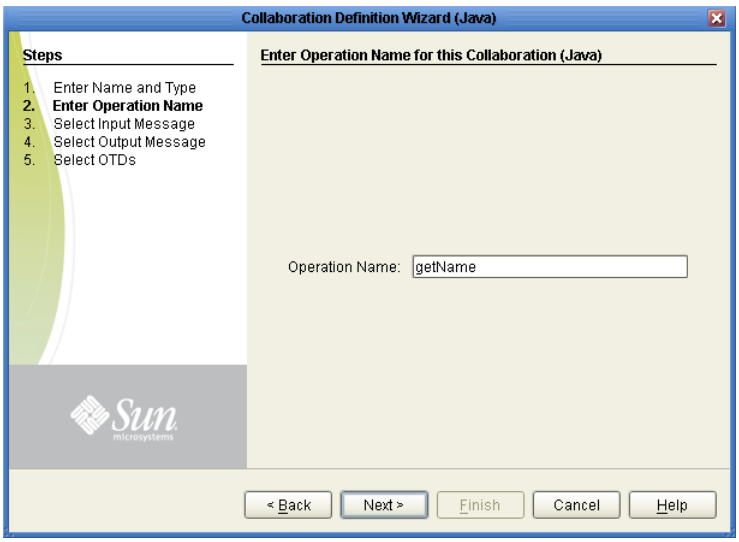
20. Give it the name “jcdWebService” and select the Web Service Type as “New : Create a New Web Service Operation”.



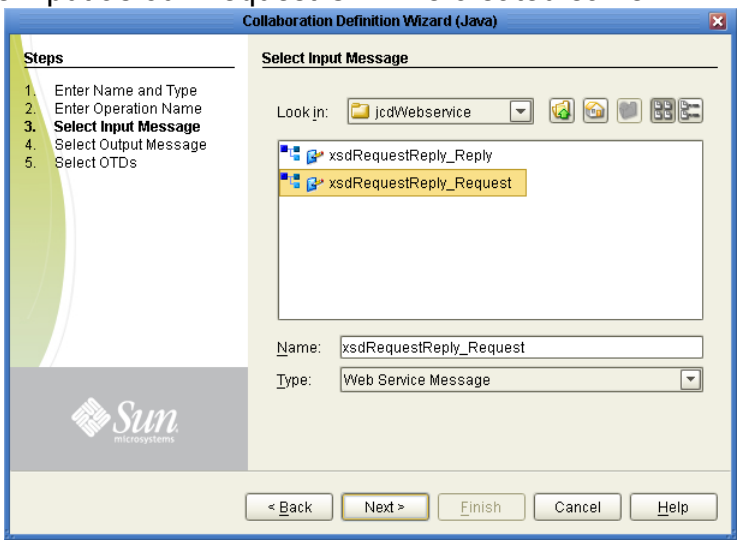
Finally, make sure that the check box for “Callable as an external SOAP Web Service” is checked. This will make the web service available to the outside world.

Then click “Next”

21. Give the web service the Operation Name of “getName” and then click “Next”.

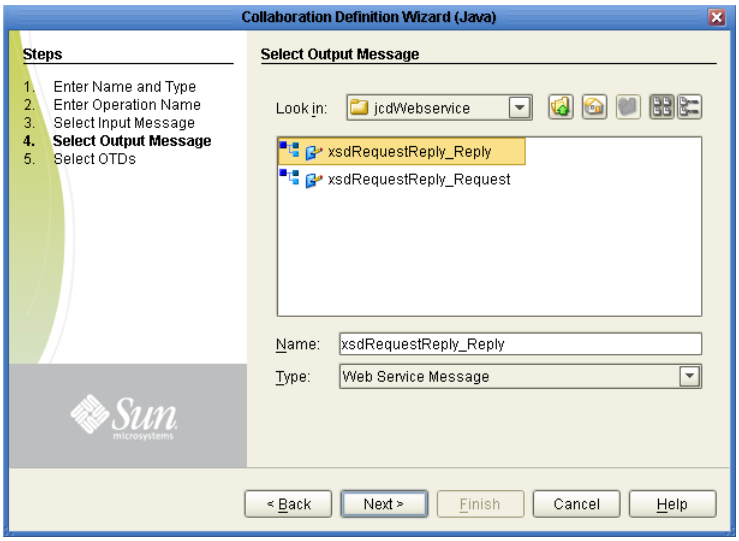


22. Select the input as our Request OTD we created earlier.



And click “Next”

23. Select the Output as our Reply OTD we created earlier.



And click Next.

The next view allows you to add more OTDs. For example, a Database OTD if you

want to interrogate a database for the required data, or a JMS OTD if you want to put the request or any other message to a queue.

We are just going to keep this simple, so no other OTDs will be required. Just click “Finish”.

Your JCD will be created.

The whole point of this tutorial is to show you how to create a jcd as a web service, so the logic in this jcd will be kept to a minimum.

The logic will simply be, if the Client ID is “1” then it will return “Holger Paffrath” otherwise it will return “John Doe”.

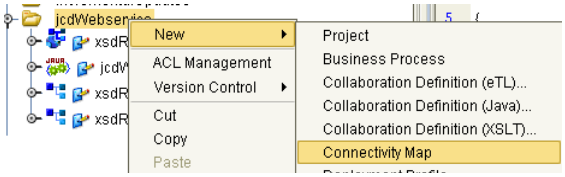
24. Add the source code to the jcd as shown below,

```
public void getName( stcgen.fcxotd.http___TM5_HPaffrath_12000_repository_HJP_JCAPS5121224690628.
    throws Throwable
{
    if (input.getRequest().getClientID().trim().equals( "1" )) {
        output.getReply().setFirstName( "Holger" );
        output.getReply().setLastName( "Paffrath" );
    } else {
        output.getReply().setFirstName( "John" );
        output.getReply().setLastName( "Doe" );
    }
}
```

and Save.

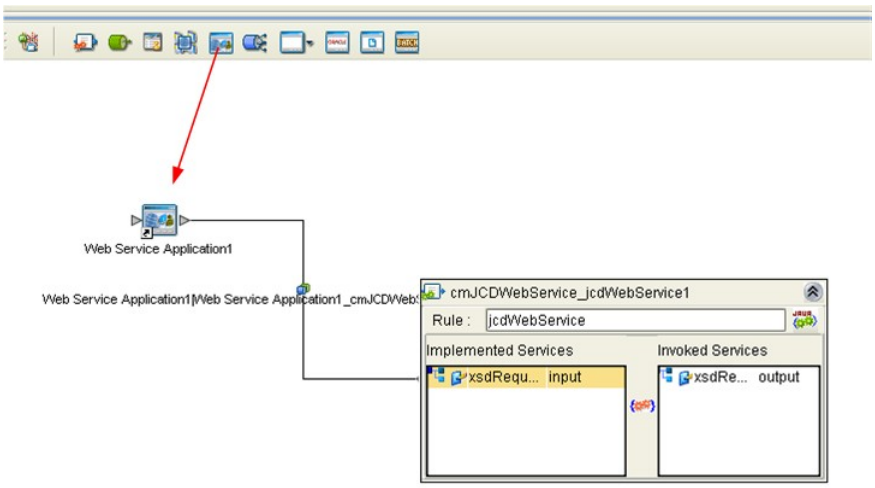
Now to create the connectivity map and deployment.

25. Right Click the project and create a new “Connectivity Map”.



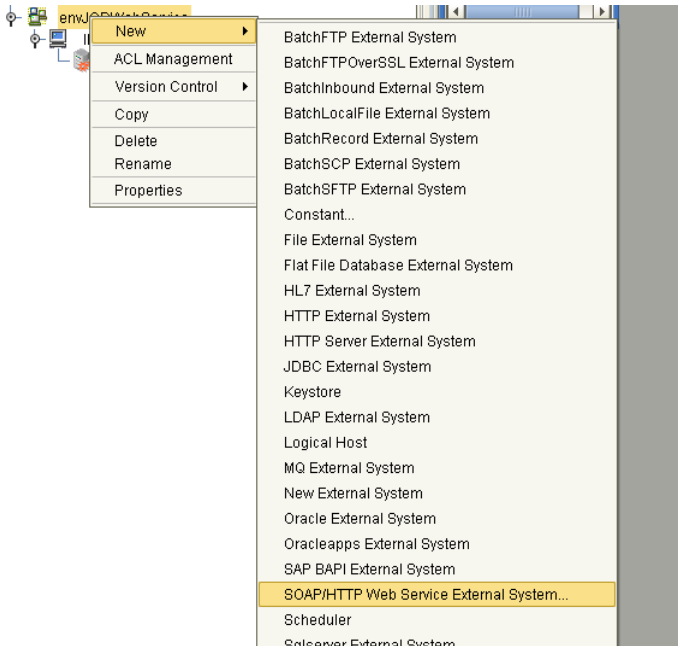
And give it the name “cmJCDWebService”

26. Drag the Web Service Application as shown onto the connectivity map and link it up

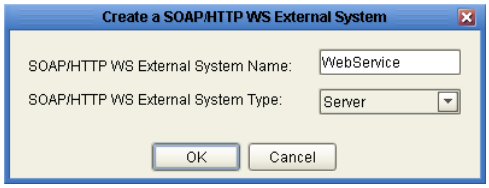


to the Java Collaboration which you should also drag across from the project. Accept all defaults for the various dialog boxes that pop up.

27. Now, before we deploy, you need to add a web service component to your environment.
    Go to the environment TAB. Right Click your environment and select “New” then “SOAP/HTTP Web Service External System”.



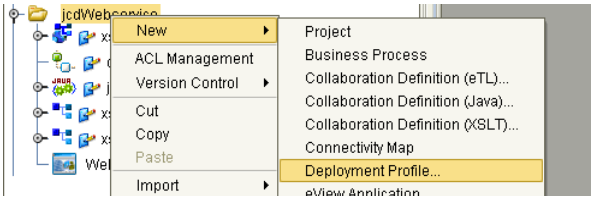
    Give it the name “Web Service” and leave it as a “Server” as we are hosting the service.



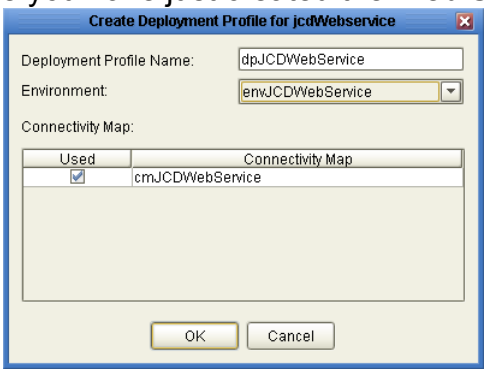
> Note : By default, the web service is configured to give a URL with a host name of “localhost”. This setting is fine for this tutorial, but can be changed after the first build of the project. The reason I have suggested after the first build is because all settings are filled in automatically based on the web service being deployed and its easier to change just the host-name rather than making entries for everything else. Steps to change the host-name are at the end of this tutorial.

    Click OK and return back to the Project View.
    We can now create the deployment profile.

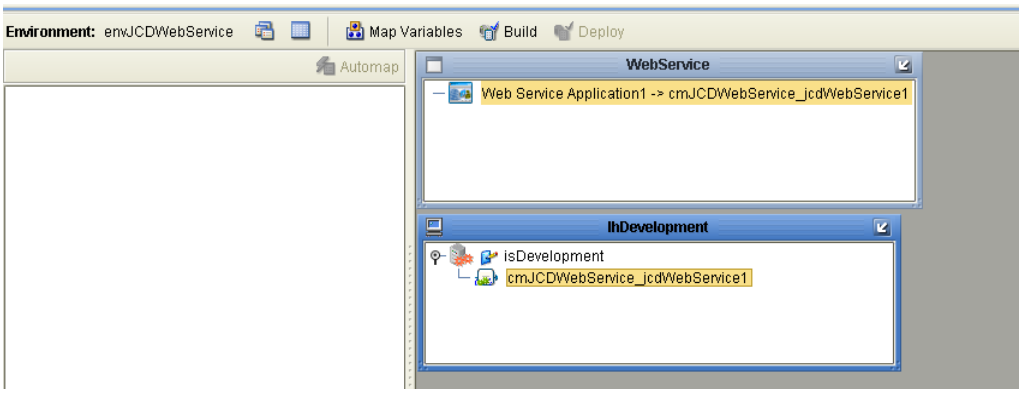
28. Right click the “jcdWebService” project and select “New” then “Deployment Profile”



29. Give the deployment profile the name “dpJCDWebService” and select your environment where you have just created the Web Service Application.



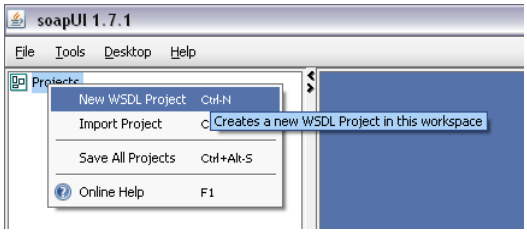
30. Map your components and build.



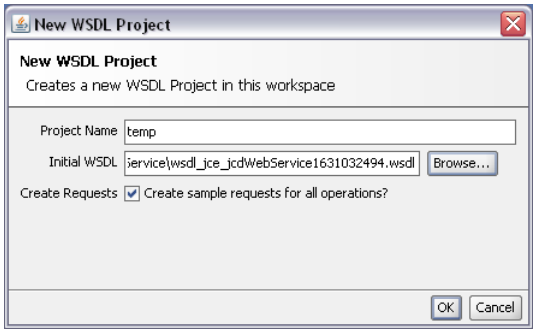
31. During the build, the compiler will ask if you want to deploy the WSDL to a UDDI server or to a file. If you do not have a UDDI server set up (As is the case in this demo), it will automatically ask where you would like to save the file.
    Save the file to a temporary directory. It contains a jar file that contains the details of the WSDL that would normally be stored in the UDDI registry.
32. Extract the jar file to a temporary directory. We are going to use it for testing.
    For any developers that wish to call the Web Service, give them a copy of the jar file created.

33. For testing, I like to use an open source program called “SoapUI” which is freely available from http://www.soapui.org.
    Download and install SoapUI.

34. Right Click the Project, and select “New WSDL Project”

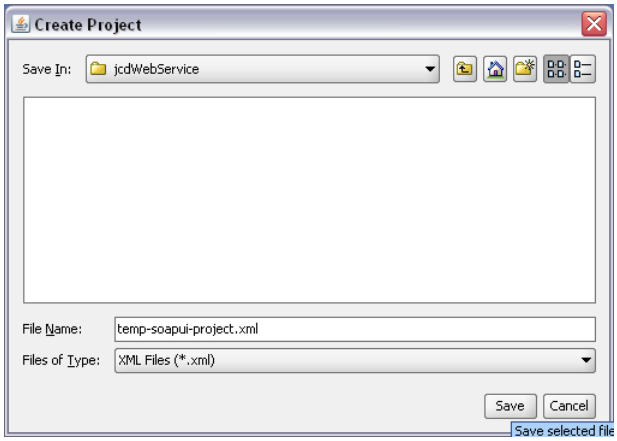


35. Give the project the name of “temp”. For the Initial WSDL, browse to the *.WSDL file found in the expanded .jar file created by the build created earlier. You might have to go down several levels of directories before you find the *.wsdl file.

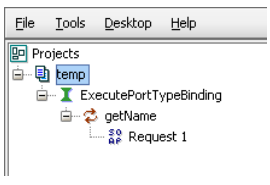


Finally, check “Create sample requests for all operations” This will give you a “?” in all valid fields.
Click OK.

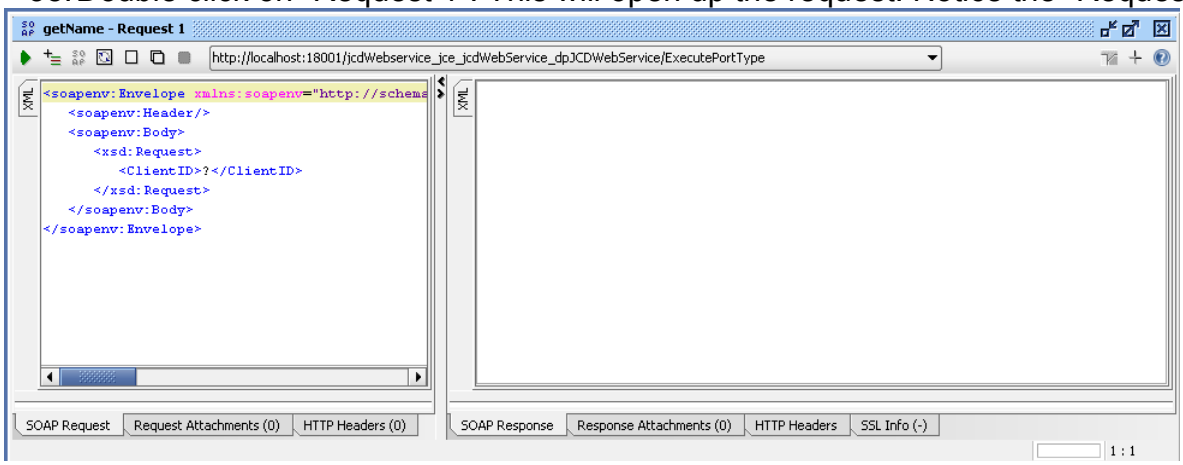
36. Save your project to a working directory.



37. A tree view of the Web service will now be visible. Expand this out.

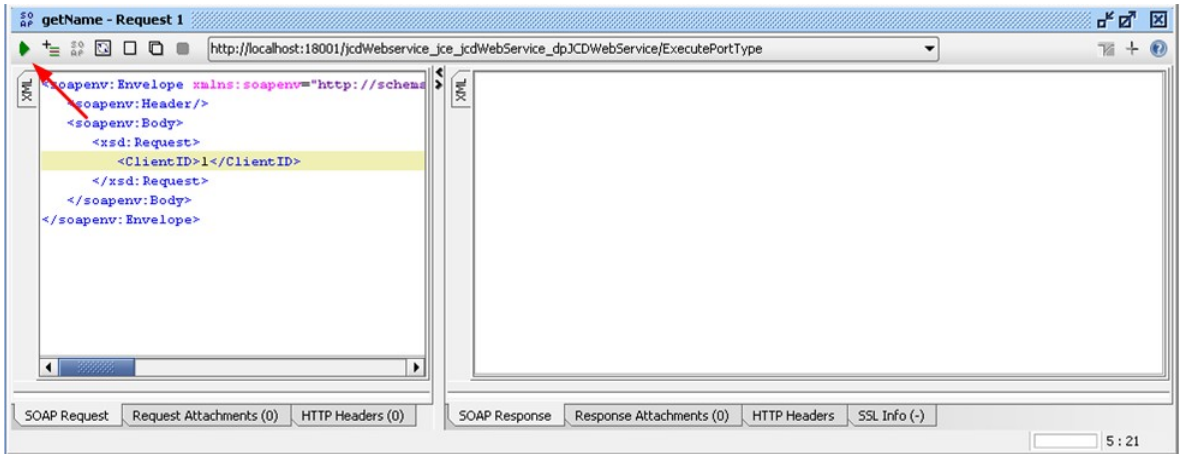


38. Double click on “Request 1”. This will open up the request. Notice the “Request”

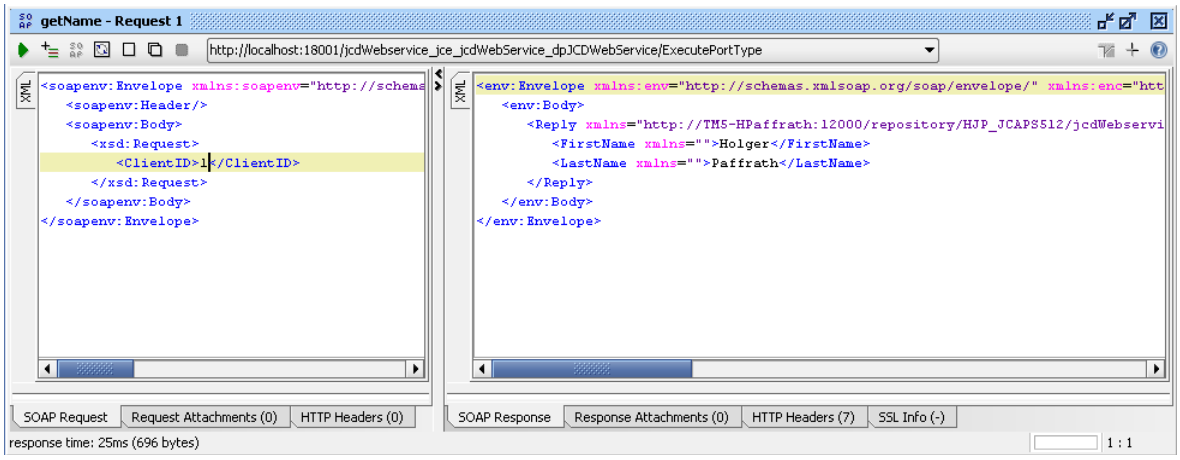


and “ClientID” field. This is the field we created in the beginning of this tutorial. It has been given the default value of “?” This view shows our XSD created earlier wrapped up in a SOAP envelope.
Also notice that on the right pane, there is a tab called “SOAP Response”. This is the window where our reply will be shown.
Now to run a test.

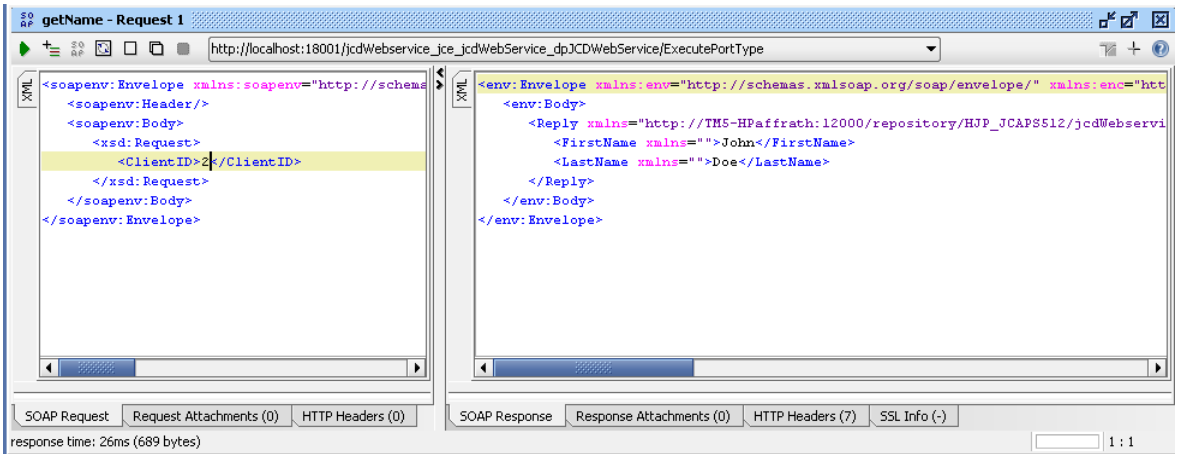
39. Change the Question Mark in the SOAP Request to “1” and press the “Run” button as shown.



40. The Result will be ...



If you enter a number other than “1”, the result will be ...



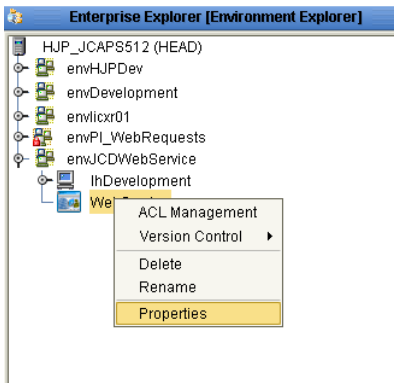
Changing the Host Name for the WSDL

As mentioned in Step 27, by default the host name of the web service is set to “localhost”.

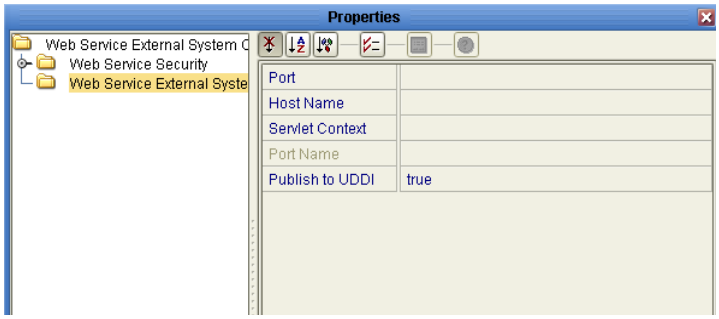
It is highly unlikely that a web service developed will only be accessed by its localhost. Most of the time, it will be accessed by clients residing on other hosts.

The change to the host name is relatively minor.

1. Go to the Environment Configuration tab and right click on the Web Service Container created and select “Properties”

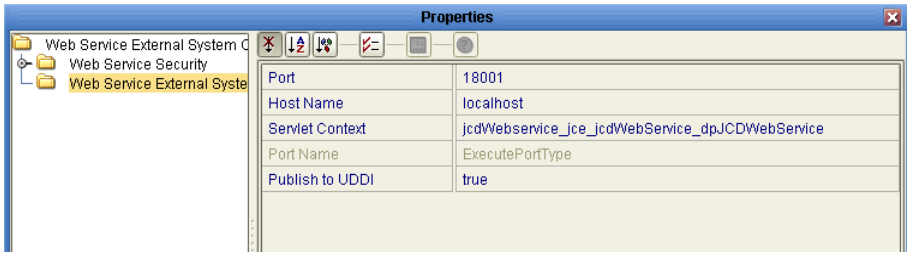


2. The properties dialog box will appear. Select “Web Service External System Configuration” in the tree view. You should now see the following.



This is the default settings when the container is created. Leave these settings and continue the tutorial above to build your project.
Once your project is built, return here.

3. After the build, go back to the properties of the Web Service container and you will see that the properties have changed.

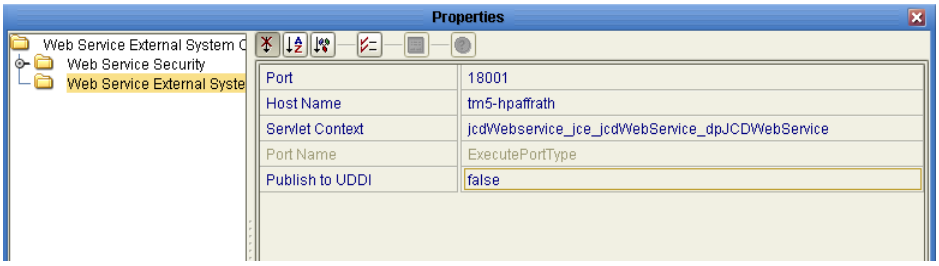


The Host Name is the hostname that the web service will reside – change this value to the correct host name or ip address.

The Servlet Context is a path to your web service. This path can be left as is, or you

can change it to something more meaningful to your environment.

Publish to UDDI tells Java CAPS to publish the Web Service to a UDDI server. In this example, one has not been configured. So this setting can be changed to “false”.
The following shows the changed settings.



4. Build the project. You will be prompted to save the WSDL.jar file. Save this to a temporary directory and finally deploy.

   Decompress the WSDL.jar file and drill down to the WSDL file.

   If you open it up with a text editor, you should see that the host name has changed in the URL for the SOAP Address.

   Now you can continue on from Step 33.

Summary

In this tutorial you have been shown how to create a web service from a JCD. This is one method of many in Java CAPS to create web services. This particular method is handy if you do not have eInsight installed or in use.
There is the limitation though that you are only allowed to have one operation per JCD. If you need more than one operation per Web Service, then it is better to use eInsight.