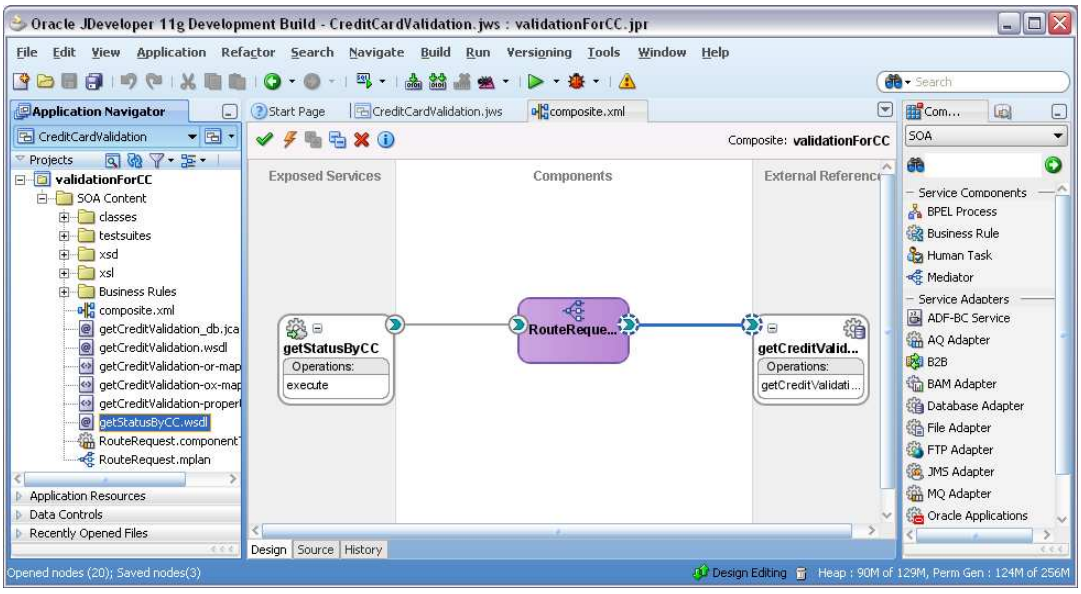
2 Creating the Credit Card Validation Service

2.1 Introduction

Note: The solution for this chapter can be found in `c:\po\solutions\ch2`. To run this solution, you must first complete Chapter 1 to set up the application.

In this section you will build a simple credit service. This service will look up the status (valid or not valid) of a given credit card from a database.

The implementation of this service uses a Mediator to route the request to the database adapter which executes the query on the database and returns a result. Once done, your Mediator flow will look like this.

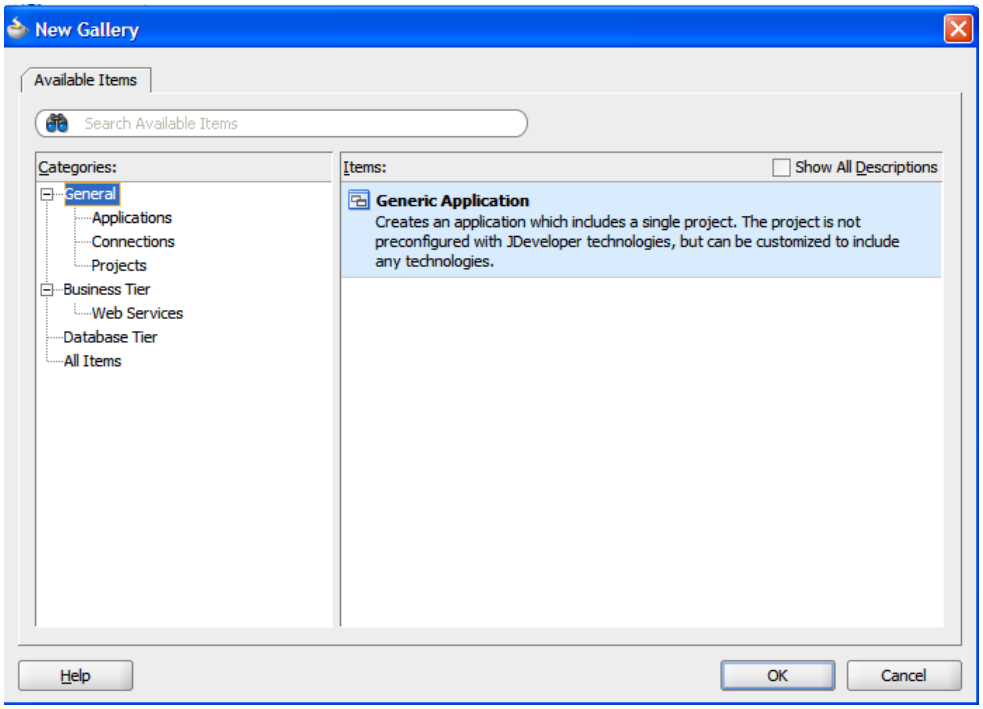


2.2 Designing the flow

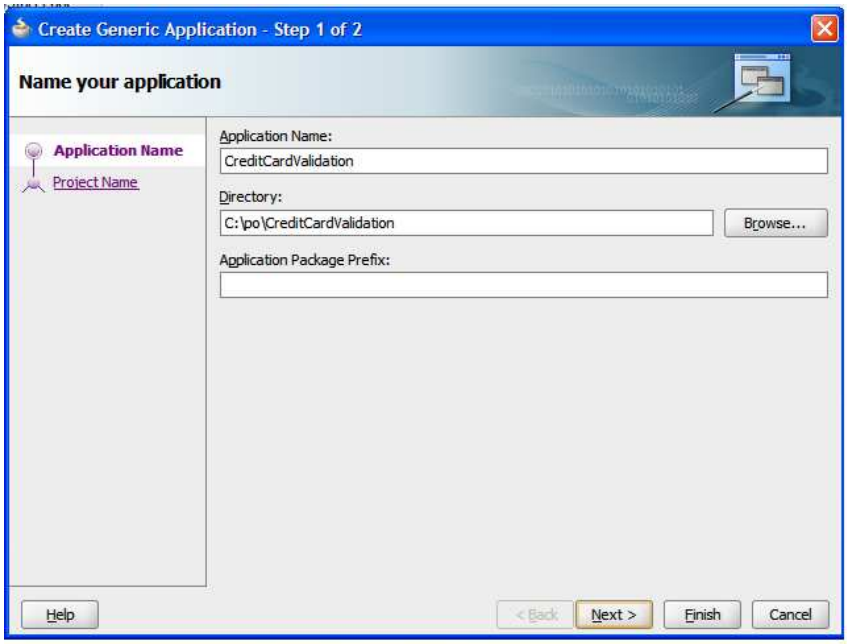
You will create a new composite application that will be invoked as a service from your main application to be designed later. This synchronous composite queries the database for the credit card number and returns its status.

2.3 Creating a new application

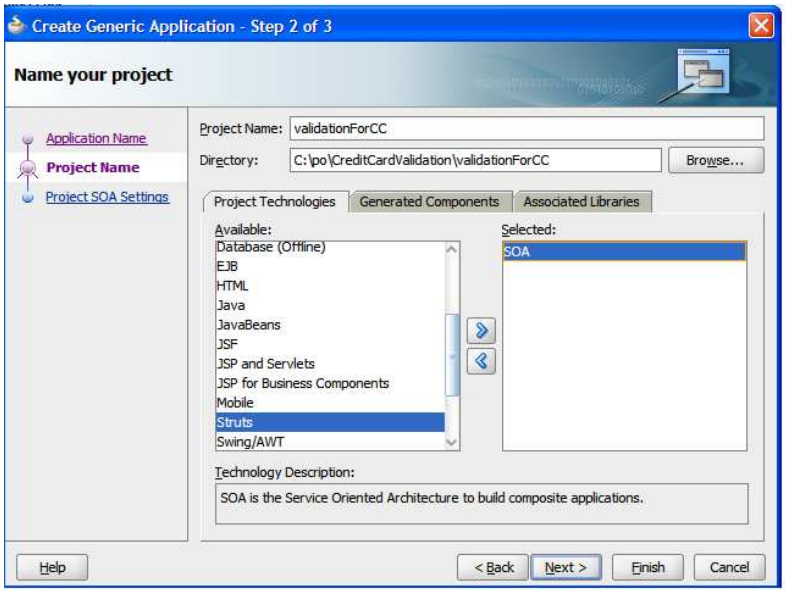
1. Start JDeveloper.
2. Create a new application. There are various ways and shortcuts to do this, and in this case choose **File > New…** from the menu.
3. Select **All Technologies** in the top tab if present (only if you have a project open)
4. From the **Categories** tree, click on **General** (don't expand it).
5. Select **Generic Application** from the **Items** field.



6. Click **OK**.
7. In the subsequent **Create Generic Application** dialog set the following fields, leaving the others with their default values:
   - **Application Name:** CreditCardValidation
   - **Directory:** C:\po\CreditCardValidation



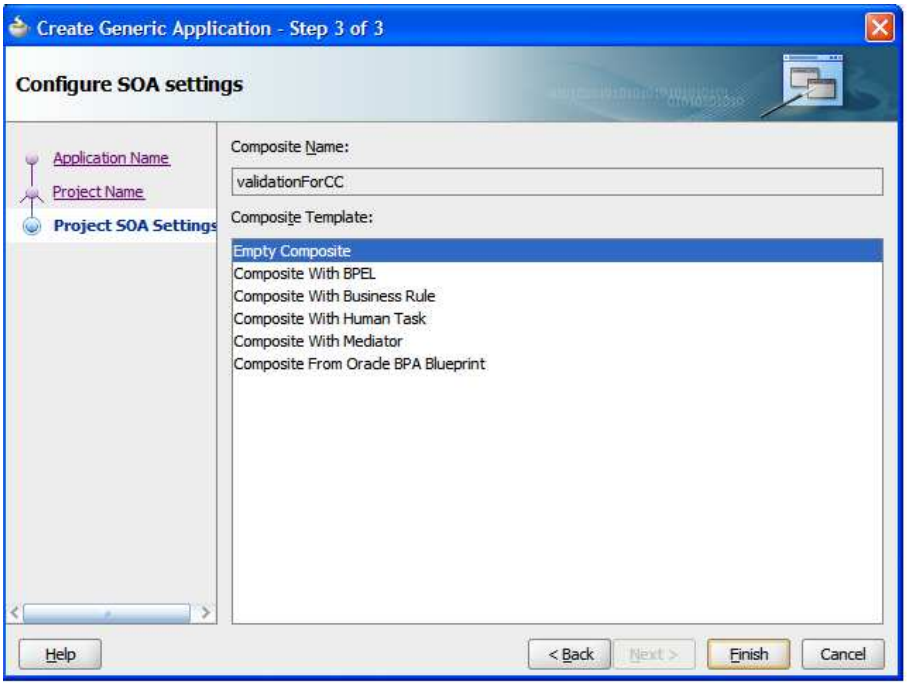
8. Click **Next**.
9. When you create a new application you are prompted to create a new project. Set the following fields:
   - **Project Name:** validationForCC
   - **Directory:** C:\po\CreditCardValidation\validationForCC
   - **Project Technologies:** SOA



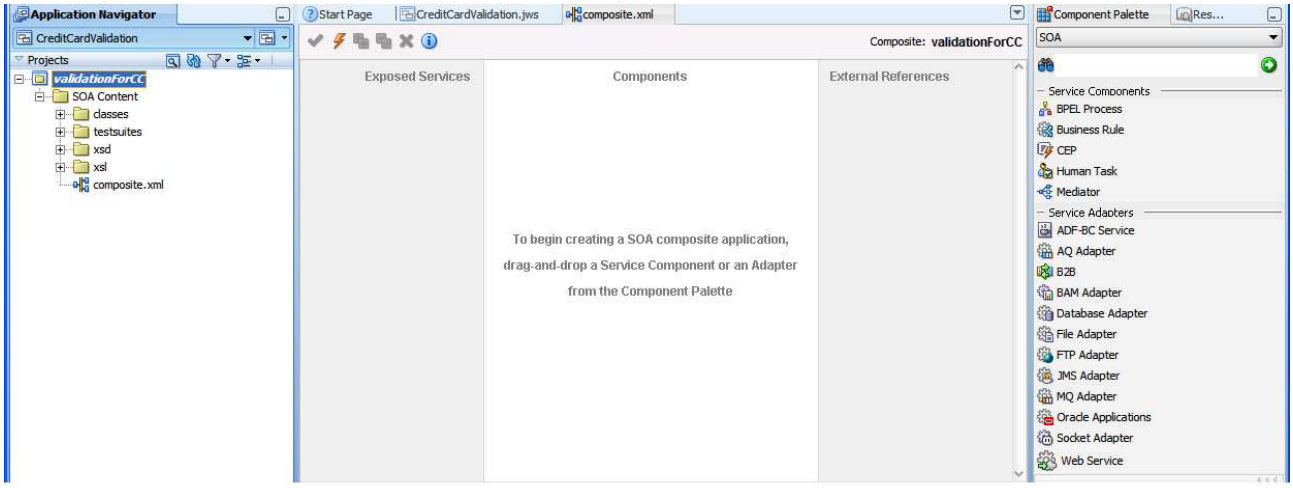
10. Click **Next**.

11. The next step allows you to choose to start with an empty composite, or with a component already added. In this case, select **Empty Composite.**

12. Click **Finish.**

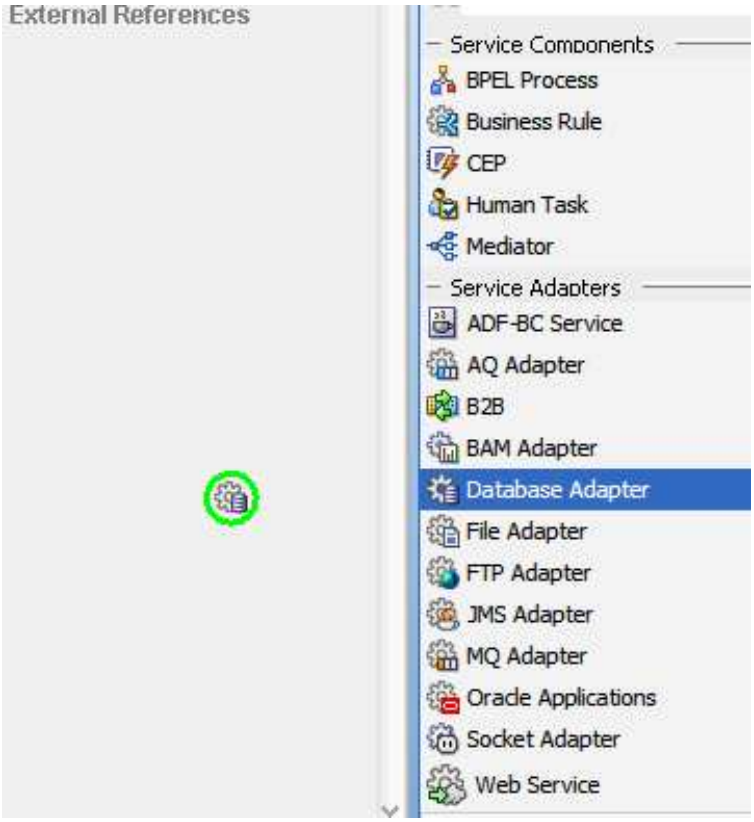


13. You will now have an empty canvas displaying three swim lanes: services, components, and references.



2.4 Adding the database adapter

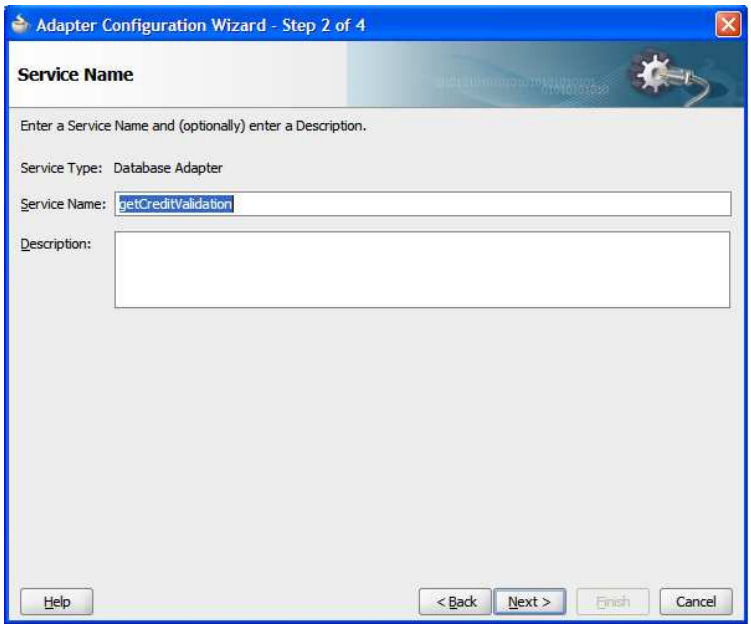
1. You can now start building the application. Drag-and-drop a database adapter onto the **External References** swim lane. If you don't have the Component Palette open, from the menu select **View > Component Palette**. If the Component Palette is not showing the SOA components, select **SOA** from the drop down list.



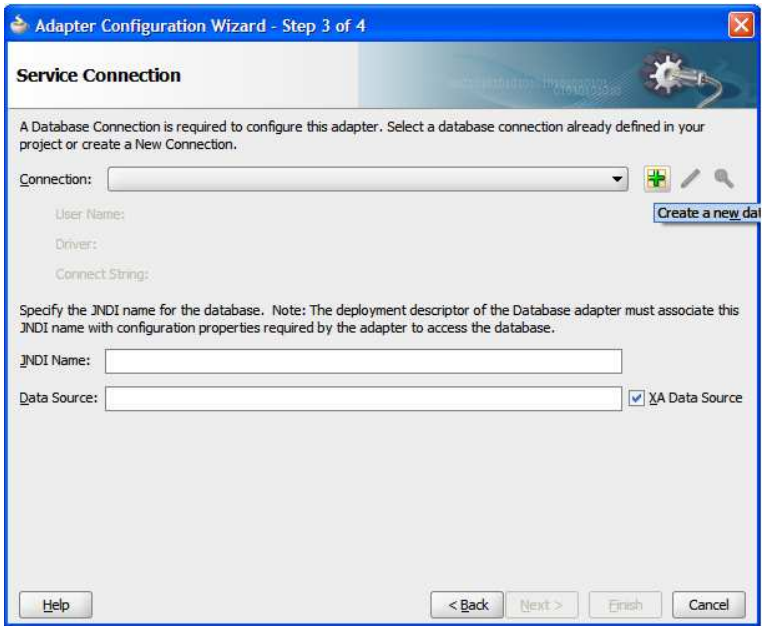
2. This database adapter call will return a result of valid or invalid for a given credit card from the database. A wizard takes you through the steps of configuring the database adapter.

   The title bar of the wizard dialog shows the step number. Click **Next** on step 1.

3. Enter the following details on step 2:
   - **Service Name:** getCreditValidation



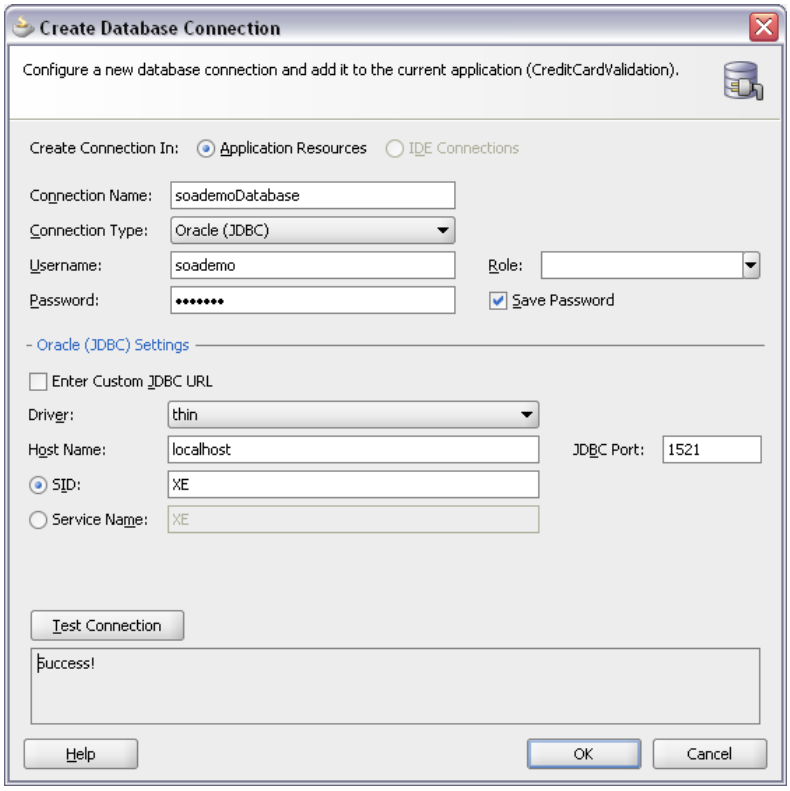
4. Click **Next** to go to step 3.



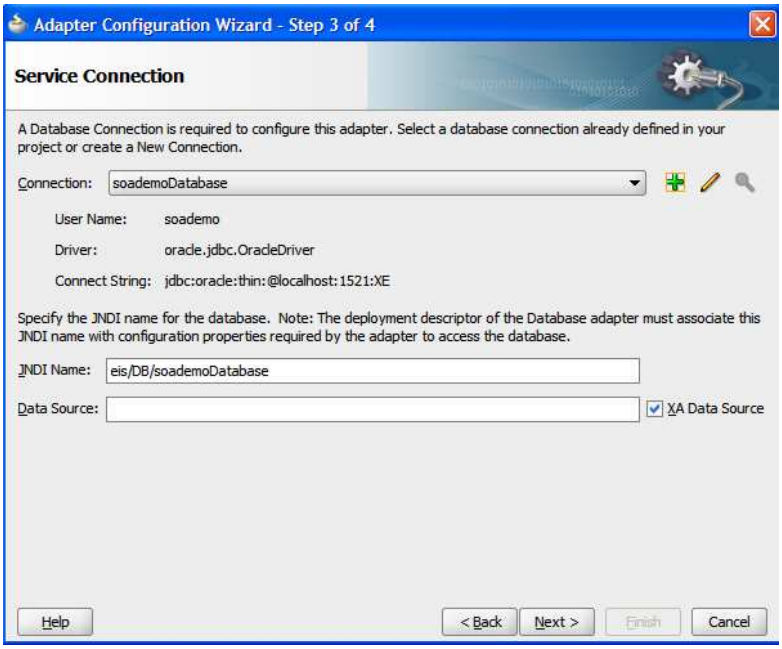
5. The database adapter will connect to the database, and in order to do that it needs a database connection which contains all the details needed to connect to the database. You can pre-create a database connection or create one on the fly. Since a database connection hasn't been created yet, you will create one from here.

Click the green plus icon to the right of the **Connection** poplist to create a new database connection.

6. In the **Create Database Connection** dialog, enter the following details:
   - **Connection Name:** soademoDatabase
   - **Connection Type:** Oracle (JDBC)
   - **Username:** soademo
   - **Password:** soademo
   - **Save Password:** Checked
   - **Enter Custom JDBC URL:** Unchecked
   - **Driver:** thin
   - **Host Name:** localhost
   - **JDBC Port:** 1521 (or the port number of your database)
   - **SID:** XE (or the SID of your database)



7. Click the **Test Connection** button and verify that your connection works.
8. Click **OK** to return to step 3 of the **Database Adapter Configuration** wizard.



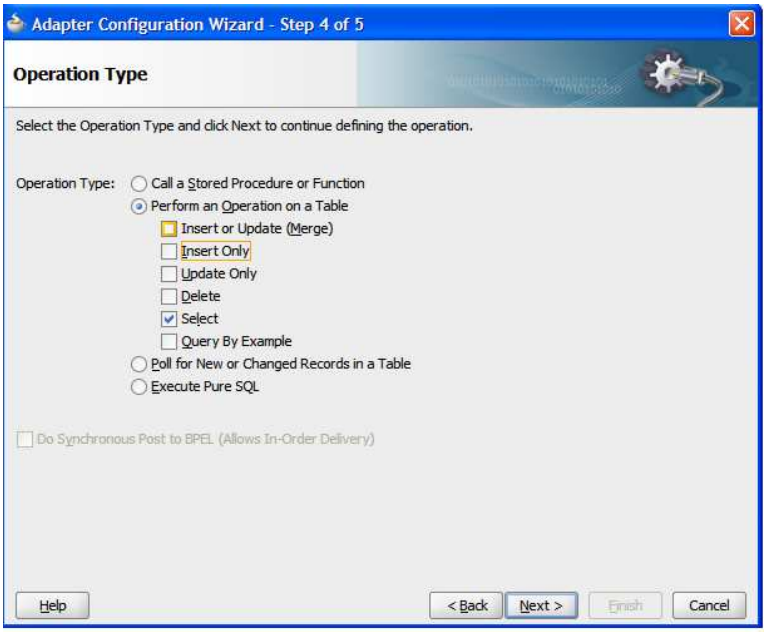
9. Make sure the JNDI Name matches the data source connection pool JNDI Name you entered in Chapter 1.

   Click **Next**.

10. Set the following fields:

    - **Perform an Operation on a Table:** Selected
    - **Select:** Checked

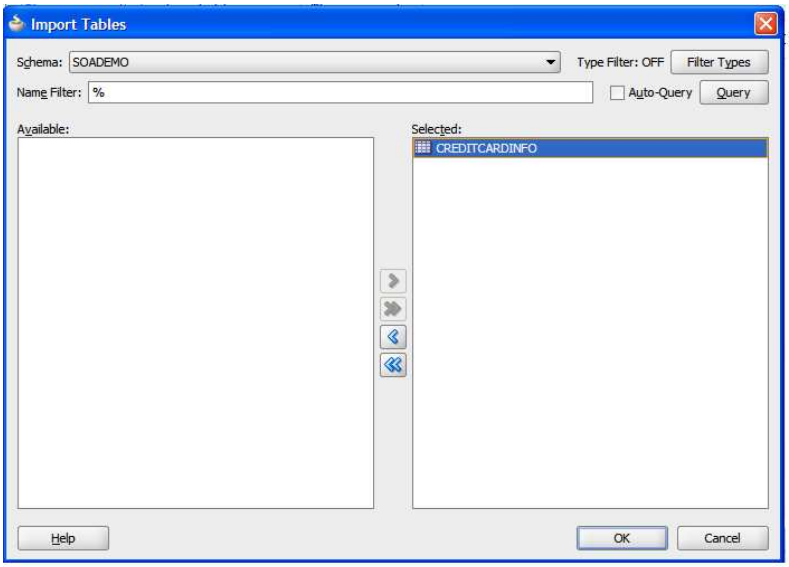
All other fields should be unchecked.



11. Click **Next**.

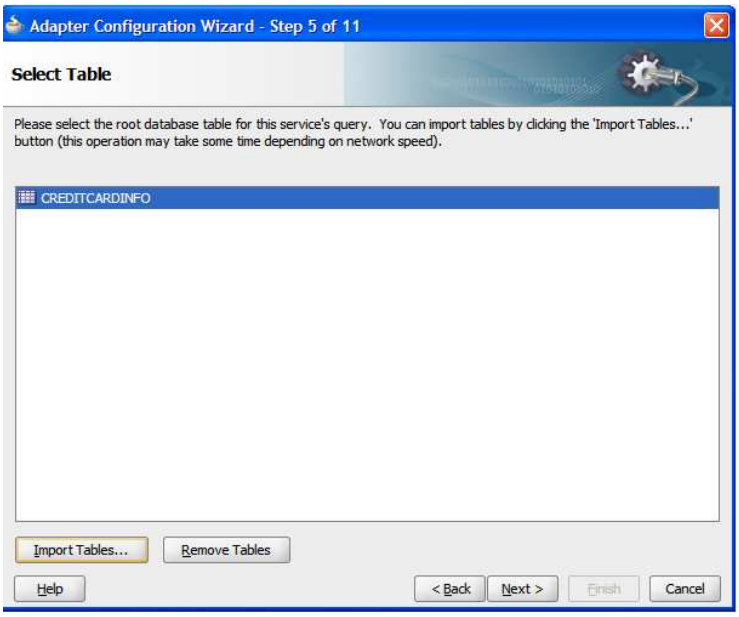
12. In step 5, click the **Import Tables** button and then click the **Query** button to retrieve the list of tables for the *soademo* user from the database. If you do not see any tables, go back to Chapter 1 to make sure the table was created properly.

13. Select the **CREDITCARDINFO** table and move it to the **Selected** field by pressing the **Add** shuttle button.



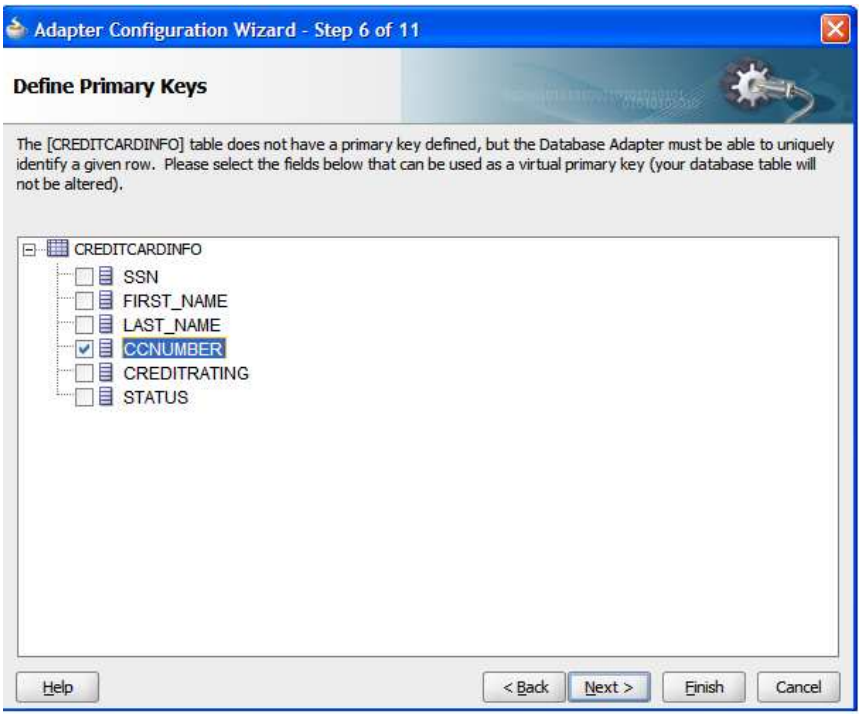
14. Click **OK**.

15. Back in step 5 of the **Database Adapter Configuration** wizard, click the **CREDITCARDINFO** table to select it.



16. Click **Next**.

17. Step 6 of the wizard lets you override or define the primary key for your table. In this case no primary key is defined in the database, so you'll need to specify it. Check CCNUMBER and leave the rest unchecked.

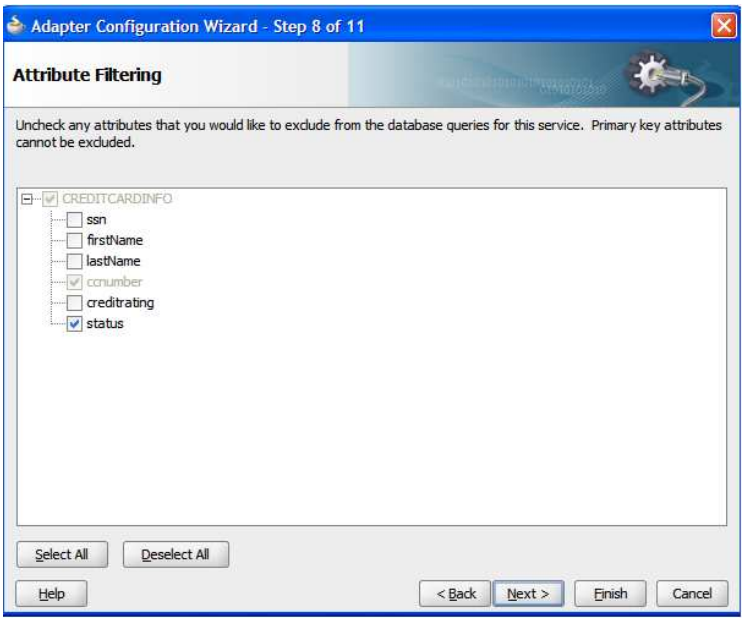


18. Click **Next**.

19. Step 7 lets you define relationships if you are selecting from multiple tables. Since there is only a single table there is no relationship to define.

20. Click **Next**.

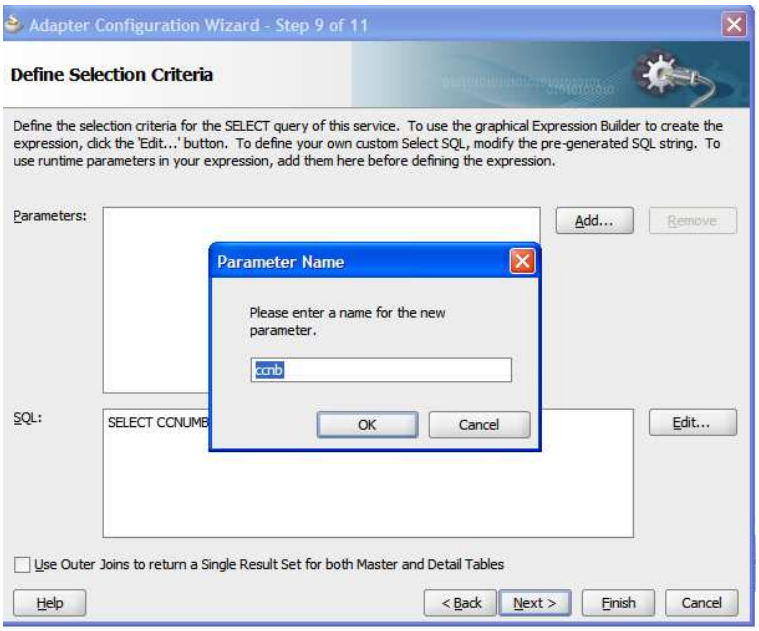
21. Uncheck all fields, in step 8, except **status**. That is the only column we want to query from the database.



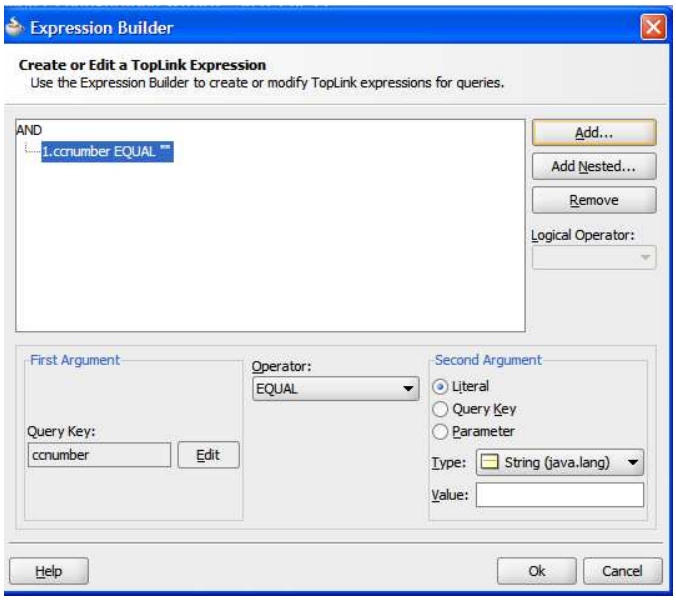
22. Click **Next**.

23. Step 9 is where you can specify your selection criteria. In this case you will add a parameter to the query. The parameter will be set at runtime to select the row for the credit card you want.

    Click the Add button to add a new parameter called **ccnb**. Click **OK**.

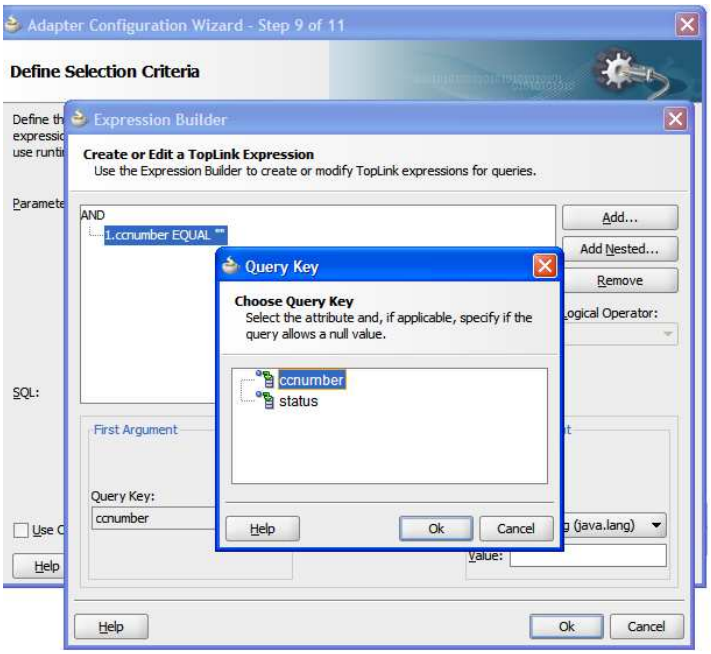


24. Back in step 9, click the **Edit** button to bring up the **Expression Builder** dialog and click the **Add** button to add a new condition

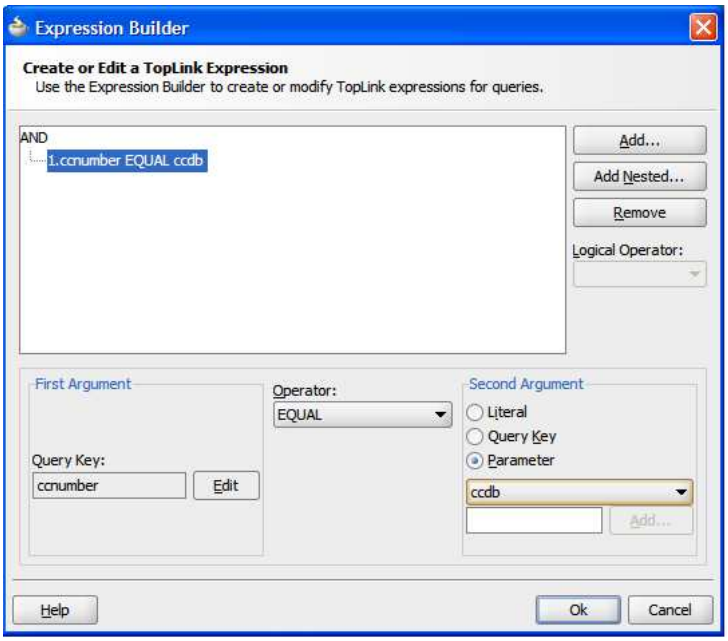


25. Press the **Edit** button in the **First Argument** section.

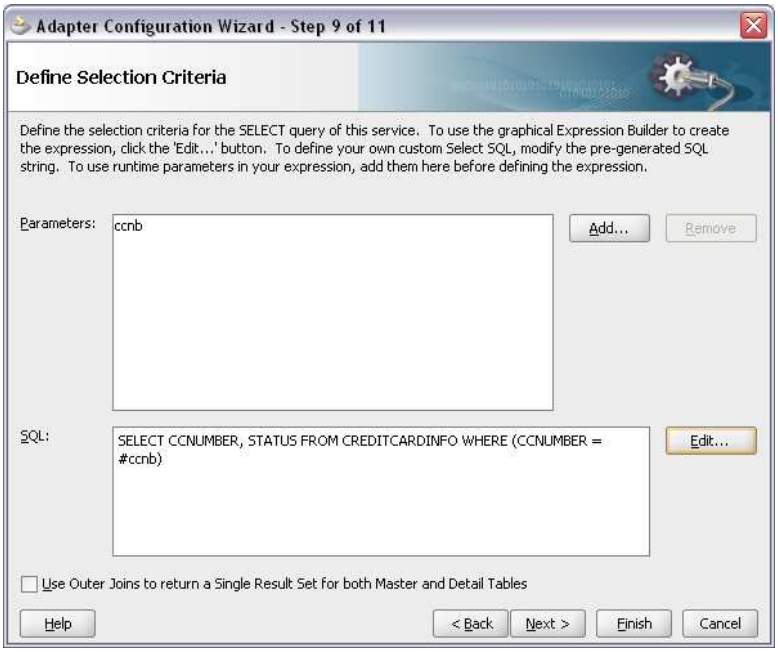
26. In the **Query Key** dialog select **ccnumber**.



27. Click **OK**.
28. In the **Second Argument** section, select **Parameter**.
29. Ensure **ccnb** is selected in the poplist.

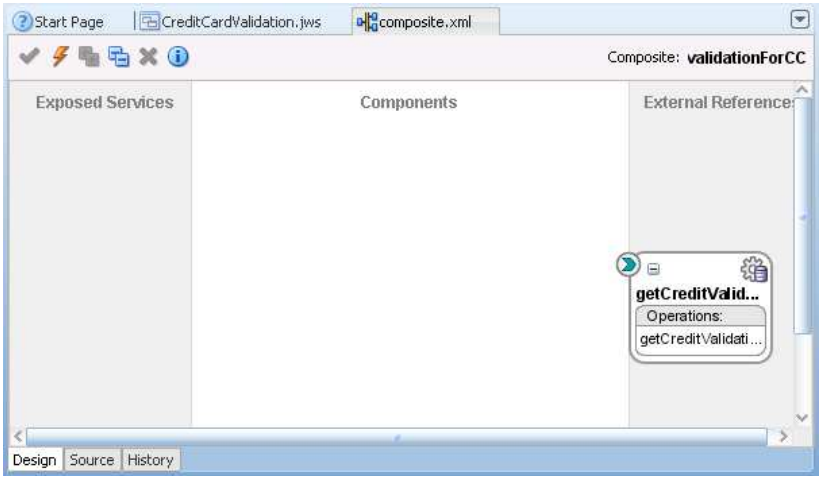


30. Click **OK**.
31. Back in step 9 you see the summary of the query that you specified. The parameter **ccnb** will be populated at runtime and the query will select a row based on that parameter



32. Click **Finish**.

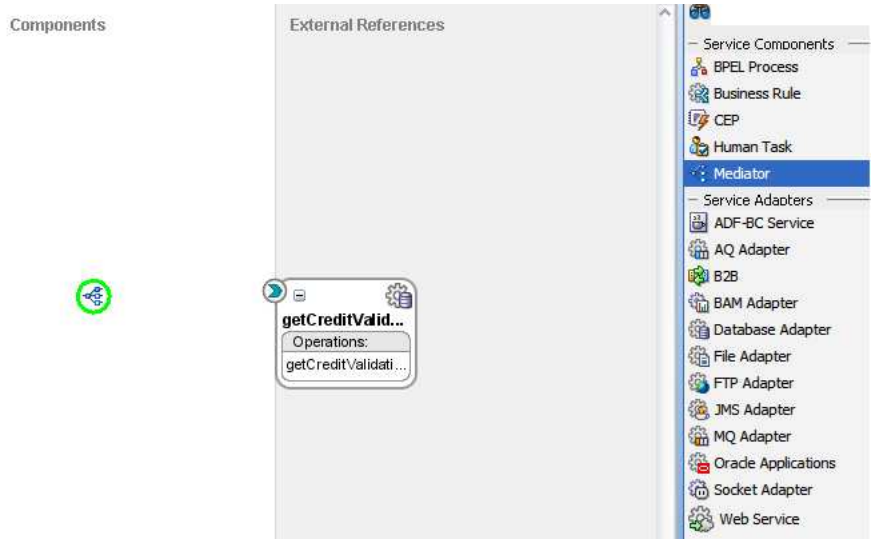
    The database adapter will process your choices and generate a service that implements the operation you specified, which means your project will now contain a new WSDL file to represent that service: `getCreditValidation.wsdl`. The composite diagram has been updated with this service.



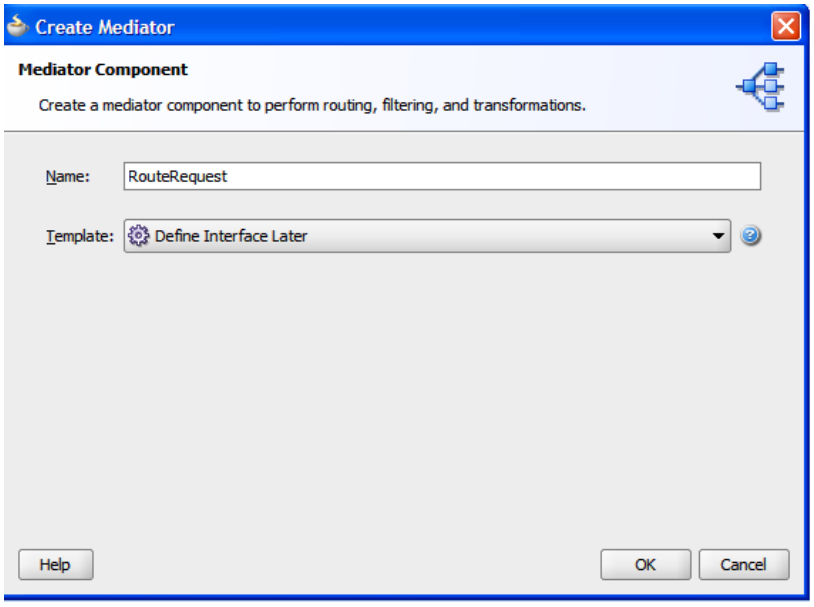
2.5 Adding the Mediator Component

The Mediator is the component in charge of routing in the SOA Suite. You will add a Mediator component to route a request to the service you just created with the database adapter.

1. Drag a Mediator component onto the composite diagram in the **Components** swim lane.

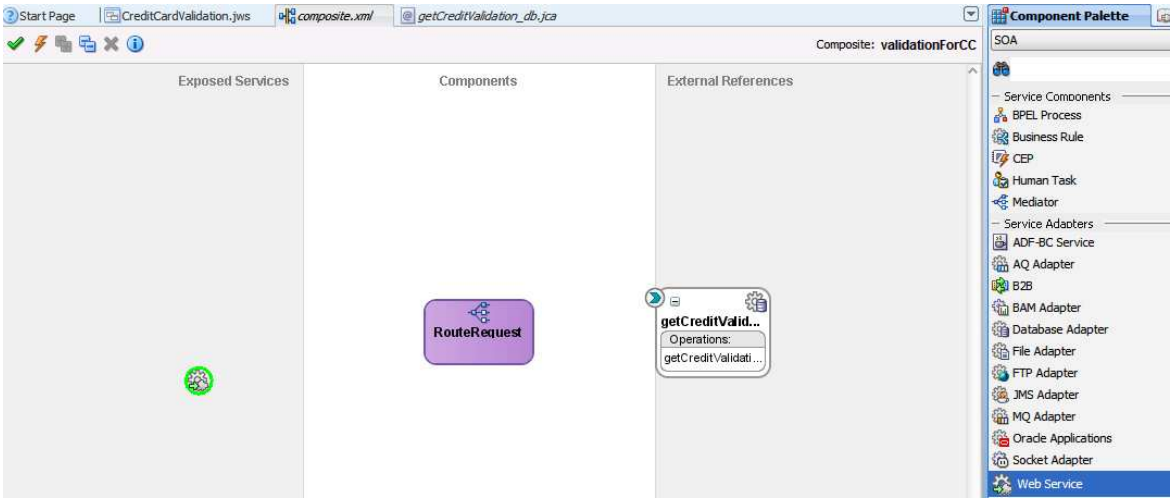


2. In the **Create Mediator** dialog, specify these settings:
   - **Name:** RouteRequest
   - **Template:** Define Interface Later

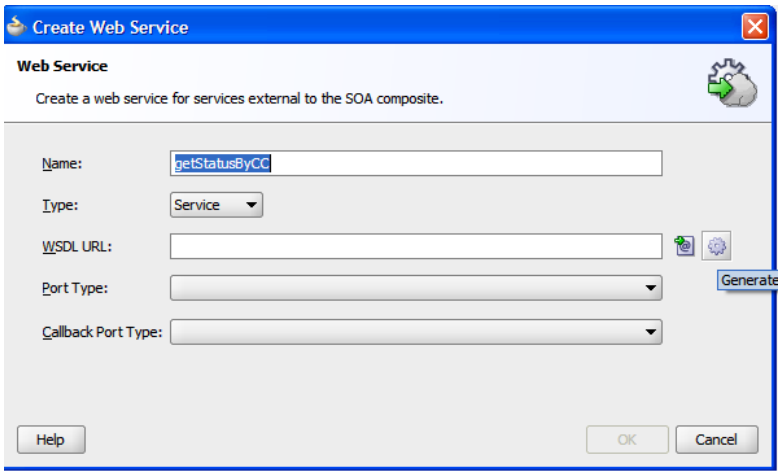


A typical composite application will have many components, but only some of them will have a publically exposed web services interface. The composite editor in JDeveloper gives you the flexibility to define the interface now, choose an existing interface, or to define the interface later.

3. Click **OK**.
4. You will create a Web Service interface to expose this service using SOAP bindings. Drag a Web Service adapter to the Exposed Services swim lane.

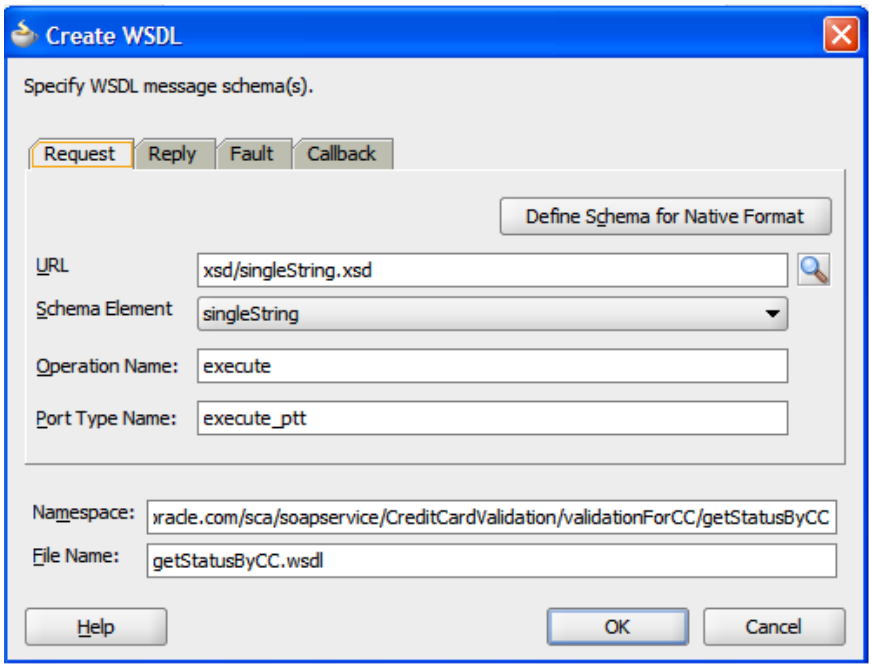


5. In the **Create Web Service** dialog that appears, set the following fields:
   - **Name:** getStatusByCC
   - **Type:** Service

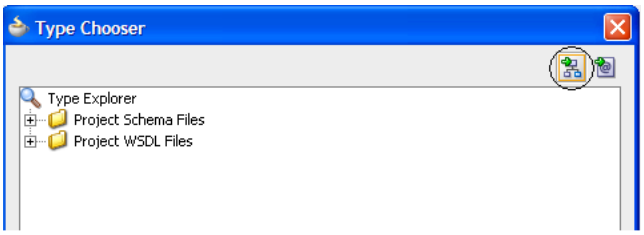


6. Click the cog icon next to the **WSDL File** field to define the interface.
7. The **Create WSDL** dialog lets you specify the message invocation types for the service. You have been given an existing XML schema definition (XSD) that specifies the types of the input and output for the service which you will reuse for this application.

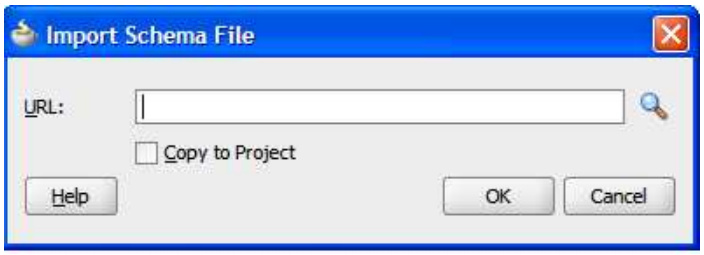
   Click the magnifying glass icon to the right of the **URL** field to browse for a schema file.



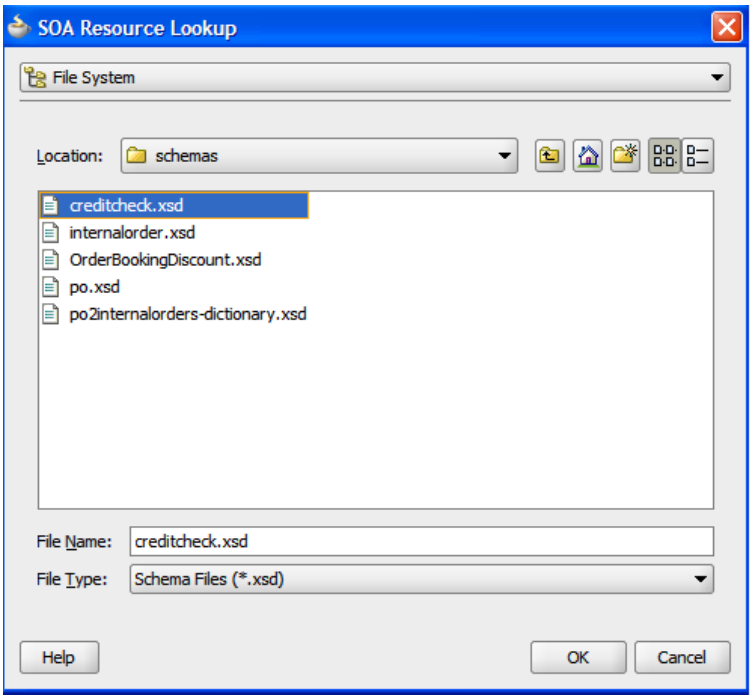
8. In the **Type Chooser** dialog, click the **Import Schema File** button.



9. In the **Import Schema File** dialog, click the magnifying glass to browse for the schema file.

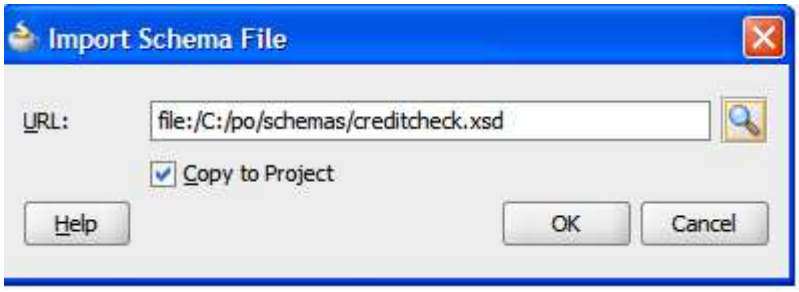


10. In the **SOA Resource Lookup**, make sure **File System** is selected.
11. Navigate to and select `c:\po\schemas\creditcheck.xsd`.



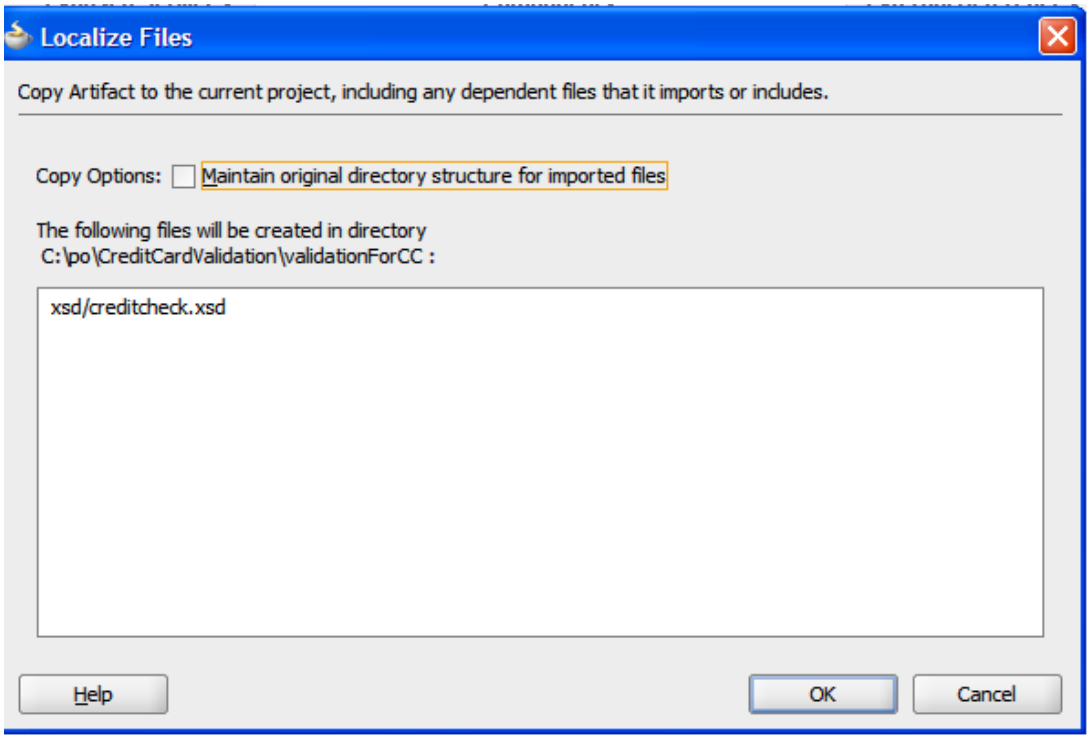
12. Click **Ok**.

13. Back in the **Import Schema File** dialog, make sure **Copy to Project** is selected.



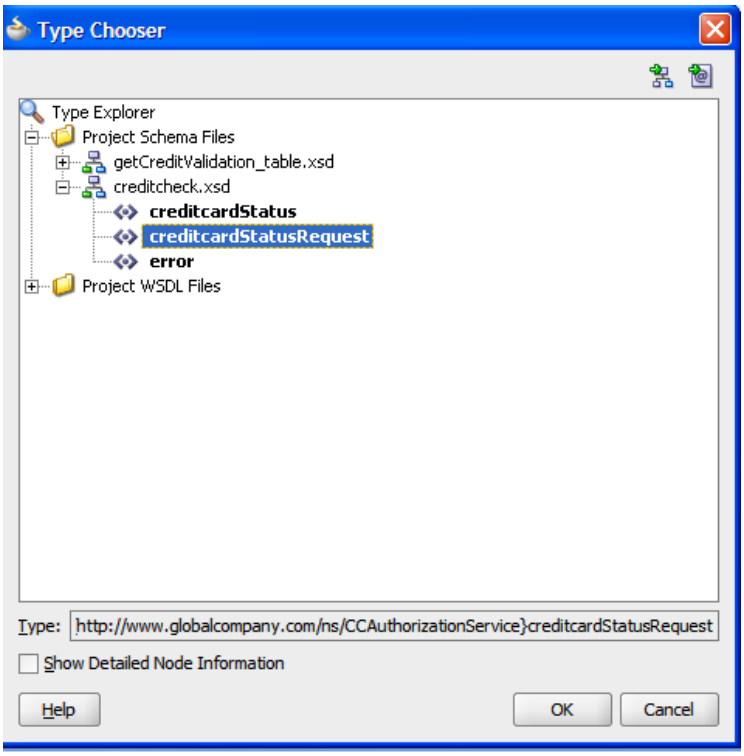
14. Click **OK**.

15. In the **Localize Files** dialog, deselect **Maintain original directory structure for imported files** and click **OK**.



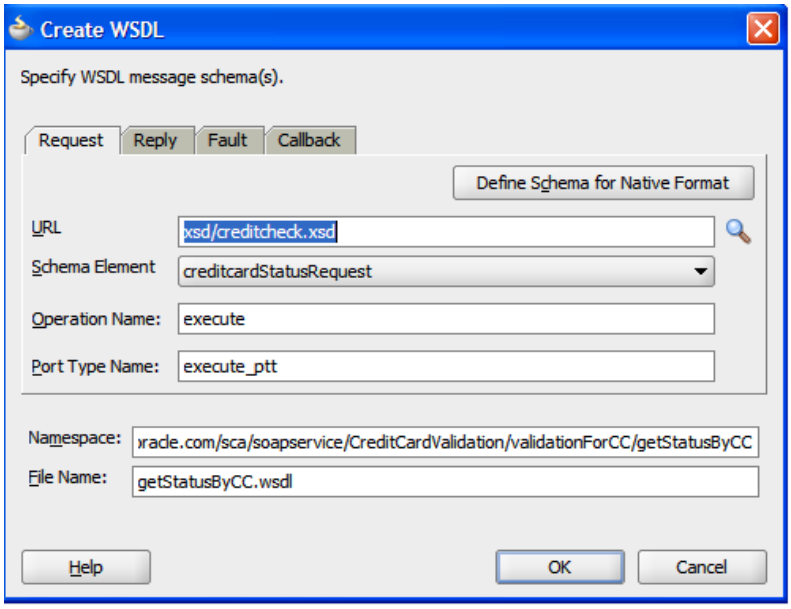
16. Back in the **Type Chooser** dialog, expand the **Project Schema Files** > **creditcheck.xsd** nodes.

17. Select **creditcardStatusRequest**.



18. Click **OK**.

19. The Create WSDL dialog is as shown. Do not click OK yet!



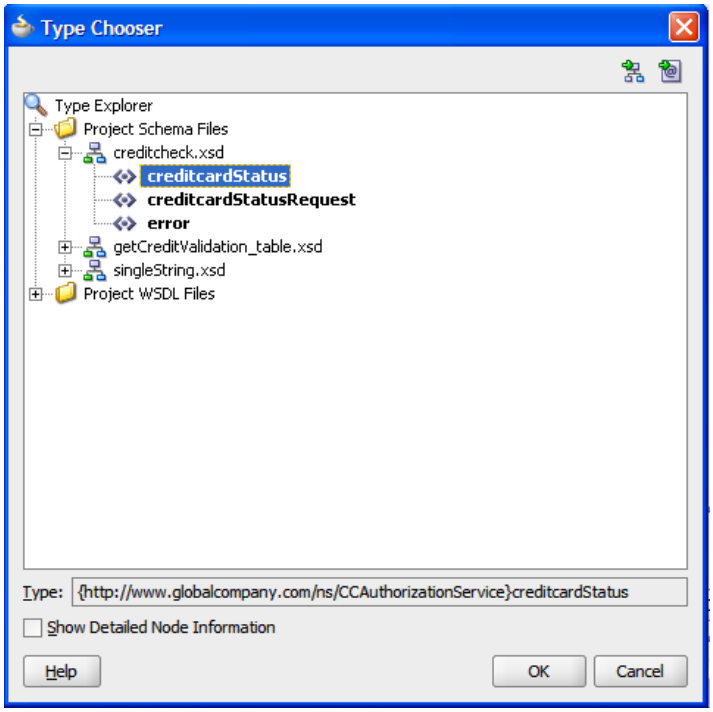
20. In a similar way, set the message types for the reply and fault to **creditcardStatus** and **error**. The only difference is that you don't need to import the schema definition again because it's now part of your project.

    Click the **Reply** tab.

21. Click the flashlight icon to the right of the **URL** field.

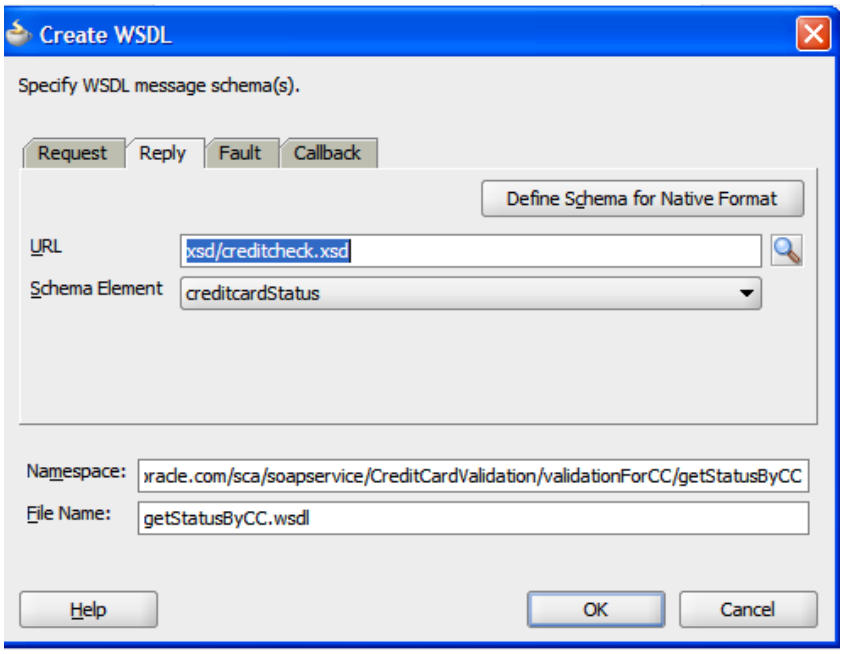
22. Expand the **Project Schema Files > creditcheck.xsd** nodes.

23. Select **creditcardStatus**.



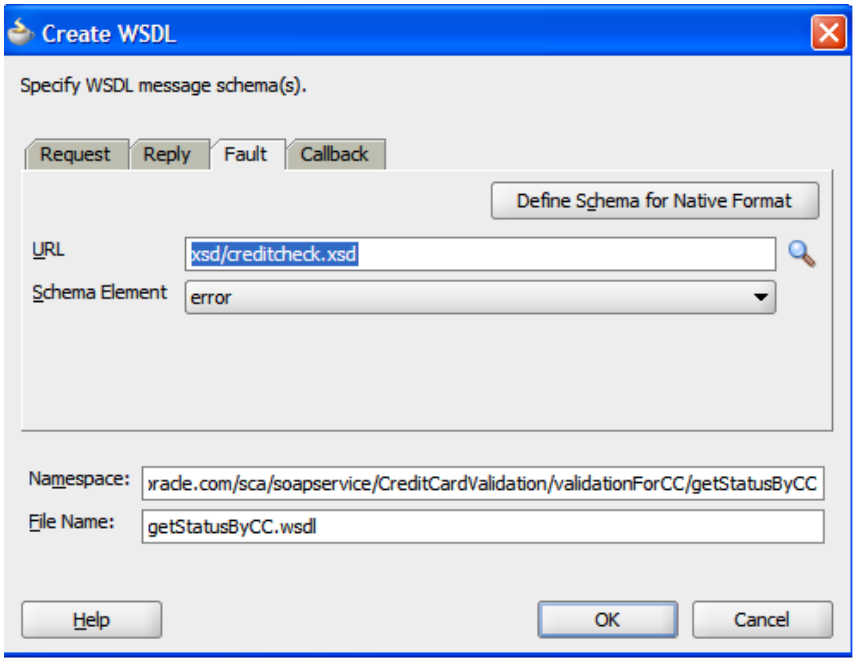
24. Click **OK**.

25. The reply message definition is as shown.



26. Click the **Fault** tab.

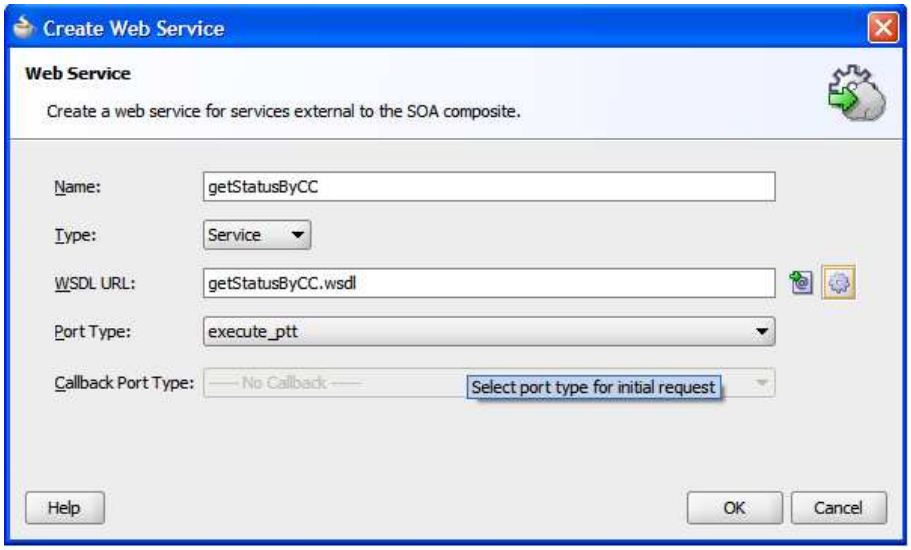
27. Using the same method, select the error type. The Fault message definition is as shown.



28. Click **OK** to close the **Create WSDL** dialog.

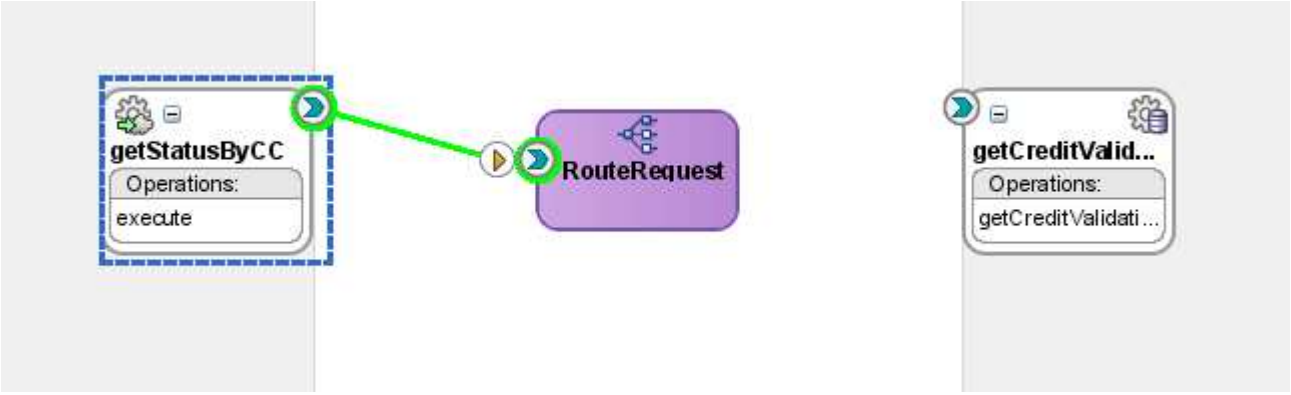
29. The **Create Web Service** dialog should now look like this.

    Be careful here -- If you click the cog icon again you will have to set all three invocation types again (request, reply, and fault).

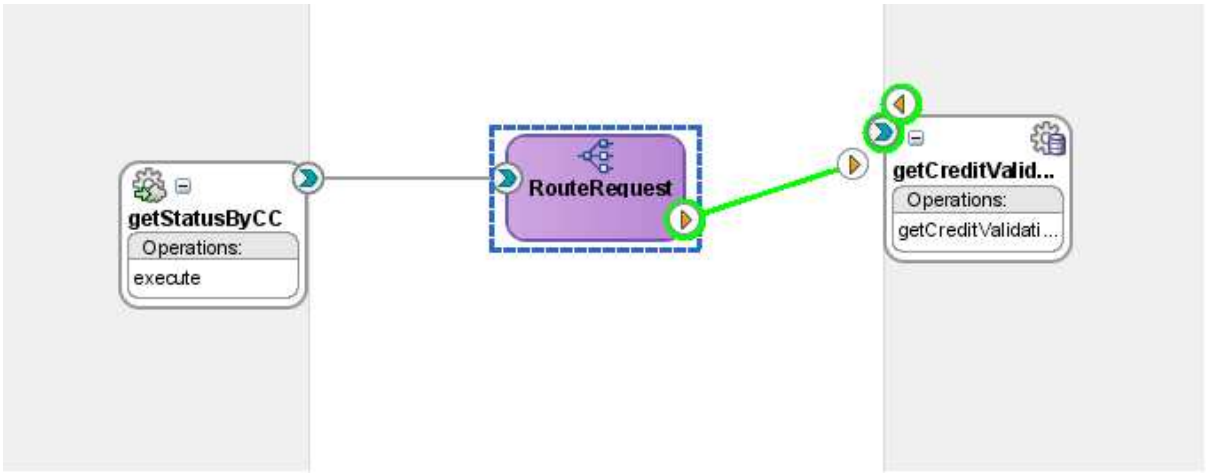


30. Click **OK**.

31. Now the components can be connected -- or "wired" -- together. Wire the inbound web service binding to the Mediator component:

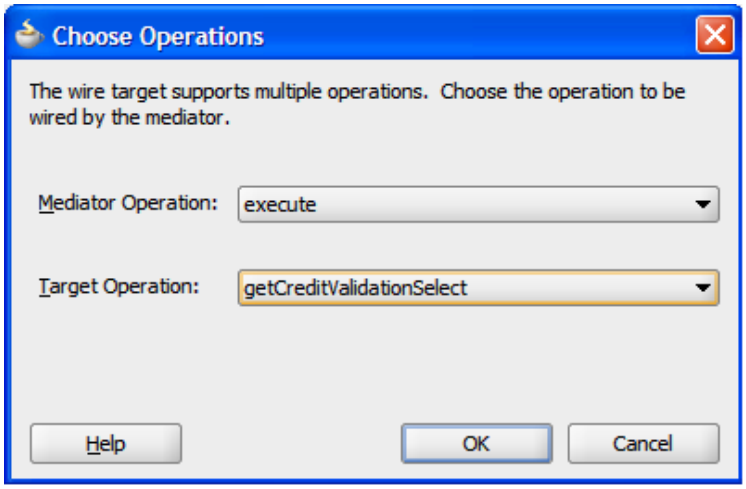


32. Wire the Mediator component to the database adapter service:



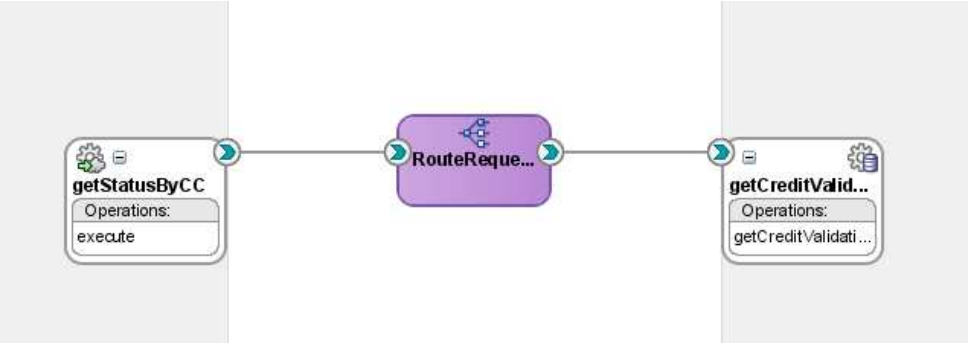
33. You may be prompted to choose the operations (only if you selected more than one database operations when you set up the database adapter service). If so, use these settings:

    - **Mediator Operation:** execute
    - **Target Operation:** getCreditValidationSelect



34. Click **OK**.

35. The composite diagram gives an overview of your application.

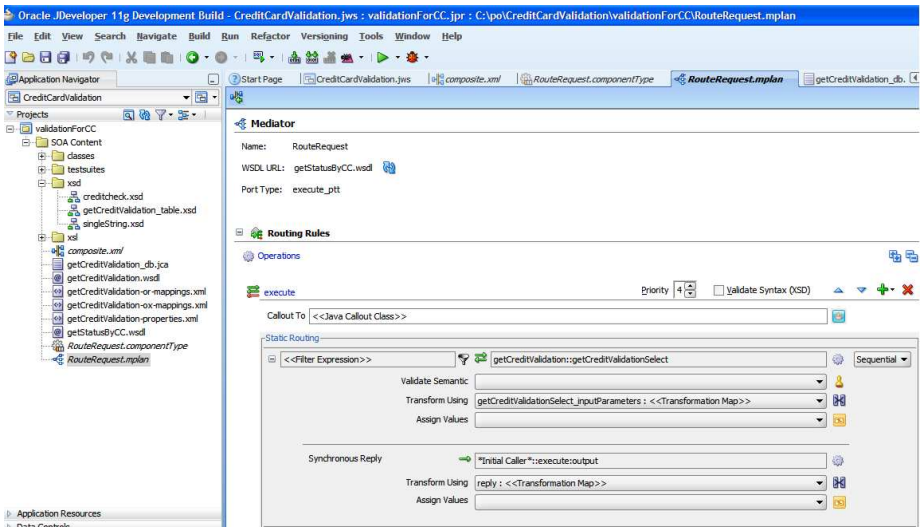


2.6 Adding a transformation to the Mediator component

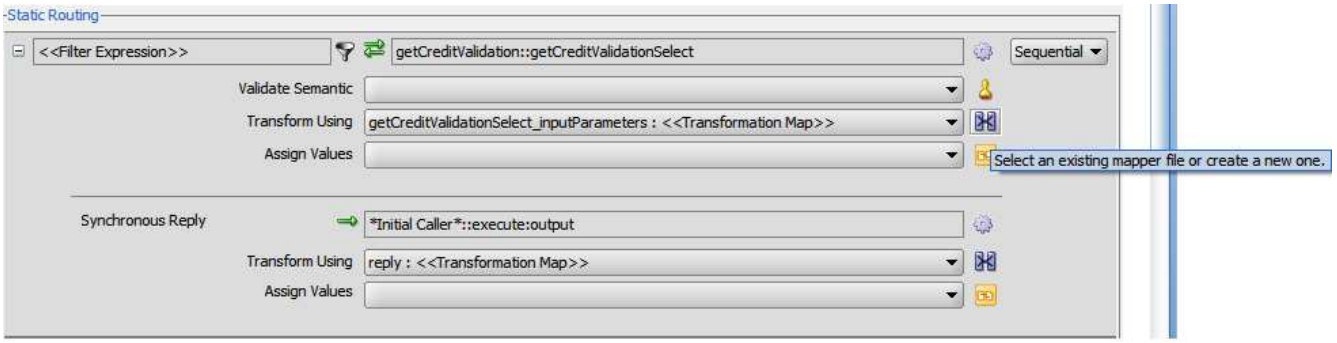
In this section you will modify the **RouteRequest** Mediator component to do an XSLT transformation on the message payload. That's because the incoming message to our publicly exposed service (**getStatusByCC**) is in a different format than the service created by the database adapter (**getCreditValidation**).

Not only does the Mediator route requests between endpoints (which you did by wiring them together) but it can also transform the data as it passes through.

1. One way to drill down into a specific component is to double-click it from the composite diagram. Double-click the **RouteRequest** Mediator component to open the Mediator editor.



2. Click the transformation icon to the right of the **Transform Using** field in the request section (that's the first **Transform Using** field).

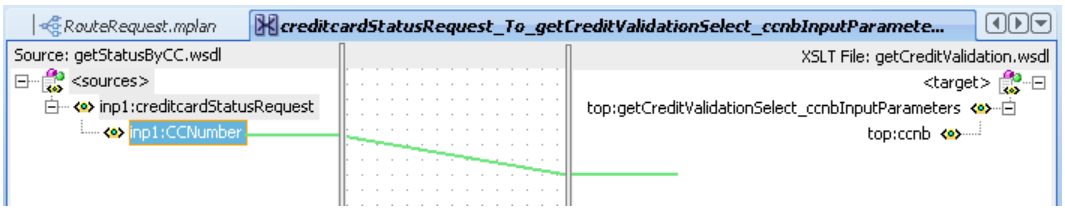


3. Select **Create New Mapper File** and accept the default name.

4. Click **OK** to open the mapping editor.

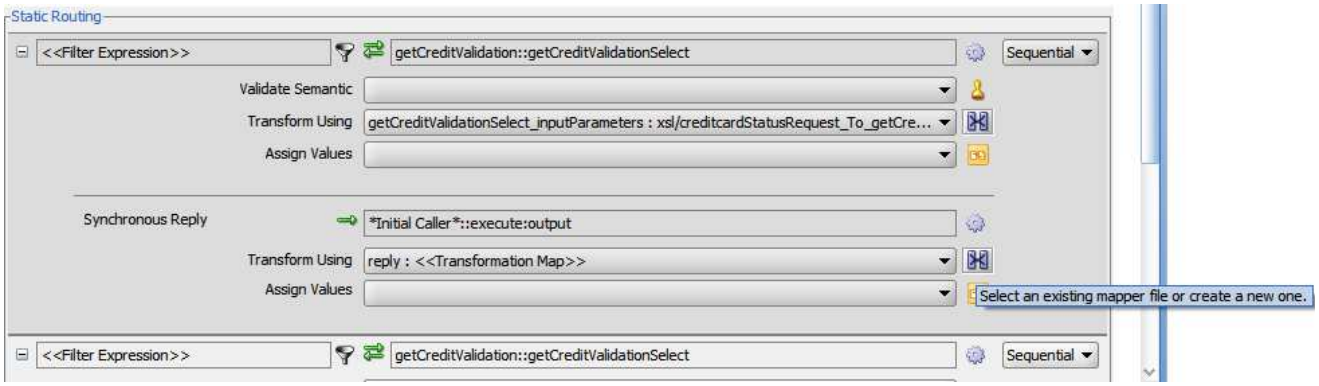
5. Expand all the nodes on both sides. You can do it manually, or right-click **<sources>** and select **Expand All**. Do the same for **<target>** on the right-hand side.

6. Map **CCNumber** from the source side to **ccnb** on the target side by dragging a wire from one element to the other.



7. Save and close the mapper by selecting the close window icon on the right side of the tab. You may need to use the tab bar scroll buttons to navigate right to get to the close window icon. Alternatively you can press Ctrl-S to save the mapping, then Ctrl-W to close it.

8. Back in the Mediator editor, click the transformation icon to the right of the **Transform Using** field in the reply section (that's the second **Transform Using** field).

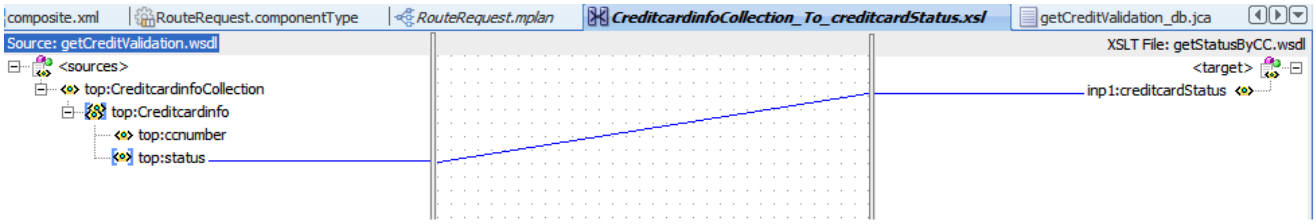


9. In the **Reply Transformation Map** dialog select **Create New Mapper File** and accept the default name.

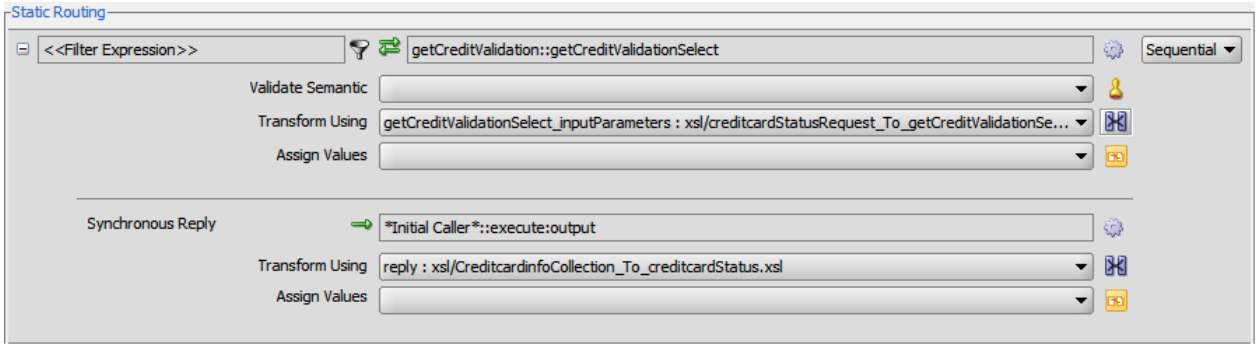
10. Click **OK** to open the mapper.

11. Expand all the source and target nodes.

12. Map **status** from the source to **creditcardstatus** from the target.



13. Save and close the mapper to return to the Mediator editor.



14. Save and close the Mediator editor to return to the composite diagram. You can use the toolbar buttons or the menu, or simply press Ctrl-S to save and Ctrl-W to close this tab.

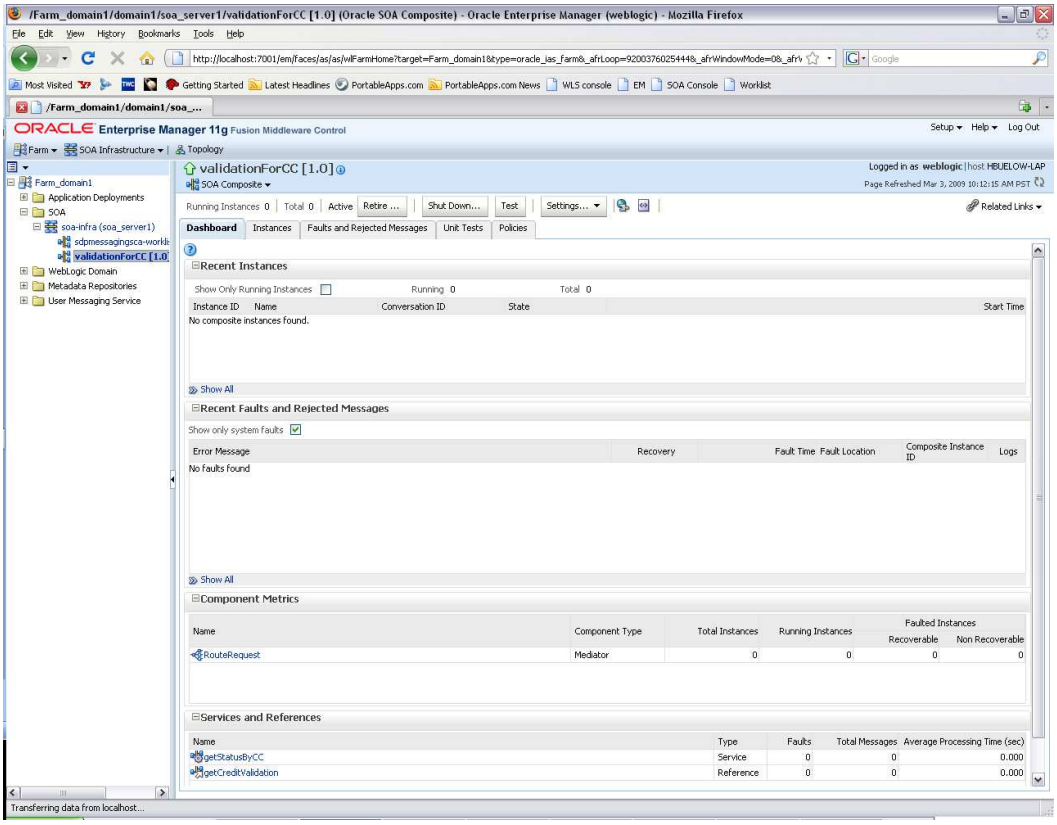
2.7 Deploying the application

The first design iteration is complete and you are now ready to deploy the composite. The steps to run and test a composite application are explained in **Appendix A Deploying and Running a Composite Application** Read that document now if this is the first time you are running and testing a composite application. **Appendix A** follows **Chapter 9** in the tutorial.

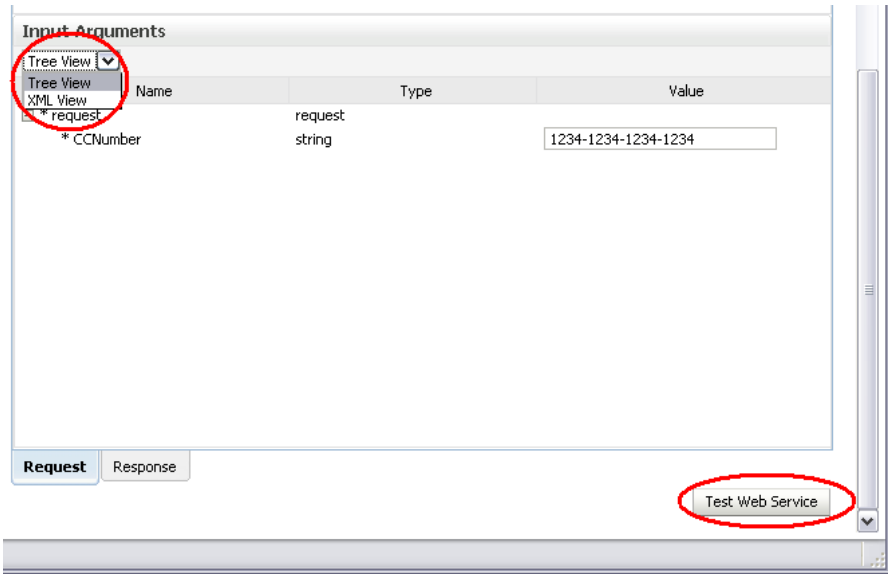
2.8 Testing the application

If you are not sure how to deploy, read the steps in **Appendix A Deploying and Running a Composite Application**.

1. After you have deployed your composite, open Enterprise Manager at the following link: http://localhost:7001/em
2. Click the **validationForCC** link in the **SOA** section.



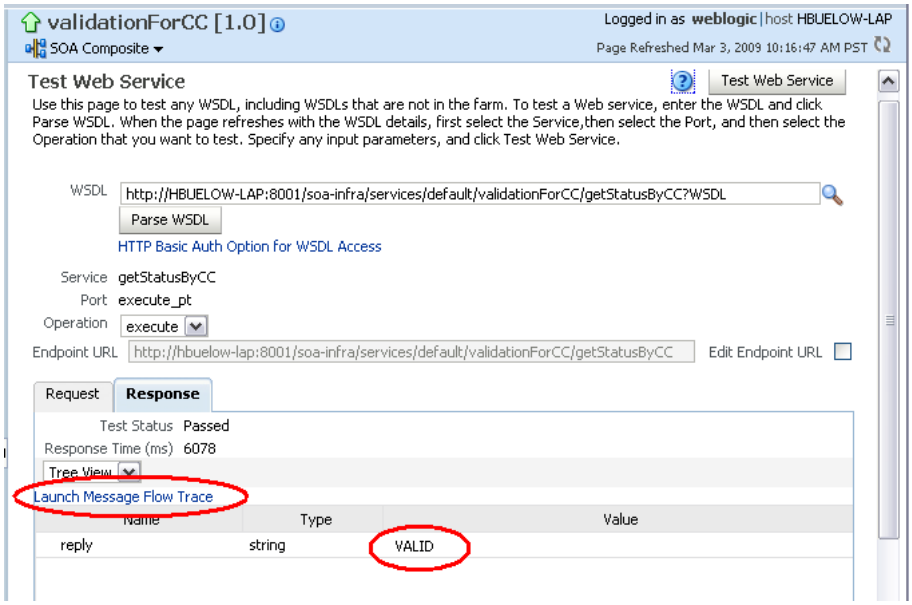
3. Open the Test page by clicking the **Test** button



4. In the **CCNumber** field, enter a credit card number. In this case, enter **1234-1234-1234-1234** in the **CCNumber** field which is a valid credit card and returns the status, 'VALID.'

5. Click the **Test Web Service** button.

6. You will get a response from the service which will indicate if the credit card is valid or not. Here is the output for a valid credit card:



7. Click **Launch Message Flow Trace** to see the details of the message flow for this instance of your composite. This pops open a new window with the flow trace.

8. Close the **Flow Trace** window.

9. Try another value. Click the **Request** tab, and this time enter **4321-4321-4321-4321** in the **CCNumber** field which is an invalid credit card.

10. Click the **Test Web Service** button to see the output.

11. In an upcoming chapter, you will create a new application that calls this service, so you'll need to know the WSDL location for it. You can see the WSDL location listed at the top of the Test page.

    In this case it looks something like:

    http://localhost:8001/soa-infra/services/CreditCardValidation/validationForCC/getStatusByCC?WSDL

2.9 Operations and naming

This section gives you all of the operations and names for objects created in this chapter. Experienced users can use this for creating the objects in this chapter quickly. Any questions on details for a particular operation listed here can be found in the preceding sections. The information is divided by the sections in this document.

Section 2.3 Creating a new application

- **Application Name:** CreditCardValidation

- **Directory:** C:\po\CreditCardValidation
- **Project Name:** validationForCC
- **Directory:** C:\po\CreditCardValidation\validationForCC
- **Project Technologies:** SOA
- **Empty Composite**

Section 2.4 Adding the database adapter

- **DbAdapter Service Name:** getCreditValidation
- **Create Database Connection** details:
  - **Connection Name**: soademoDatabase
  - **Connection Type**: Oracle (JDBC)
  - **Username**: soademo
  - **Password**: soademo
  - **Save Password**: Checked
  - **Enter Custom JDBC URL**: Unchecked
  - **Driver**: thin
  - **Host Name**: localhost
  - **JDBC Port**: 1521 (or the port number of your database)
  - **SID**: XE (or the SID of your database)
  - **Jndi name**: eis/DB/soademoDatabase
- **Operation**: Select
- **Import** CREDITCARDINFO table
- **Primary key** CCNUMBER
- **Attribute Filtering**: status
- **Add a new parameter**: ccnb
- **Add Where clause**: ccnumber = #ccnb

Section 2.5 Adding the Mediator Component

- **Create Mediator:** RouteRequest
- **Template:** Define Interface Later
- **Create Web Service:** getStatusByCC
- **Type:** Service
- **WSDL based on:** c:\po\schemas\creditcheck.xsd
- **Request:** creditcardStatusRequest

- **Reply:** creditcardStatus
- **Fault:** error
- **Wire:** service to mediator to adapter

Section 2.6 Adding a transformation to the Mediator component

- **Mediator transform:** Map CCNumber from the source side to ccnb
- **Mediator transform for reply:** Map status from the source to creditcardstatus from the target.

The application is completed. Continue with Section 2.7 above to deploy and test your application.