Practices for Lesson 5: Overview

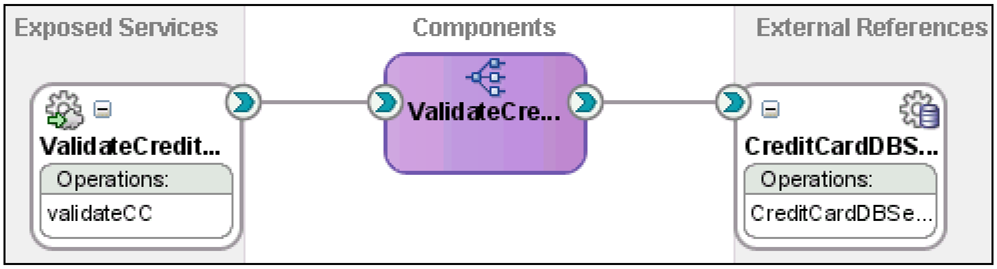
Practices Overview

In this practice, you create a composite application that receives a credit card number and a purchase order total. The application uses a Database adapter to query credit card information in an Oracle Database. Based on the results of that query, a status of VALID or INVALID is returned.

To create this application, perform the following steps:

- You create a new SOA project with a Database adapter that is configured to perform a `select` operation on the database.
- You then add a synchronous Mediator, defining a WSDL interface based on the elements of the `creditcheck.xsd` file.
- You deploy the composite application and test it with both valid and invalid credit card numbers.

The following image represents the composite application design for your CCValidate project:



Practice 5-1: Configuring the Database

Overview

The SOA Quickstart installation includes a JavaDB (Derby) database. In this section, you create and populate tables for use in this and subsequent practices.

Assumptions

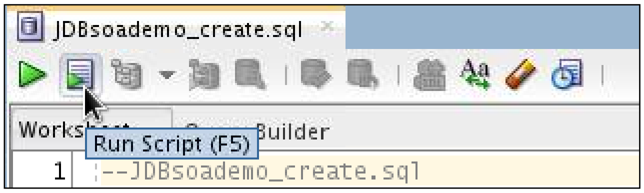
This practice assumes that JDeveloper is open and that WebLogic Server is started.

Note: Use *exact* names (both spelling and case are important) as specified for your project and components. The practice for the lesson titled "Sharing Functionality in Oracle SOA Suite" will expect to find these names in use.

Tasks

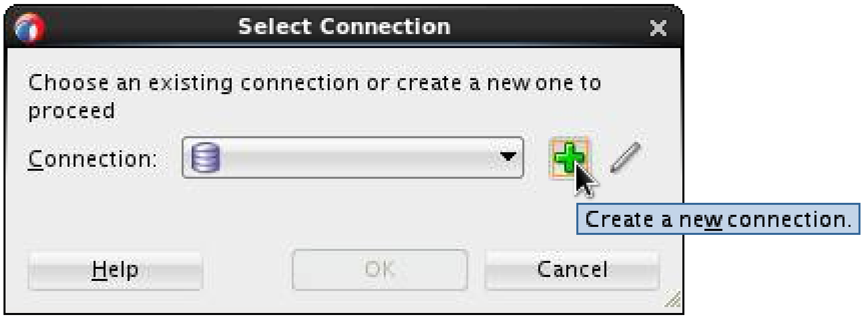
1. From the JDeveloper menu, select File > Open.
2. Navigate to `/home/oracle/labs/scripts/SQL`.
3. Select `JDBsoademo_create.sql`.

   The file is opened in a SQL worksheet.
4. Click the Run Script icon.

   ```
   1 --JDBsoademo_create.sql
   ```

   The Select Connection dialog box is displayed.
5. Click the "Create a new connection" icon.

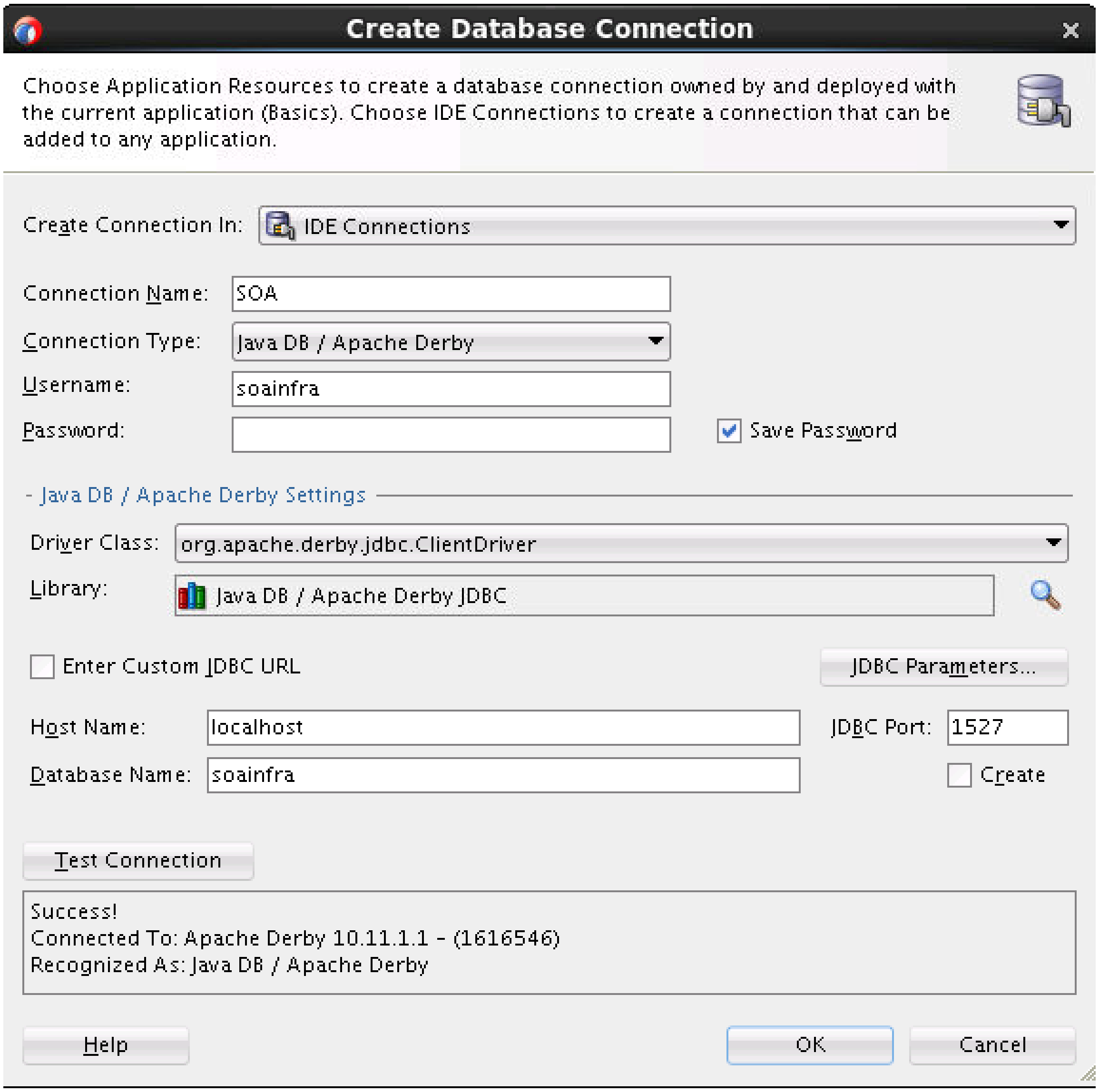
   Choose an existing connection or create a new one to proceed
6. Enter the following details:
   - Connection Name: **SOA**
   - Connection Type: **Java DB (Derby)**
   - Username: **soainfra**
   - Password: **<leave blank>**
7. Ensure that the default Java DB (Derby) settings match the following:
   - Host Name: `localhost`

- JDBC Port: `1527`
- Database Name: `soainfra`
- Library: Java DB JDBC Driver

Note: You may need to select the library from the drop-down list. You may also need to select the `org.apache.derby.jdbc.ClientDriver` for the Driver class.

8. Click the **Test Connection** button to verify that the connection works.

   The following message is displayed: Success!



Create Database Connection

Choose Application Resources to create a database connection owned by and deployed with the current application (Basics). Choose IDE Connections to create a connection that can be added to any application.

Create Connection In: IDE Connections

Connection Name: SOA

Connection Type: Java DB / Apache Derby

Username: soainfra

Password:

☑ Save Password

- Java DB / Apache Derby Settings

Driver Class: org.apache.derby.jdbc.ClientDriver

Library: Java DB / Apache Derby JDBC

□ Enter Custom JDBC URL

Host Name: localhost

JDBC Port: 1527

Database Name: soainfra

□ Create

```
Success!
Connected To: Apache Derby 10.11.1.1 - (1616546)
Recognized As: Java DB / Apache Derby
```

9. Click OK.
10. Click OK to close the Select Connection dialog box.

    The script is run.
11. Open and run the `/home/oracle/labs/scripts/SQL/JDBsoademo_data.sql` file.

    When prompted, select the SOA connection that you defined a moment ago.

12. Open and run the `/home/oracle/labs/scripts/SQL/JDBcreate_waitingorders.sql` file.
13. Close each of the `.sql` file tabs that you opened.

Practice 5-2: Creating the CCValidate SOA Project

Overview

In this practice, you create a new SOA project in the Basics workspace, which contains a Database adapter.

Assumptions

This practice assumes that JDeveloper is open and that WebLogic Server is started.

Note: Use exact names (both spelling and case are important) as specified for your project and components. The practice for the lesson titled "Sharing Functionality in Oracle SOA Suite" will expect to find these names in use.

Tasks

1. In JDeveloper, create a new empty SOA project named `CCValidate` in the Basics application.

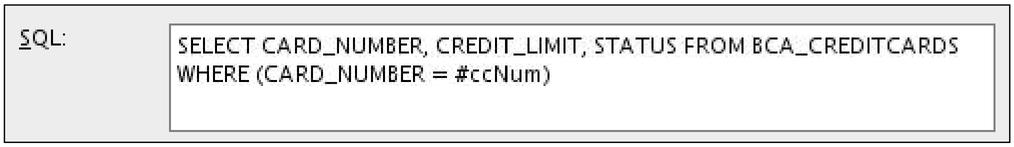
   The composite overview window opens.
2. Add a Database adapter to the External References swimlane.

   The Adapter Configuration wizard appears.
3. To configure the Database adapter, perform the following steps:

Step	Screen	Choices or Values
a.	Database Adapter Reference	Name: `CreditCardDBService` Click Next.
b.	Service Connection	Use the Browse option to copy the SOA service connection. Verify that the JNDI Name has the value `eis/DB/SOA`. Click Next.
c.	Operation Type	Select the "Perform an operation on a Table" option. Ensure that the Select check box is selected. Deselect the other operations. Click Next.
d.	Select Table	Click Import Tables.
e.	Import Tables	Set the filter to BCA_%. Click Query.
f.	Import Tables	Move the `BCA_CREDITCARDS` table from the Available list to the Selected list. Click OK.
g.	Select Table	Verify that the `BCA_CREDITCARDS` table is listed.

Step	Screen	Choices or Values
		Click Next.
h.	Relationships	Click Next.
i.	Attribute Filtering	Select the `creditLimit` and `status` attributes. Deselect the remaining options. **Note:** The cardNumber (key attribute) cannot be deselected. Click Next.
j.	Define Selection Criteria	Next to the Parameters section, click Add.
k.	Parameter Name	Enter `ccNum`. Click OK.
l.	Define Selection Criteria	Next to the SQL field, click Edit.
m.	Expression Builder	Click Add.
n.	Expression Builder	Form the condition: `cardNumber` EQUAL `ccNum`, where: First Argument Query Key: `cardNumber` Operator: `EQUAL` Second Argument: Select the Parameter option, and select `ccNum`. Click OK.
o.	Define Selection Criteria	Verify that your SQL statement is correct by comparing it to the screenshot that follows this table. Click Next.
p.	Advanced Options	Click Next.
q.	JCA Endpoint Properties	Click Next.
r.	Finish	Click Finish.

The Adapter wizard closes.



SQL:

```
SELECT CARD_NUMBER, CREDIT_LIMIT, STATUS FROM BCA_CREDITCARDS
WHERE (CARD_NUMBER = #ccNum)
```

Practice 5-3: Configuring the Mediator Component from a WSDL Definition

Overview

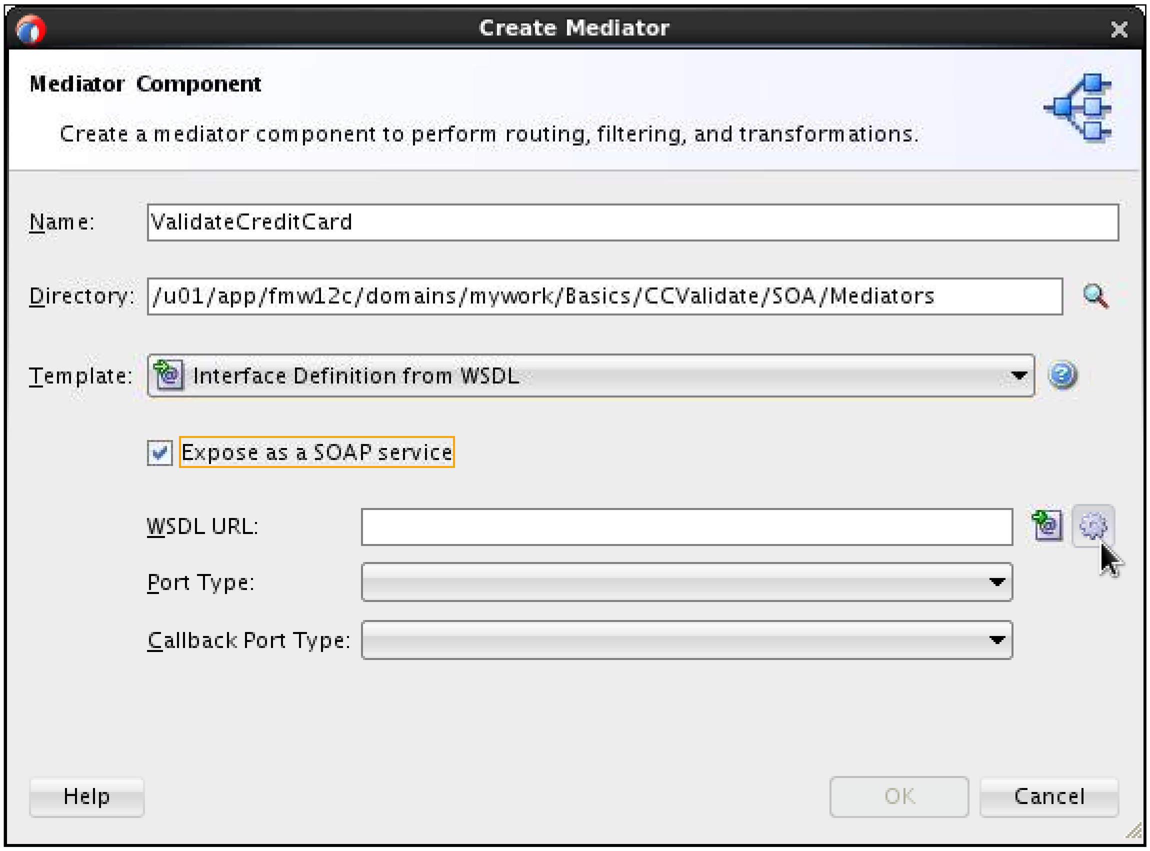
In this practice, you create a synchronous Mediator. You define its WSDL interface based on the elements of the `creditcheck.xsd` file. The `.xsd` file describes both the message that the Mediator will receive from the client that calls the application and the message that it will return to that caller.

Assumptions

This practice assumes that you have successfully completed Practice 5-2.

Tasks

1. Create and configure a Mediator component.
   a. Add a Mediator to the Component column of the composite application editor.
      The Create Mediator window opens.
   b. Set the Name to `ValidateCreditCard`.
   c. Select "Interface Definition from WSDL" from the Template drop-down list.
   d. Ensure that the "Expose as a SOAP service" check box is selected.
   e. Click the "Generate WSDL from Schema(s)." icon (next to the WSDL URL field).



Create Mediator

Mediator Component

Create a mediator component to perform routing, filtering, and transformations.

Name: ValidateCreditCard

Directory: /u01/app/fmw12c/domains/mywork/Basics/CCValidate/SOA/Mediators

Template: Interface Definition from WSDL

☑ Expose as a SOAP service

WSDL URL:

Port Type:

Callback Port Type:

The Create WSDL window opens.

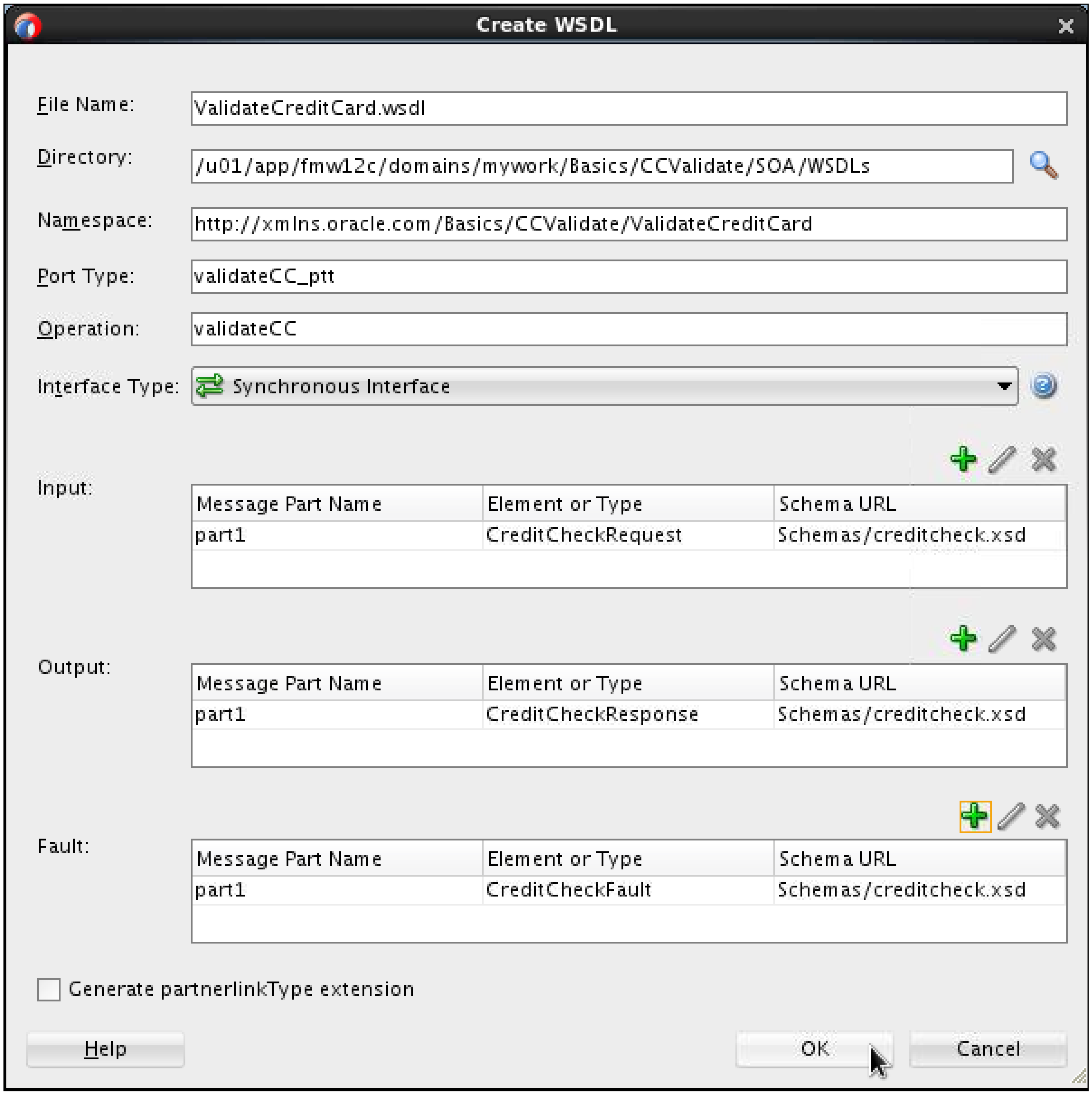
2. Define the Mediator WSDL Interface by modifying the following field values:

Port Type:	`validateCC_ptt`
Operation:	`validateCC`
Interface Type:	Synchronous

3. Define the request, response, and fault messages for the Mediator component.

Step	Screen	Choices or Values
a.	Create WSDL	Click the **Input** Add a new Message Part icon.
b.	Add a new Message Part	Click the Browse icon next to the URL field.
c.	Type Chooser	Click the Import Schema icon.
d.	SOA Resource Browser	Using the File System option, locate and select the `creditcheck.xsd` file in the `/home/oracle/labs/files/xsd` folder. Click OK.
e.	Localize Files	Verify that `creditcheck.xsd` is selected. Click OK.
f.	Type Chooser	Expand the Project Schema Files > `creditcheck.xsd` nodes (if needed). Select the CreditCheckRequest element. Click OK.
g.	Add a new Message Part	Click OK.
h.	Create WSDL	Click the **Output** Add a new Message Part icon.
i.	Add Message Part	Click the Browse icon next to the URL field.
j.	Type Chooser	Expand the Project Schema Files > `creditcheck.xsd` nodes (if needed) and select the CreditCheckResponse element. Click OK.
k.	Add a new Message Part	Click OK.
l.	Create WSDL	Click the **Fault** Add a new Message Part icon. **Note:** The subject of faults is covered in the lesson titled “Handling Faults in Composite Applications.”
m.	Add Message Part	Click the Browse icon next to the URL field.

Step	Screen	Choices or Values
n.	Type Chooser	Expand the Project Schema Files > `creditcheck.xsd` nodes (if needed) and select the `CreditCheckFault` element. Click OK.
o.	Add Message Part	Click OK.
p.	Create WSDL	Compare your work to the screenshot that follows this table. Click OK.



Create WSDL

File Name: ValidateCreditCard.wsdl

Directory: /u01/app/fmw12c/domains/mywork/Basics/CCValidate/SOA/WSDLs

Namespace: http://xmlns.oracle.com/Basics/CCValidate/ValidateCreditCard

Port Type: validateCC_ptt

Operation: validateCC

Interface Type: Synchronous Interface

Input:

Message Part Name	Element or Type	Schema URL
part1	CreditCheckRequest	Schemas/creditcheck.xsd

Output:

Message Part Name	Element or Type	Schema URL
part1	CreditCheckResponse	Schemas/creditcheck.xsd

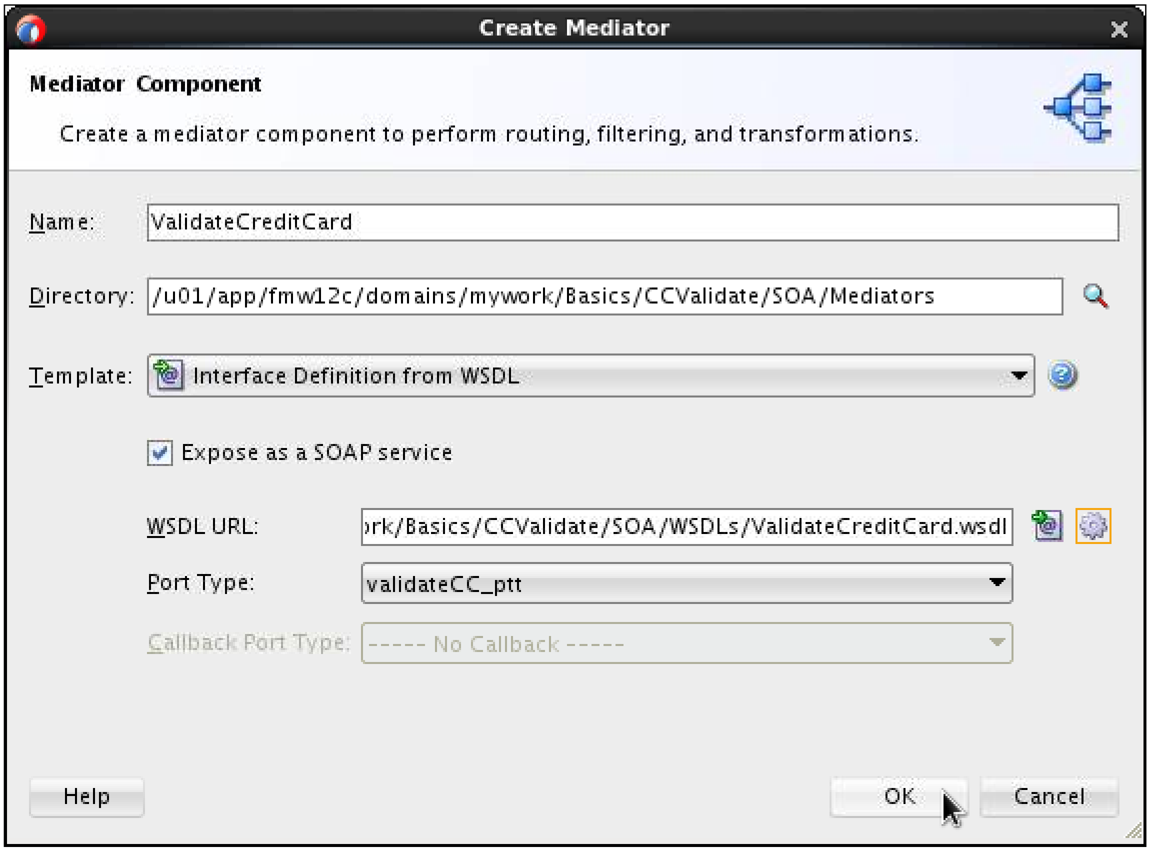
Fault:

Message Part Name	Element or Type	Schema URL
part1	CreditCheckFault	Schemas/creditcheck.xsd

☐ Generate partnerlinkType extension

You are returned to the Create Mediator window.

4. Verify your work and click OK.



Create Mediator

Mediator Component

Create a mediator component to perform routing, filtering, and transformations.

Name: ValidateCreditCard

Directory: /u01/app/fmw12c/domains/mywork/Basics/CCValidate/SOA/Mediators

Template: Interface Definition from WSDL

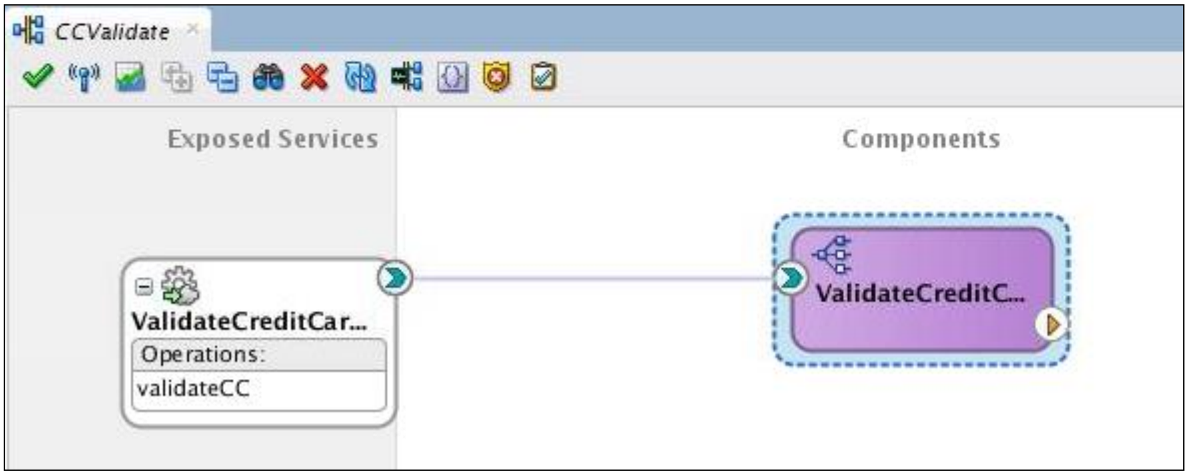
☑ Expose as a SOAP service

WSDL URL: ›rk/Basics/CCValidate/SOA/WSDLs/ValidateCreditCard.wsdl

Port Type: validateCC_ptt

Callback Port Type: ----- No Callback -----

5. Verify and save your work.



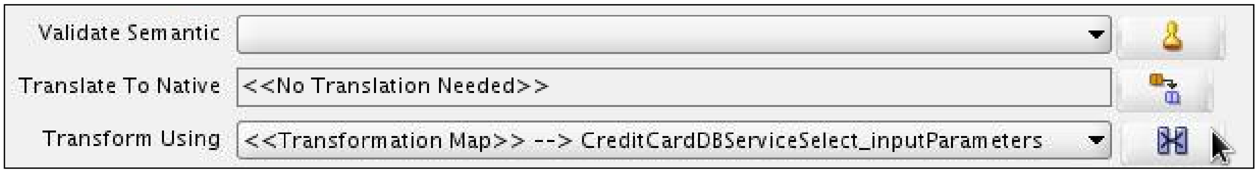
Wiring the Mediator to the DB Adapter and Creating Transformations

6. Create a wire from the Mediator component to the Database adapter.

7. Edit the Mediator component to add transformations, by right-clicking the Mediator icon and selecting Edit.

   The `ValidateCreditCard.mplan` window opens.

8. Expand the Static Routing rule that was created by the wire.
9. In the upper (request) section of the Static Routing rule, click the "Select an existing mapper file or create a new one" icon.

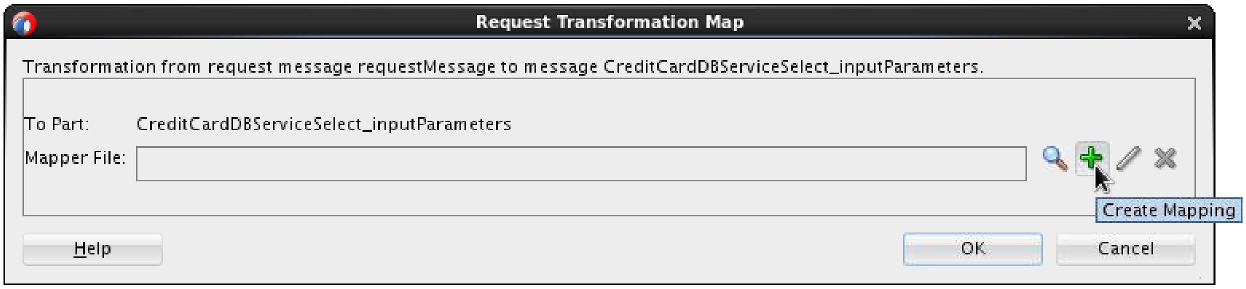


Validate Semantic

Translate To Native <<No Translation Needed>>

Transform Using <<Transformation Map>> --> CreditCardDBServiceSelect_inputParameters

   a. In the Request Transformation Map dialog box, click the Create Mapping icon.



Request Transformation Map

Transformation from request message requestMessage to message CreditCardDBServiceSelect_inputParameters.

To Part: CreditCardDBServiceSelect_inputParameters

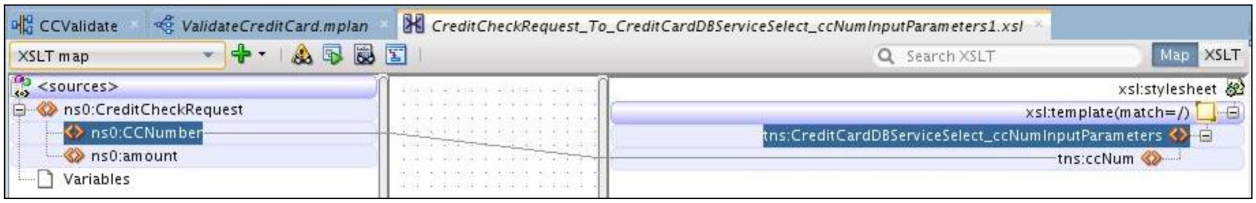
Mapper File:

   The Create Transformation Map dialog box is displayed.

   b. Accept the default file name supplied. Click OK.
   c. Click OK to close the Request Transformation Map dialog box.

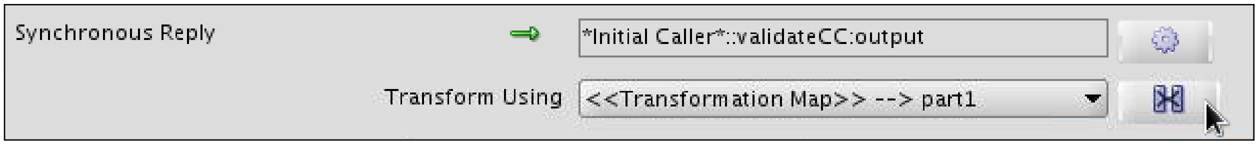
   The XSLT Mapper window opens.

   d. Map the `CCNumber` element in the Source column to the `ccNum` element in the target column.



   e. Save your work.
   f. Close the XSLT editor and return to the `ValidateCreditCard.mplan` window.

10. Configure the Reply transformation.
   a. In the Synchronous Reply section of the routing rule, click the "Select an existing mapper file or create a new one" icon.



Synchronous Reply *Initial Caller*::validateCC:output

Transform Using <<Transformation Map>> --> part1

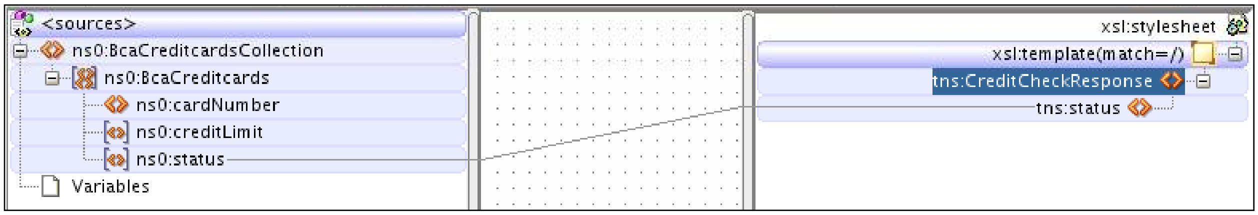
   b. In the Reply Transformation Map dialog box, click the Create Mapping icon.

   The Create Transformation Map dialog box is displayed.

   c. Accept the default file name supplied. Click OK.
   d. Click OK to close the Reply Transformation Map dialog box.

   The XSLT Mapper window opens.

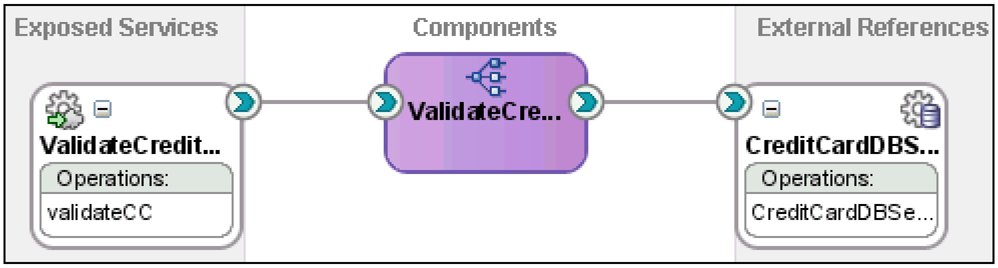
e. Map the status element in the Source column to the status element in the target column.



f. Save your work.

g. In the JDeveloper window, close the XSLT Mapper and the `ValidateCreditCard.mplan` windows.

11. Verify that the composite application assembly model resembles the following image:



Practice 5-4: Deploying and Testing the CCValidate Composite

Overview

In this practice, you deploy and test the CCValidate composite application.

Assumptions

This practice assumes that you have completed all work through Practice 5-3 successfully.

Tasks

1. In the JDeveloper Application Navigator, right-click the CCValidate project and select Deploy > CCValidate.

   The Deploy CCValidate wizard appears.

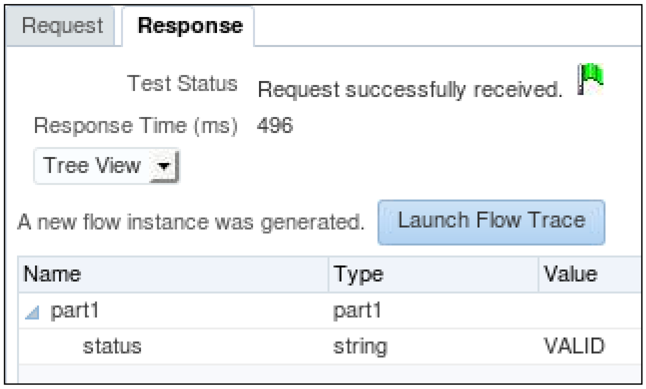
2. Use the instructions in the following table to complete the deployment:

Step	Window	Choices or Values
a.	Deployment Action	Deploy to Application Server. Click Next.
b.	Deploy Configuration	Select the “Overwrite any existing composites with the same revision ID” check box. Click Next.
c.	Select Server	`IntegratedWebLogicServer` Click Next.
d.	SOA Servers	Click Next.
e.	Summary	Click Finish.

3. Deployment processing starts. Monitor deployment progress and check for successful compilation in the SOA – Log window. Verify that deployment is successful in the Deployment – Log window.
4. In a web browser window, access Enterprise Manager at http://localhost:7101/em.
   a. In the Target Navigation pane, expand the SOA folder, right-click soa-infra and select Home > Deployed Composites.
   b. Click the “CCValidate [1.0]” link.
   c. On the “CCValidate [1.0]” home page, click Test.
   d. On the Test Web Service page, scroll down to the Input Arguments section of the Request tab. Expand **part1** and enter the following values:
      - CCNumber: 1234-1234-1234-1234
      - Amount: 1000
   e. Click Test Web Service (on the lower-right corner of the screen).

      The Response tab is displayed.

f. Verify that the status value returned is VALID, as shown in the following screenshot:



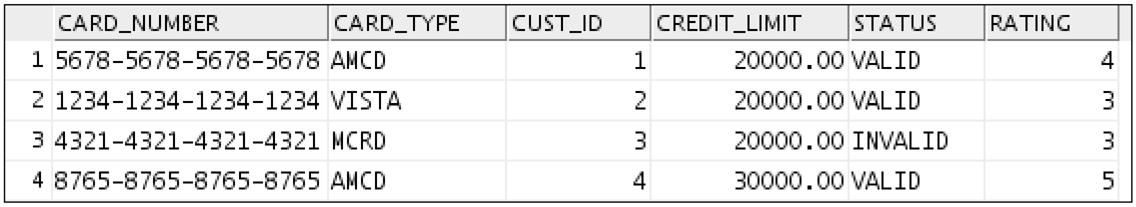
Test Status Request successfully received.

Response Time (ms) 496

A new flow instance was generated.

Name	Type	Value
part1	part1	
status	string	VALID

Note: The credit card number was found in the `BCA_CREDITCARD` table. The status field of the record is VALID, which was returned in the preceding test.

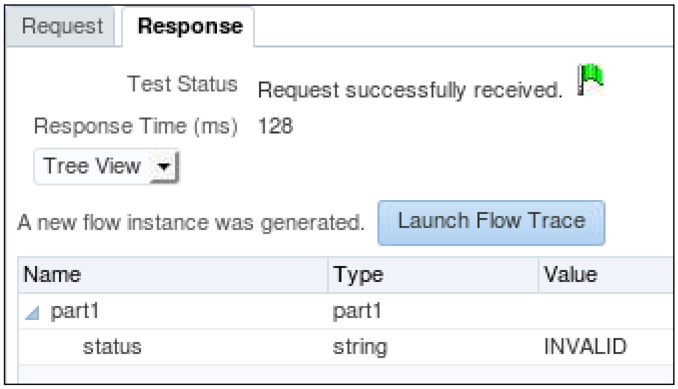


	CARD_NUMBER	CARD_TYPE	CUST_ID	CREDIT_LIMIT	STATUS	RATING
1	5678-5678-5678-5678	AMCD	1	20000.00	VALID	4
2	1234-1234-1234-1234	VISTA	2	20000.00	VALID	3
3	4321-4321-4321-4321	MCRD	3	20000.00	INVALID	3
4	8765-8765-8765-8765	AMCD	4	30000.00	VALID	5

5. To perform a second test, enter another request and click Test Web Service on the Request tab.

   a. On the Request tab page, modify the CCNumber value to be `4321-4321-4321-4321`. Leave the amount at `1000`. Click Test Web Service.

   b. On the Response tab page, verify that the status string value that is returned is INVALID, as shown in the following screenshot:



Test Status Request successfully received.

Response Time (ms) 128

A new flow instance was generated.

Name	Type	Value
part1	part1	
status	string	INVALID

6. In JDeveloper, close the CCValidate overview tab.