Course Exercises Guide

IBM App Connect Professional

Course code ZE550 ERC 1.0

IBM Training

December 2017 edition

Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Contents

Trademarks . . . . . **iv**

Exercises description . . . . . **v**

Exercise 1. Creating a project **1-1**
1.1. IBM App Connect Professional Hello 1-3
1.2. Web Management Console (WMC) 1-9
1.3. Using mapping functions 1-15
1.4. Testing project components 1-28

Exercise 2. Loading accounts from a flat file **2-1**
2.1. Flat File to Salesforce.com 2-3
2.2. Creating a flat file schema by using the Flat File Wizard 2-8
2.3. Creating an orchestration 2-14

Exercise 3. Single database table operation into Salesforce.com **3-1**
3.1. Load database records into Salesforce 3-3

Exercise 4. Loading a Salesforce.com account to ESB by using IBM MQ **4-1**
4.1. Load Salesforce.com Accounts to ESB by using IBM MQ 4-3
4.2. Run the Load Salesforce.com Accounts to ESB by using IBM MQ Lab 4-21

Exercise 5. Invoking web services **5-1**
5.1. Invoke a web service with a fixed symbol 5-3
5.2. Invoke web service with Input 5-14

Exercise 6. Interacting with RESTful Services and Salesforce **6-1**
6.1. Expose an orchestration as a RESTful service that invokes a second REST service and synchronizes data into Salesforce.com 6-3

Exercise 7. Developing with OData **7-1**
7.1. Download the OData TIP 7-3
7.2. Configure the OData TIP 7-5
7.3. Deploy to your runtime environment and test 7-16
7.4. Set up external object in Salesforce 7-19

Exercise 8. Combining multiple sources with IBM App Connect Designer **8-1**
8.1. Prework 8-3
8.2. Build App Connect Flow 8-9
8.3. Test the API 8-20

Exercise 9. Salesforce account API **9-1**
9.1. Prerequisites for loading data into Salesforce 9-3

Appendix A. Creating a Salesforce.com developer account **A-1**

Appendix B. Updating the Salesforce account object **B-1**

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

Bluemix®	DB™	DB2®
IBM Bluemix™	Notes®	WebSphere®

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.

Exercises description

This course includes the following exercises:

- Creating a project
- Load accounts from flat file
- Single database table operation into Salesforce.com
- Load Salesforce.com account to ESB by using IBM MQ
- Invoke web services
- Interacting with RESTful services and Salesforce
- OData
- Combining multiple sources using IBM App Connect Designer
- Salesforce account API

In the exercise instructions, you can check off the line before each step as you complete it to track your progress.

Most exercises include required sections, which should always be completed. It might be necessary to complete these sections before you can start later exercises. If you have sufficient time and want an extra challenge, some exercises might also include optional sections that you can complete.

Important

The exercises in this course use a set of lab files that might include scripts, applications, files, solution files, PI files, and others. The course lab files can be found in the following directory:

`C:\student\Lab Resources` for Windows

The exercises point you to the lab files as you need them.

Connection parameters for labs

Table 1.

Type	User ID	Password
Windows administrator	`Administrator`	`passw0rd`
Web Manager user ID (WMC)	`admin`	`!n0r1t5@C`
Salesforce user ID	Create at salesforce.com	Your password+security token

Table 2. IP address

IBM App Connect Studio on Windows	`192.168.246.140`
Appliance EMgmt (WMC)	`192.168.246.133`
Appliance EData (URL)	`192.168.246.134`

Table 3. Database parameters

Database Name	`SAMPLE`
Database Server	`192.168.246.140`
Database Table Name	`ACCOUNT`
Database User	`db2admin`
Database Password	`db2admin`
WMQ Queue Manager	`CASTIRON`
WMQ Port	`1414`
WMQ Server Channel	`SVR.CHL`
WMQ User	`db2admin`

All students have their own DB2 and MQ instances in their Windows environment.

Exercise 1. Creating a project

Estimated time

00:30

Overview

In this lab, you create a project, HTTPBasicsXX, with an orchestration that receives an HTTP request and sends back a hardcoded response.

Objectives

After completing this exercise, you should be able to:

- Create a project
- Work with basic HTTP activities
- Configure basic activity mapping
- Validate the project
- Publish and deploy the project

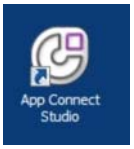
Introduction

When the orchestration is complete, you publish and deploy the project. The orchestration is then tested by using both HTTP GET and POST methods. With the orchestration, logging level set to “All”, you are able to examine both the headers and body of the HTTP request.

Exercise instructions

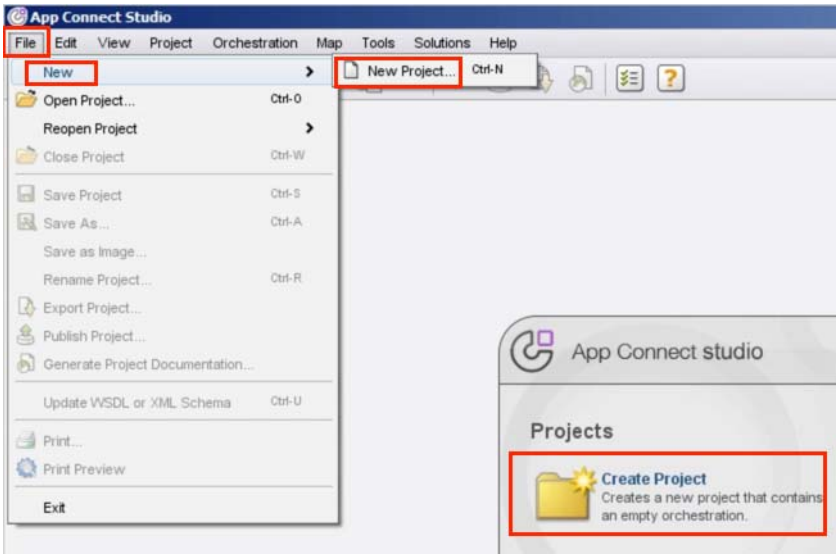
1.1. IBM App Connect Professional Hello

__ 1. Start the IBM App Connect Studio. The icon for the IBM App Connect Studio is on the desktop.



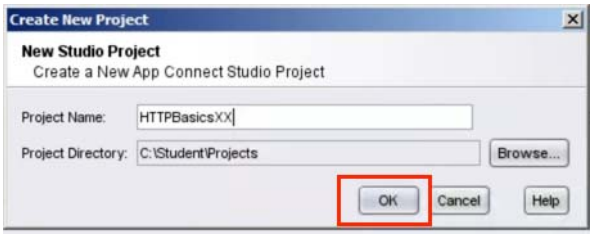
__ 2. Create a project.

__ 3. Click **File > New Project** or click **Create Project** on the splash screen.



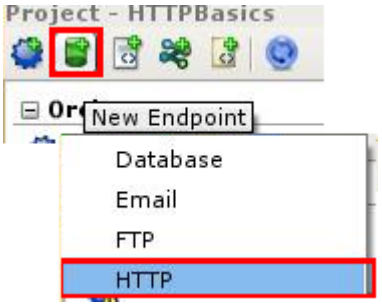
__ 4. Enter: `HTTPBasicsXX`

__ 5. Click **OK**.

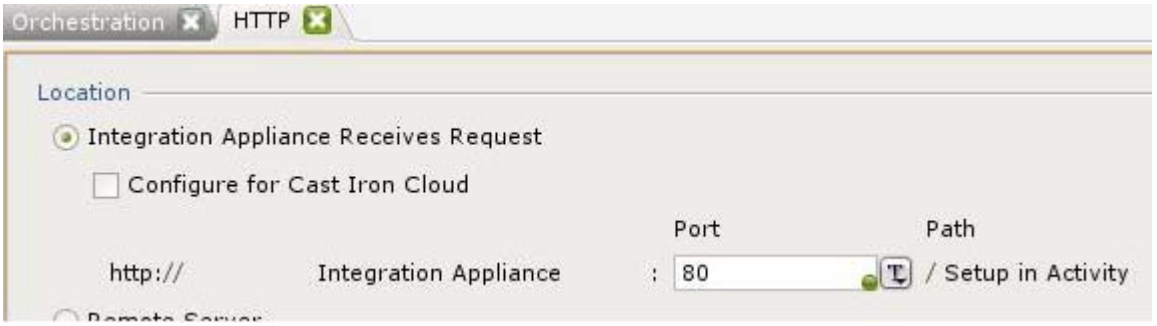


__ 6. Create an HTTP endpoint.

__ a. Use the New Endpoint icon on the **Project** tab.



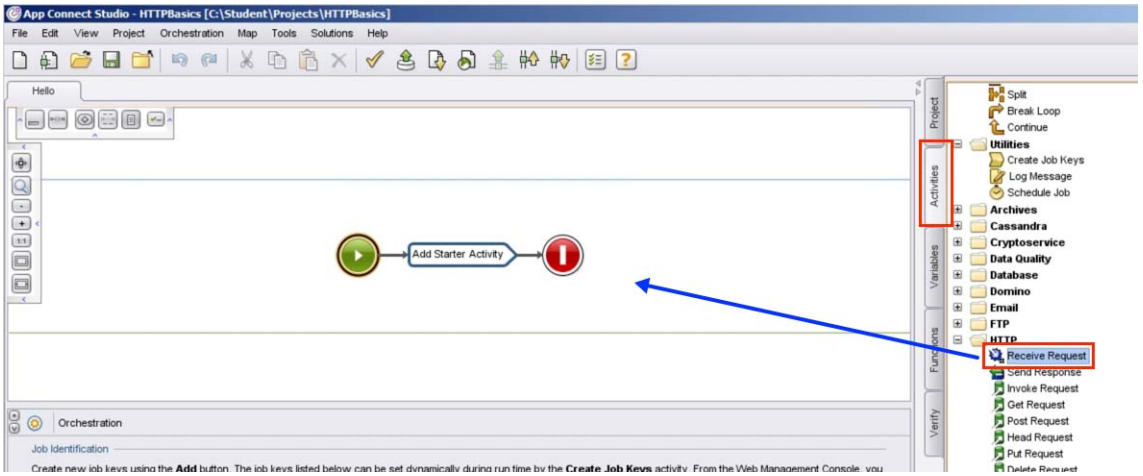
__ 7. For this HTTP Receive Request endpoint, the defaults suffice.



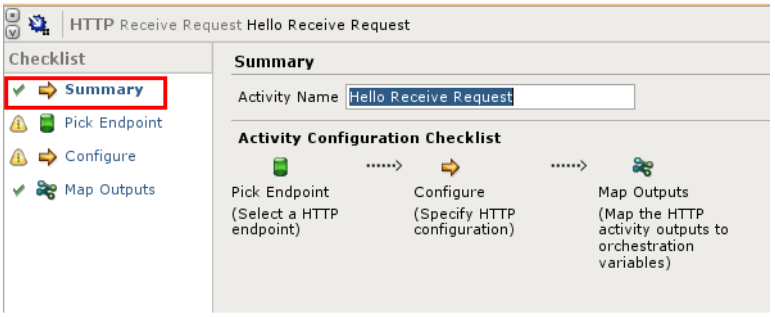
__ 8. Close the **HTTP** tab, and then rename the endpoint to `HTTPReceive` by clicking the name in the toolbox.

__ 9. Rename the default orchestration to `Hello` by right-clicking the existing name (as an alternative method).

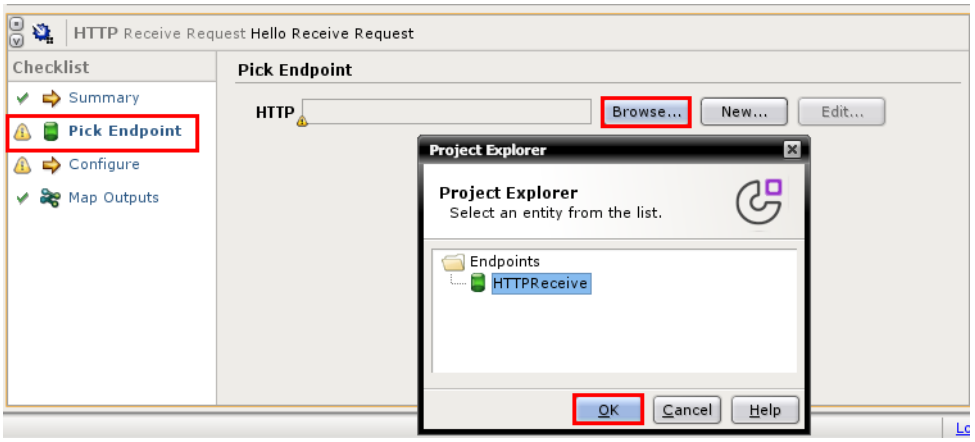
__ 10. Start the orchestration by opening the **Activities** tab in the toolbox and dragging an HTTP Receive Request activity onto the workspace. For new activities in sequence, you do not need to be precise about where you drop the object. Studio places it in the orchestration correctly.



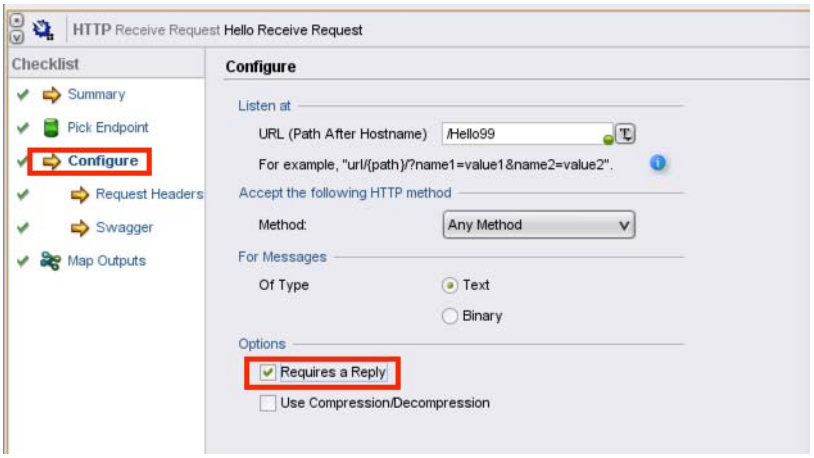
__ 11. Keep in mind that the first activity in an orchestration must be a “Starter” activity, such as this one. When it is added to the flow, the Properties pane displays the summary information for this activity. Start by renaming this activity to: `Hello Receive Request`



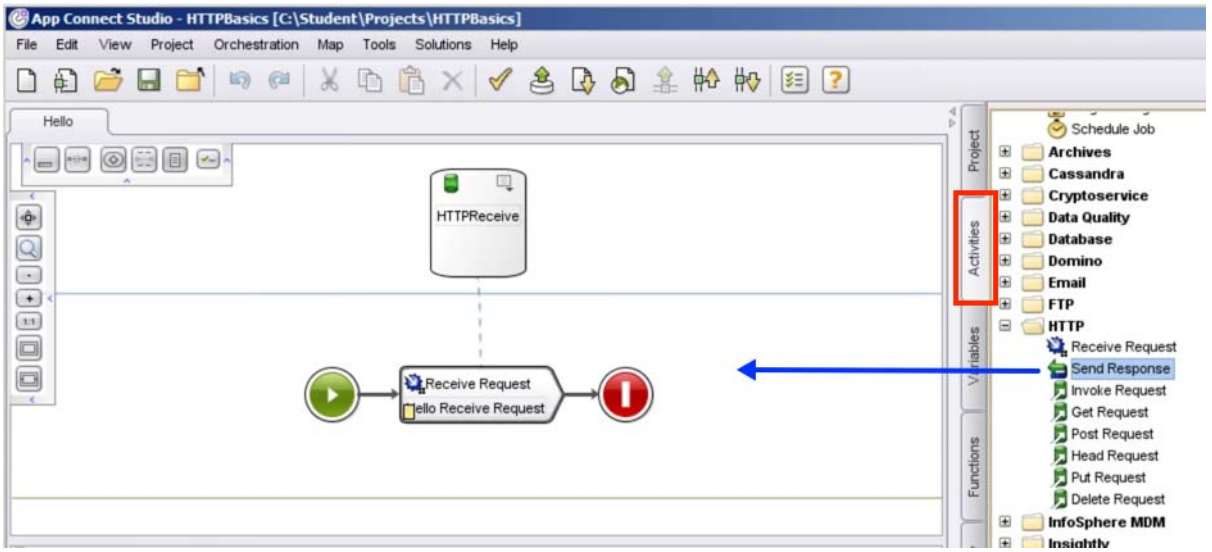
__ 12. The checklist graphically shows the configuration steps that need to be completed. Select the **Pick Endpoint** step and browse for the HTTPReceive endpoint that you defined earlier.



__ 13. Select the **Configure** step and enter `Hello` for the listening URL, and also check the option **Requires a Reply**.

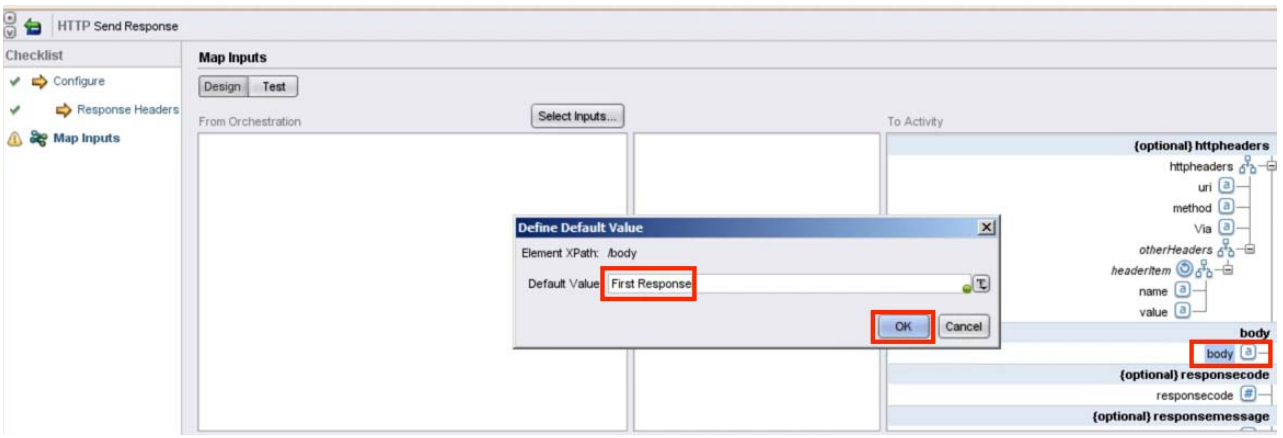


__ 14. Add an HTTP Send Response activity to the orchestration, by dragging an HTTP Send Response activity onto the workspace.



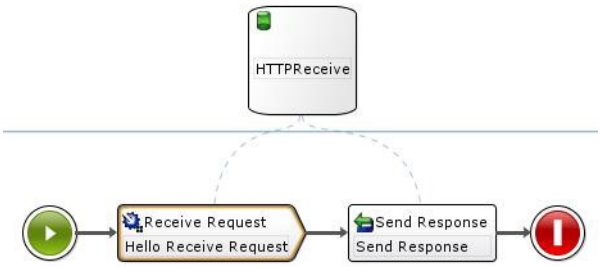
__ 15. The Configure step should already show that this activity is replying to Hello Receive Request.

__ 16. Configure the Map Inputs step to return a fixed value. Right-click the **body** object to open its menu, select the **Define Default Value** option, and type: `First Response`

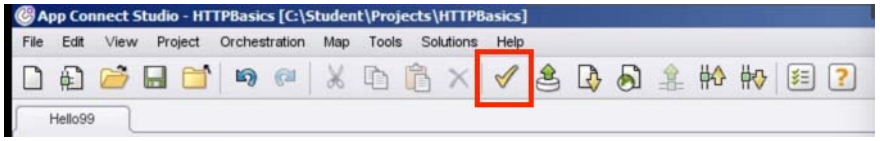


Notice that the body object has a new icon that shows it has a default value.

__ 17. The completed orchestration looks as follows:



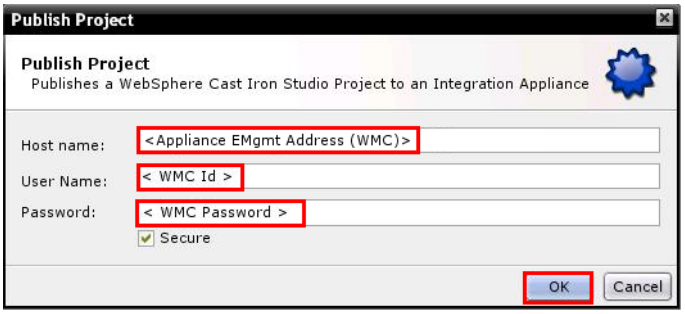
__ 18. Validate the project by selecting **Orchestration > Validate**, or by using the toolbar icon.



__ 19. **Save** your project from the toolbar by using the toolbar icon.

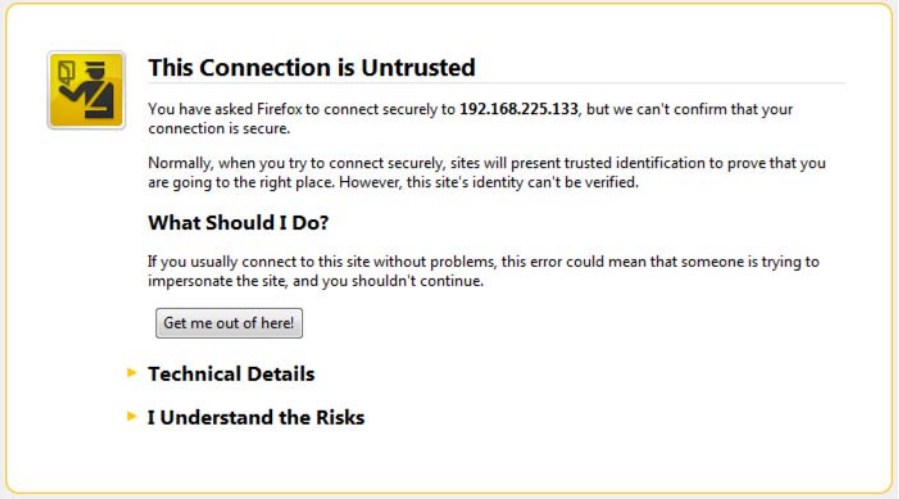
__ 20. If prompted, save the project. Then, publish the project, by selecting menu item **File > Publish Project**, or by using the toolbar icon.

__ 21. Enter the connection details the first time and click **OK**. A successful publish returns a confirmation message (see the Connection Parameters spreadsheet).

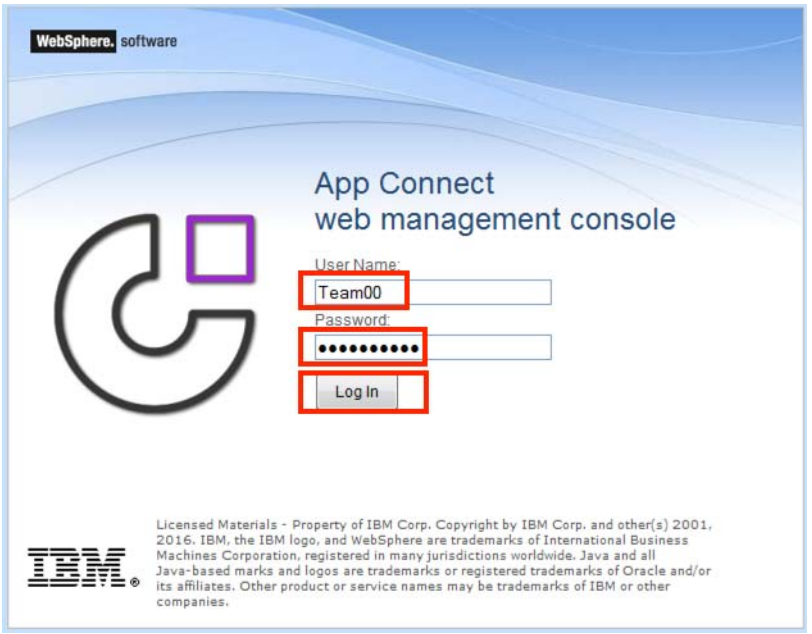


1.2. Web Management Console (WMC)

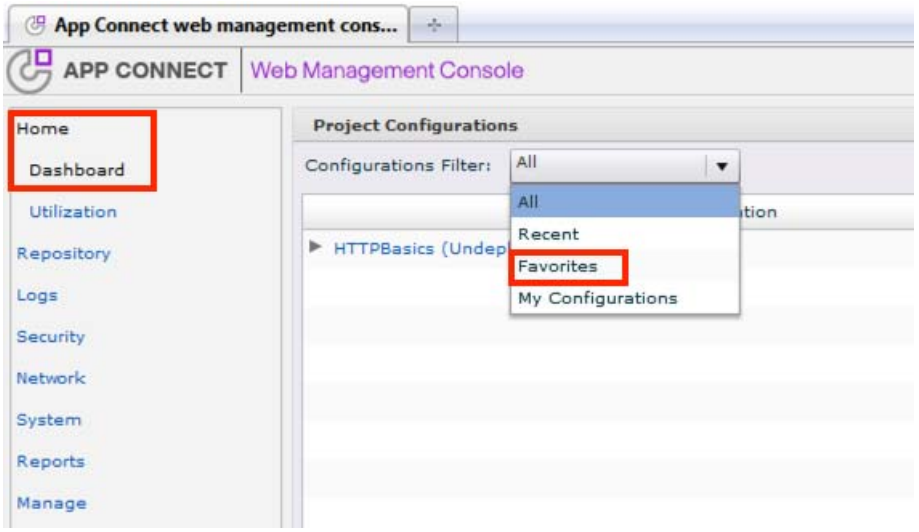
__ 1. From the Firefox browser, launch the Web Management Console by using the URL **Appliance EMgmt Address (WMC)**, which is provided on the Connection Parameters spreadsheet.
If you are prompted with a security notice, click **I Understand the Risks** and **Add Exception**. Then, click **Confirm Security Exception**.



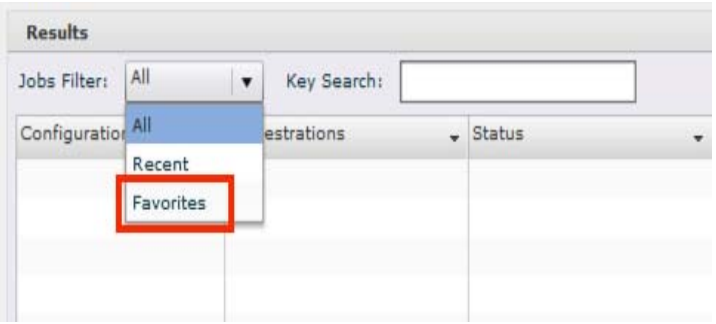
__ 2. Log in using the <WMC Id> and <WMC Password> from the Connection Parameters spreadsheet (it is OK to use the admin ID and password).



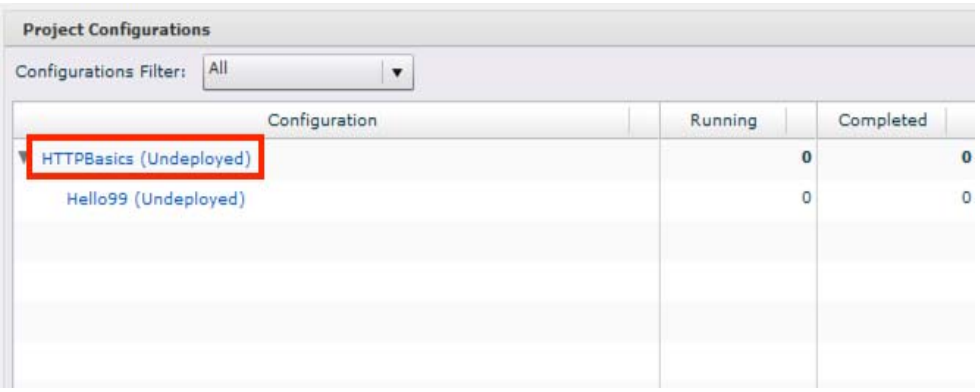
__ 3. As soon as you are in the Web Management Console, from the Home dashboard, click **ALL** in Project Configurations and select **Favorites** from the menu to limit the view to only your projects.



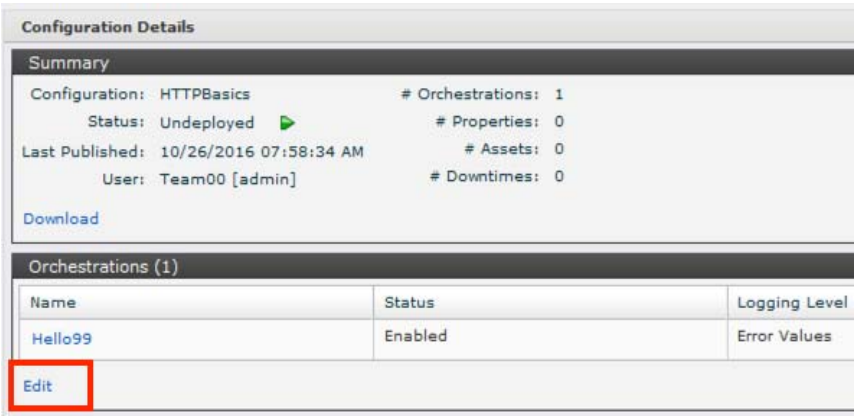
__ 4. Repeat this step in Results to limit the view to only your projects.



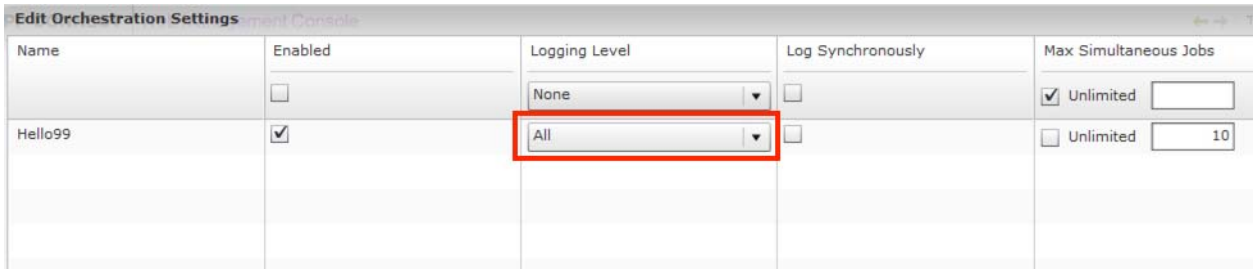
__ 5. From the **Home > Dashboard** page, click the project/configuration name link.



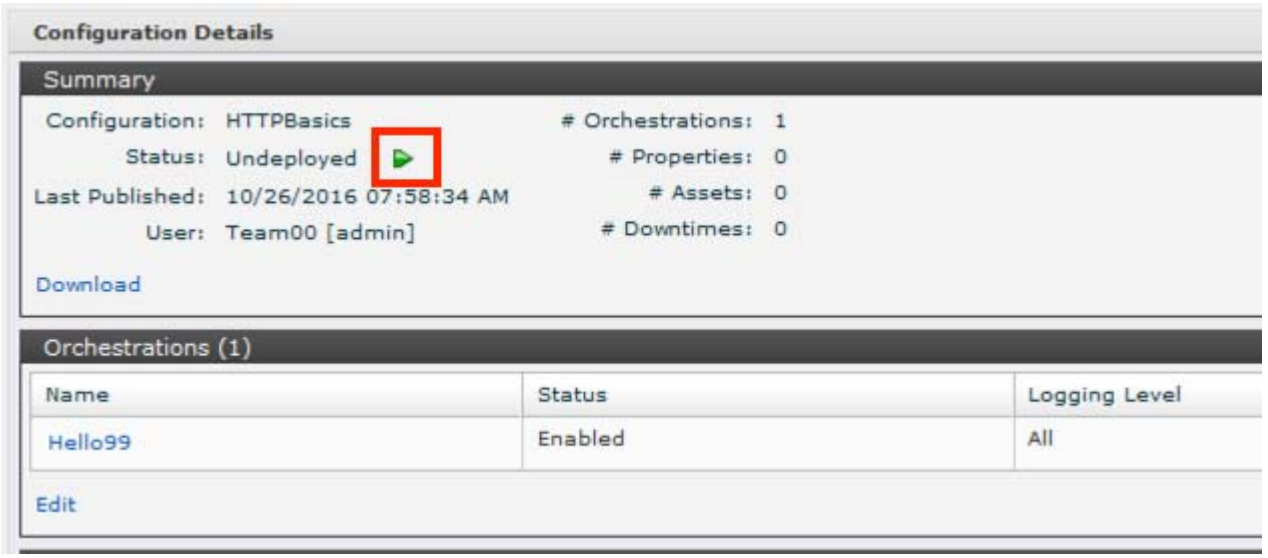
__ 6. Clicking the configuration link opens the edit page with the list of orchestrations displayed. Click **Edit** in the Orchestrations section.



__ 7. Apply changes to all the orchestrations in the project with the unnamed top row, or select the orchestrations individually. Change the logging level to **All** and click the Save link.



__ 8. Back on the Configurations page, deploy and run the project by using the Run Configuration icon:

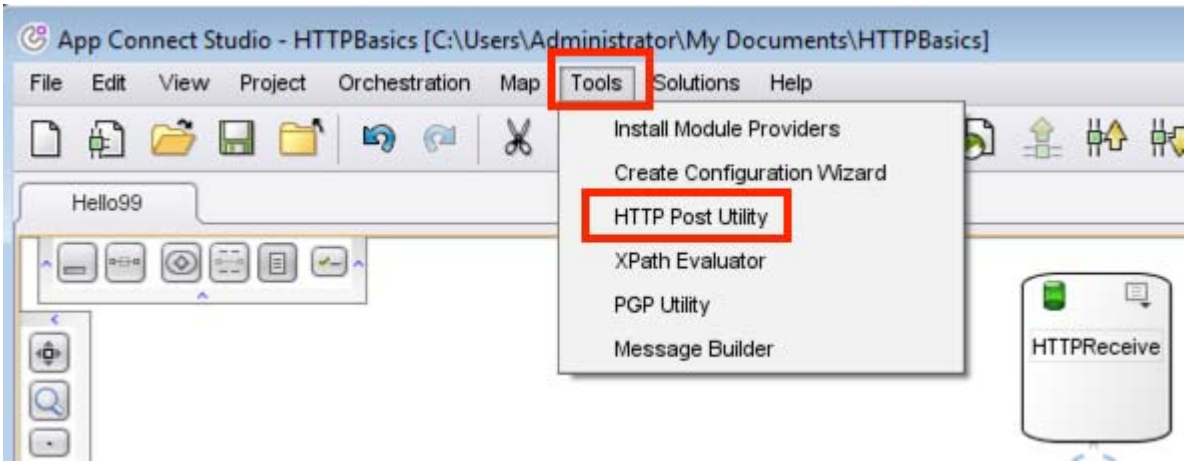


__ 9. After a few moments, the icons change to show that the status of the configuration is **Running**. If the WMC does not refresh right away, press the F5 key to force the refresh.

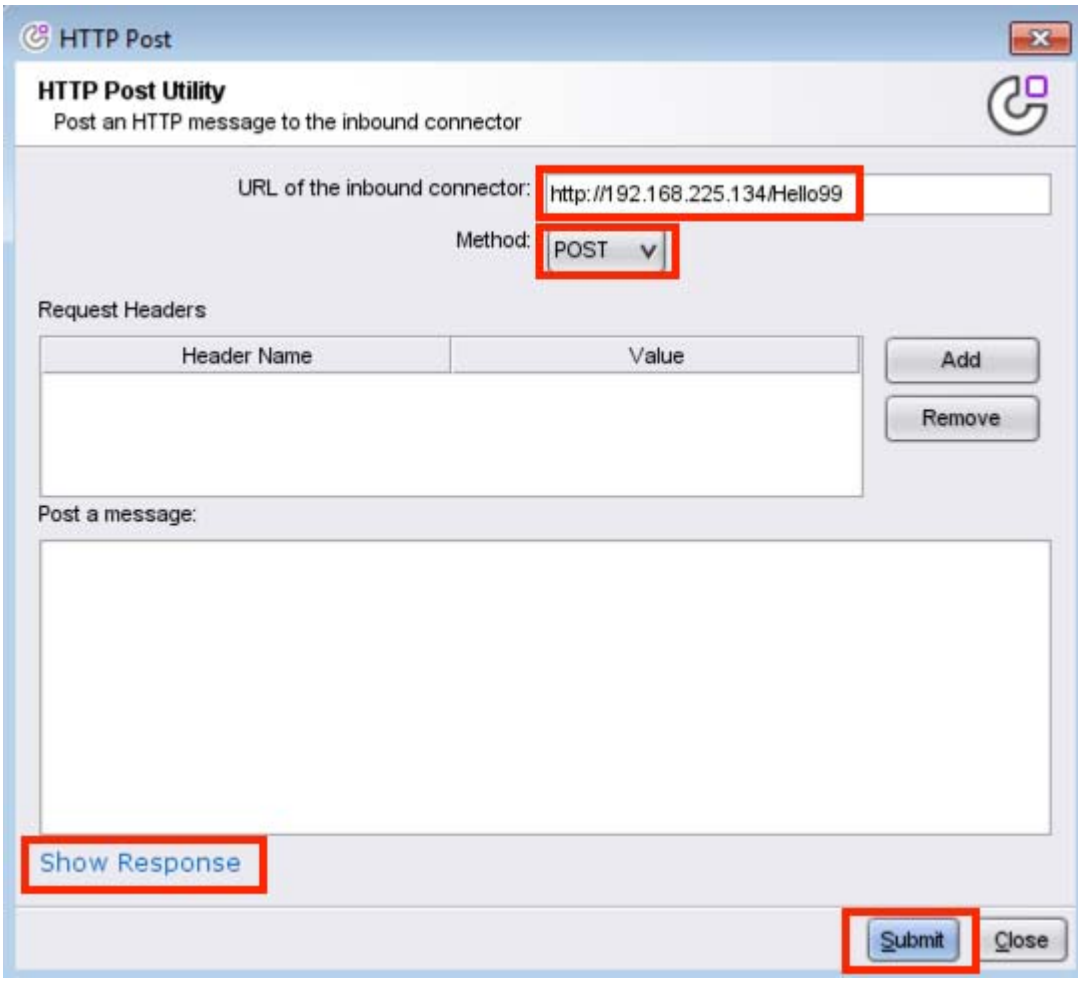
__ 10. In the App Connect Studio, trigger the orchestration twice, once for POST and once for GET. To issue a POST, use the HTTP Post Utility that is included with the Studio installation. To issue a GET, enter the URL into your browser. In either case, use the IP address of the appliance's Edata interface. See the Connectivity Parameters in the Exercises Description section of this lab guide.

POST: Launch the HTTP Post Utility with the appropriate URL. Then, browse for the input file in the `C:\Student\Lab Resources\Lab 1` folder. After submission, you should see the reply text that was entered into the body object of the HTTP Send Response activity.

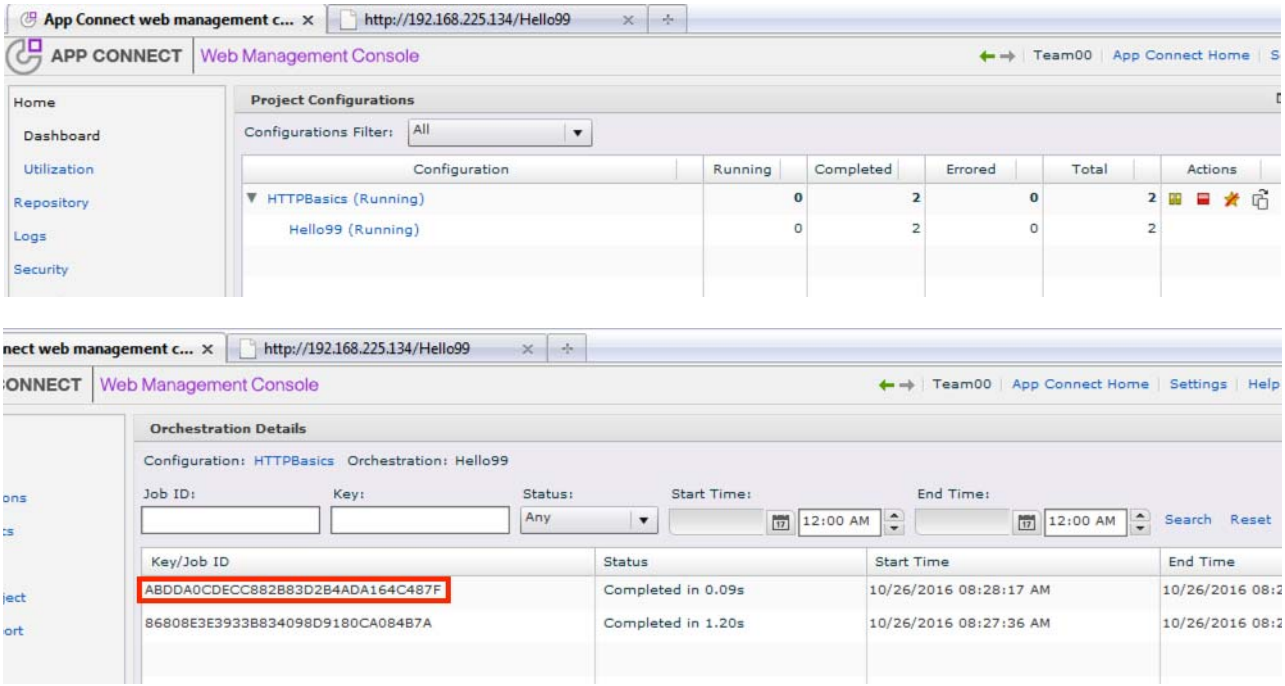
__ a. Select **HTTP Post Utility** from the toolbar.



__ b. Click **Show Response** to open the window that shows the output results. Click **Submit** to execute the orchestration.



__ c. The WMC shows the results of each run. From the dashboard, expand the configuration and click the orchestration that is being tested to display the Orchestration Details page, which shows the results of each run of the orchestration.

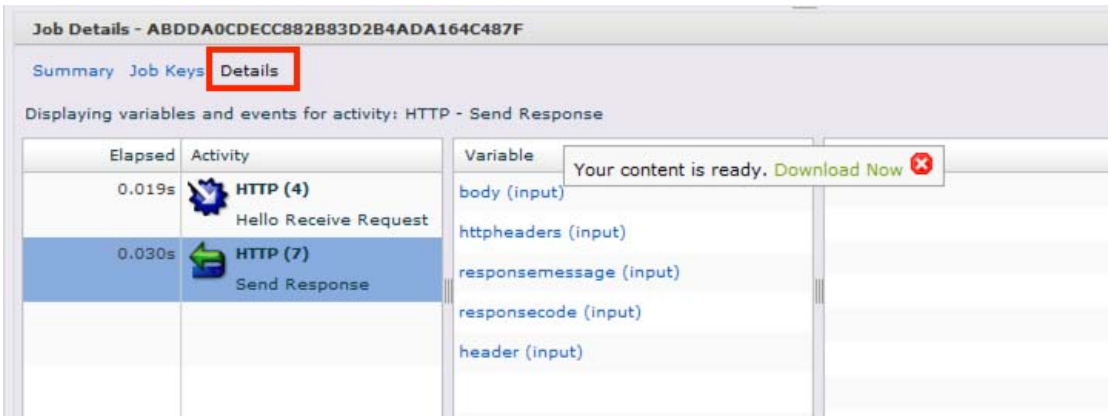


__ d. Trigger the orchestration again for the GET as previously instructed.

__ 11. Select the Job ID link for the POST run in the top pane. Click the **Details** link in the bottom pane and then the HTTP Send Response section. Click the **body (input)** link to dynamically generate a file with the variable's contents.

This action opens a file that contains the response that was displayed in the browser.

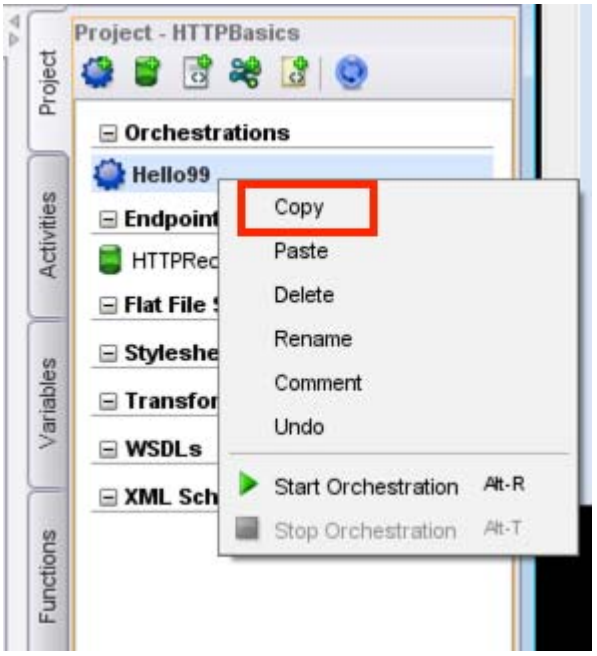
Subsequent labs present better opportunities for viewing input and output values.



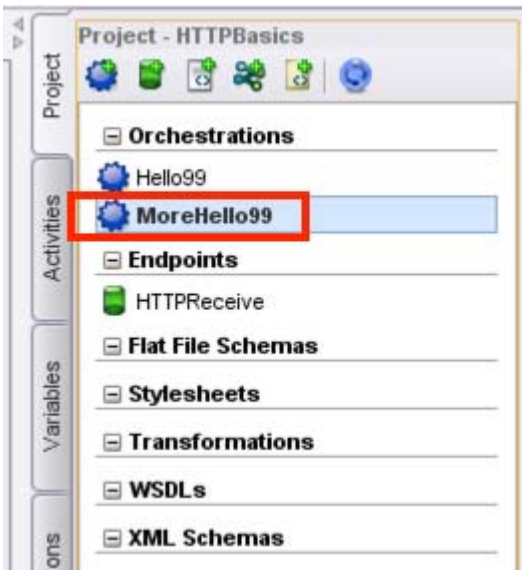
1.3. Using mapping functions

Enhanced App Connect Professional Hello

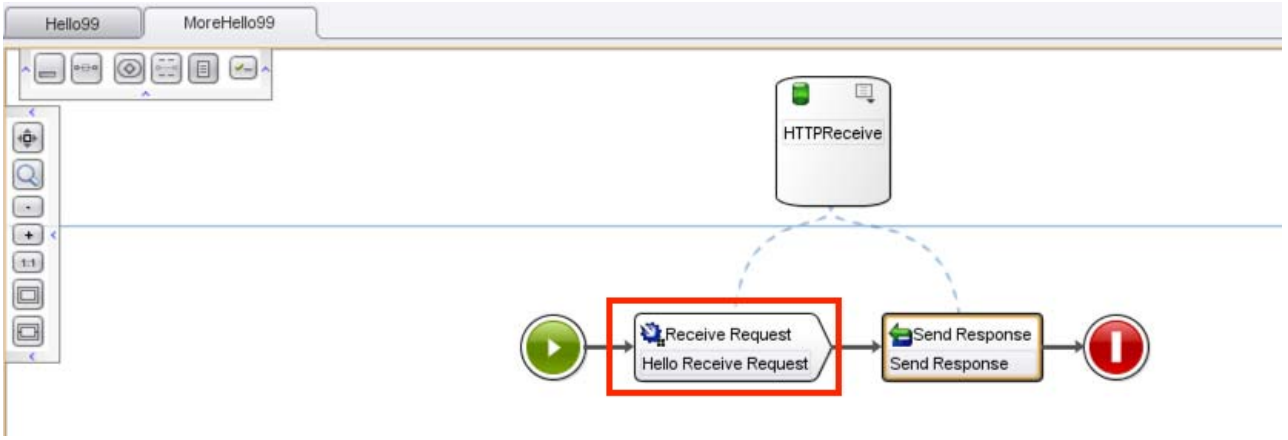
__ 1. In the App Connect Studio, right-click the orchestration to copy and paste a new orchestration from your existing orchestration.



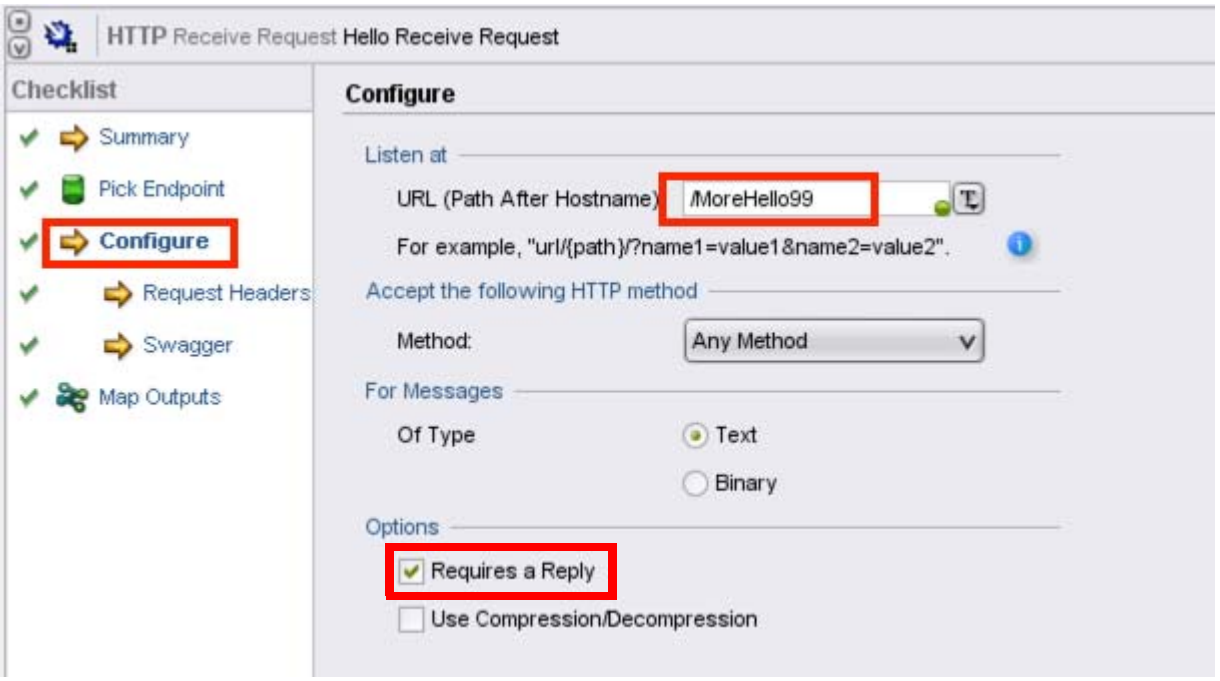
__ 2. Rename the copied orchestration to: `MoreHelloXX`



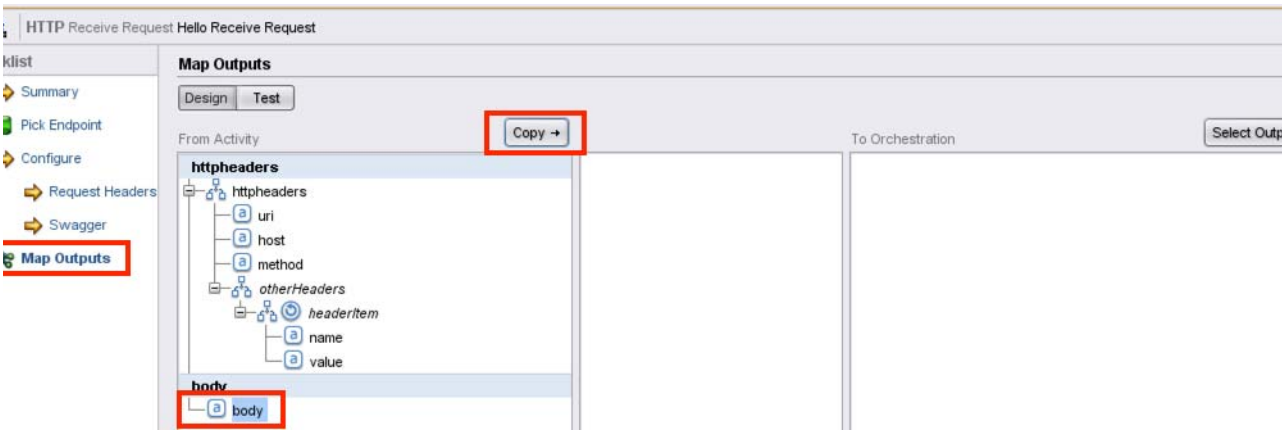
__ 3. Highlight the **Receive Request** activity on the workspace.

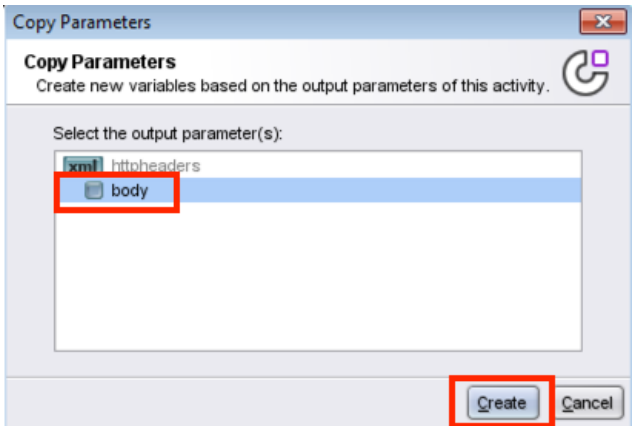


__ 4. Select **Configure** from the Checklist and change the URL to: `MoreHelloXX`
Ensure that **Requires a Reply** is selected with the check mark as indicated in the following screen capture.

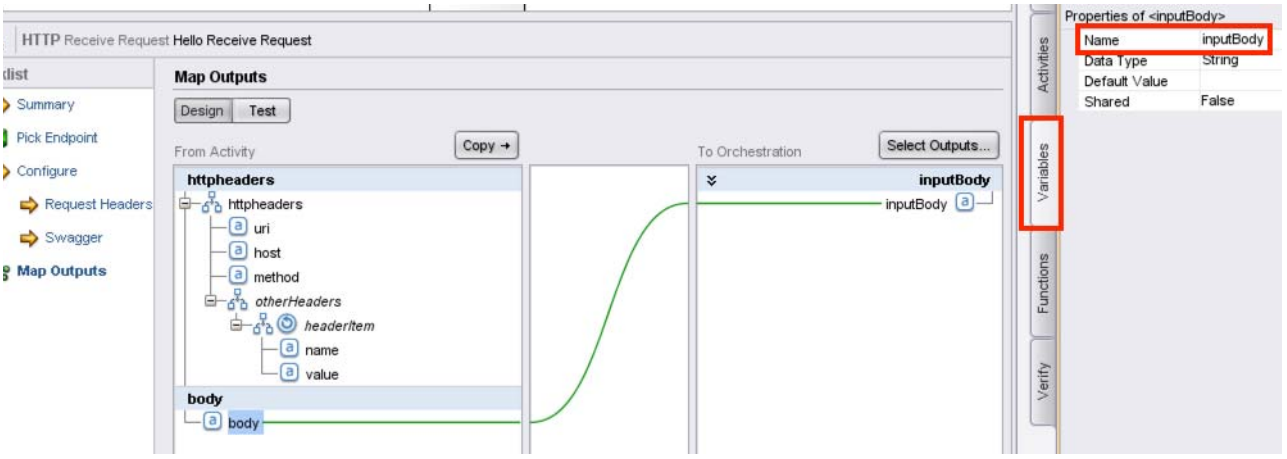


__ 5. Map Outputs: Copy the input body to orchestration, click the body, and select **Copy**.

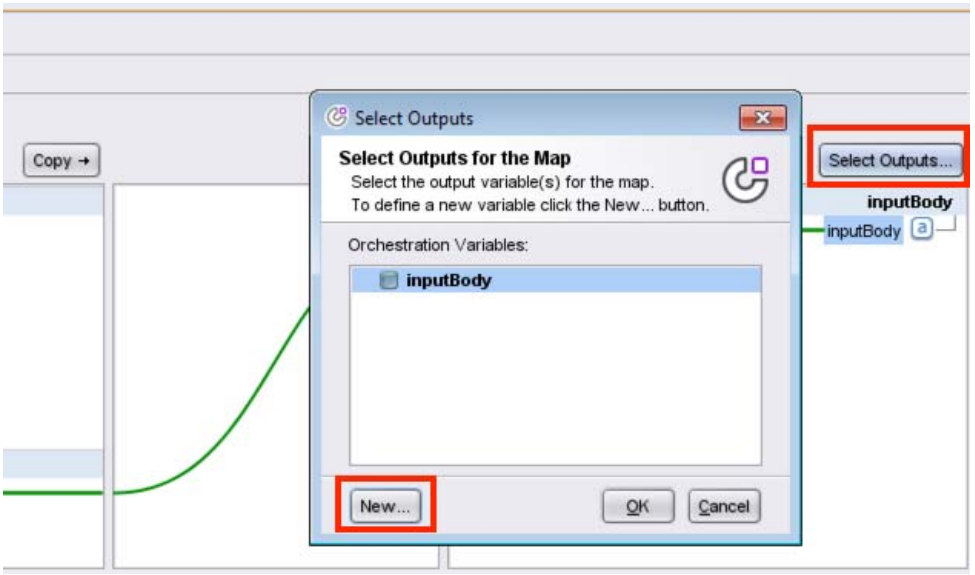


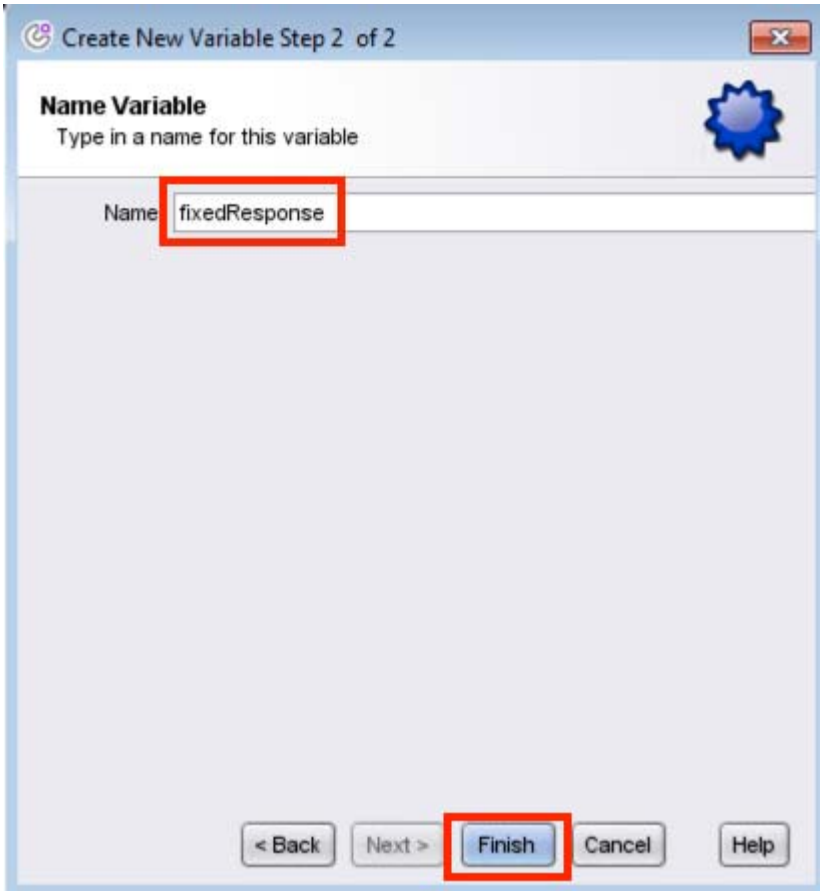


__ 6. Select the **Variables** tab and rename the orchestration variable to: `inputBody`

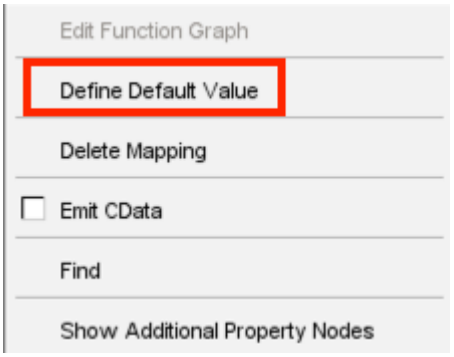


__ 7. Manually create a string variable with a default value of your choosing by clicking the **Select Outputs** and **New** buttons. Name the variable `fixedResponse` and click Finish.

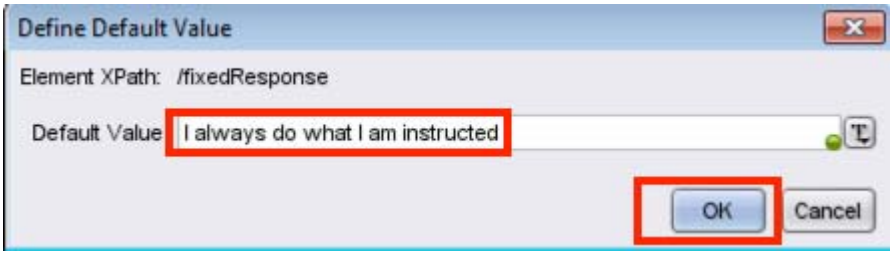


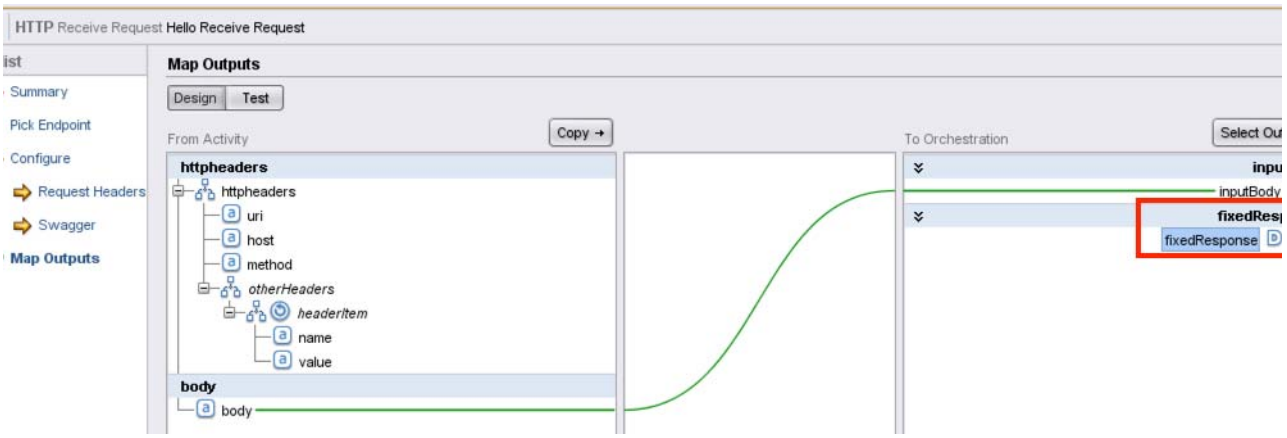


__ 8. Right-click **fixedResponse** and select **Define Default Value**.

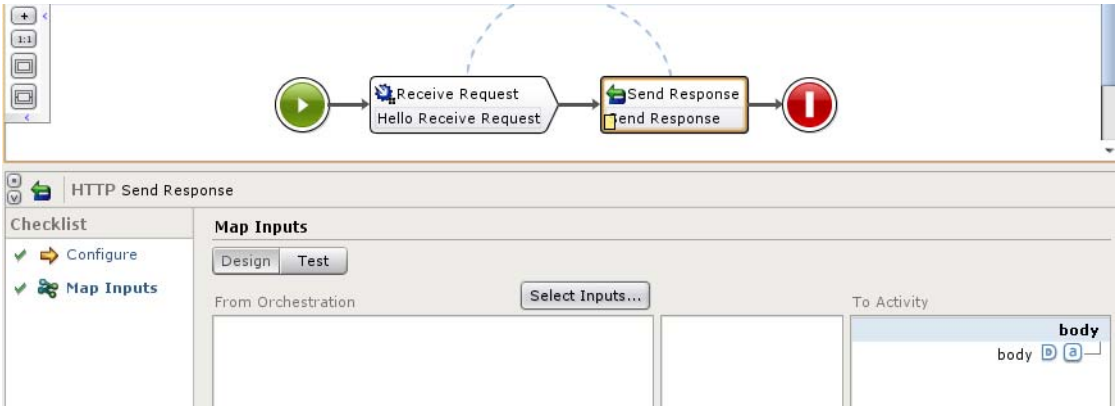


__ 9. Add the default text, for example: **fixedResponse** = `I always do what I am instructed`

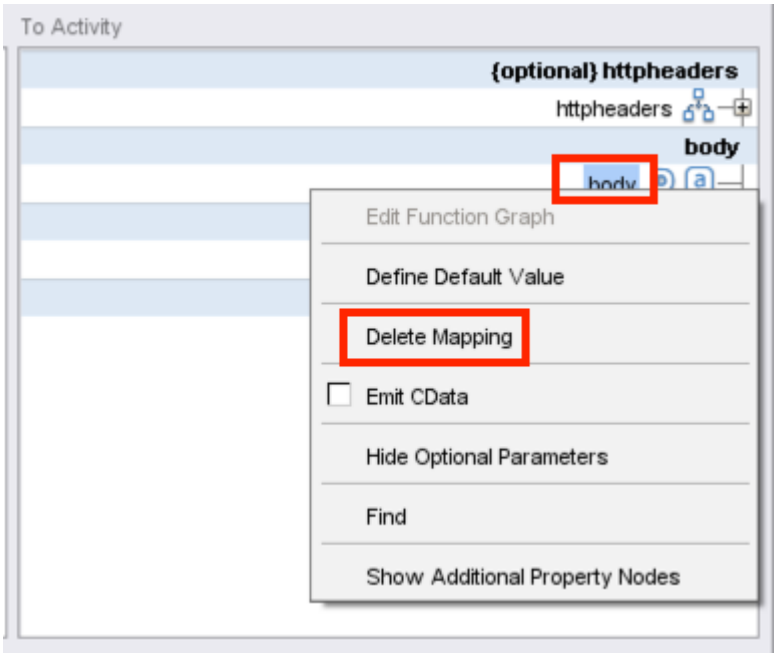




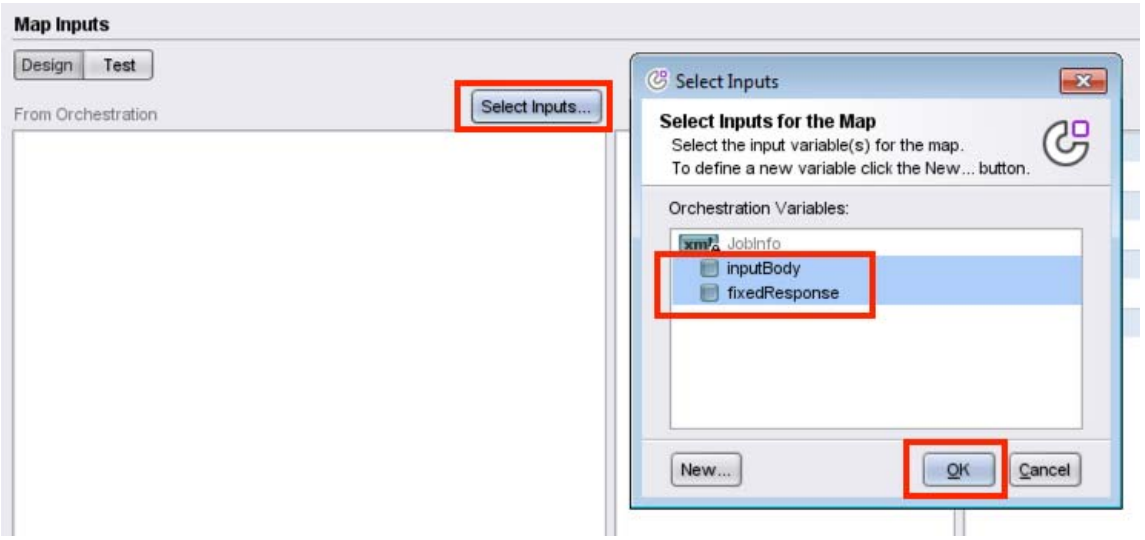
__ 10. Highlight the **Send Response** activity in the workspace.



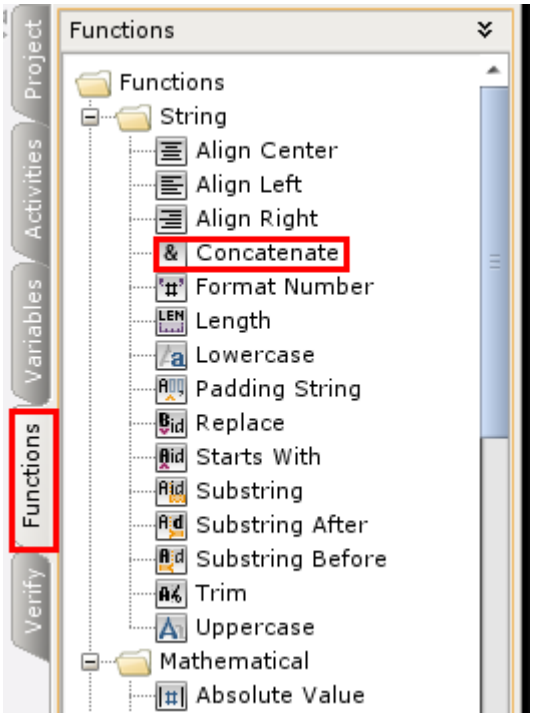
__ 11. Select the Map Inputs, right-click the body, and delete the default mapping for the body.



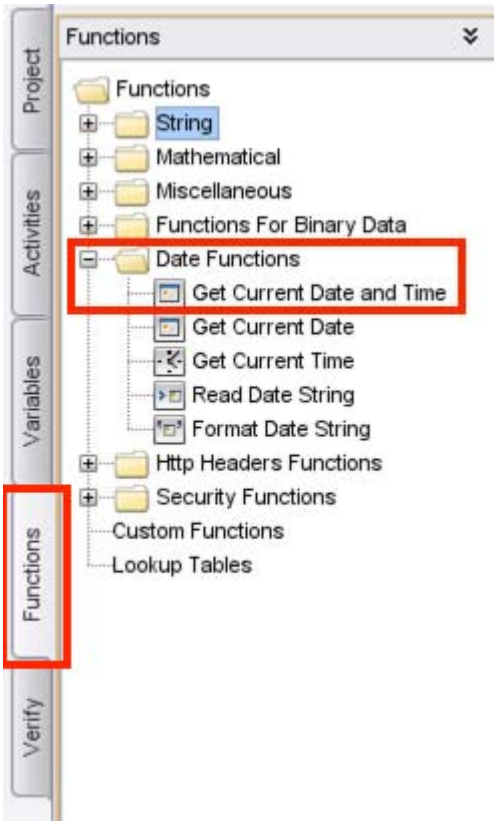
__ 12. Click **Select Inputs**. Select both variables `inputBody` and `fixedResponse` from steps 6 and 7.



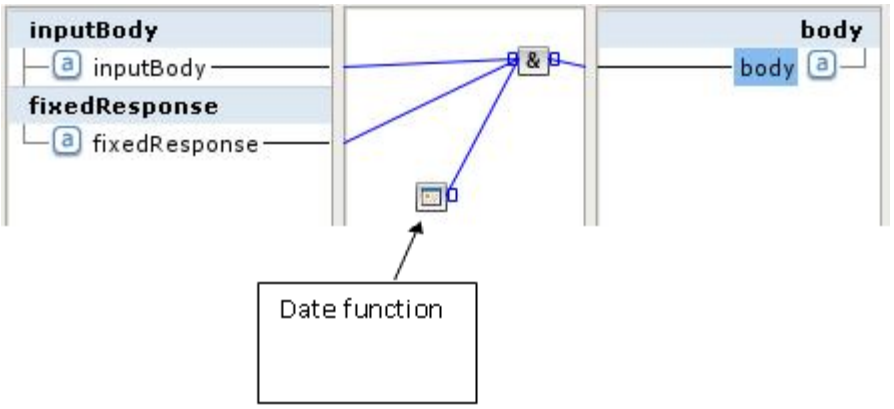
__ 13. Select the **Functions** tab and drag the **Concatenate** function to the function graph (center pane).



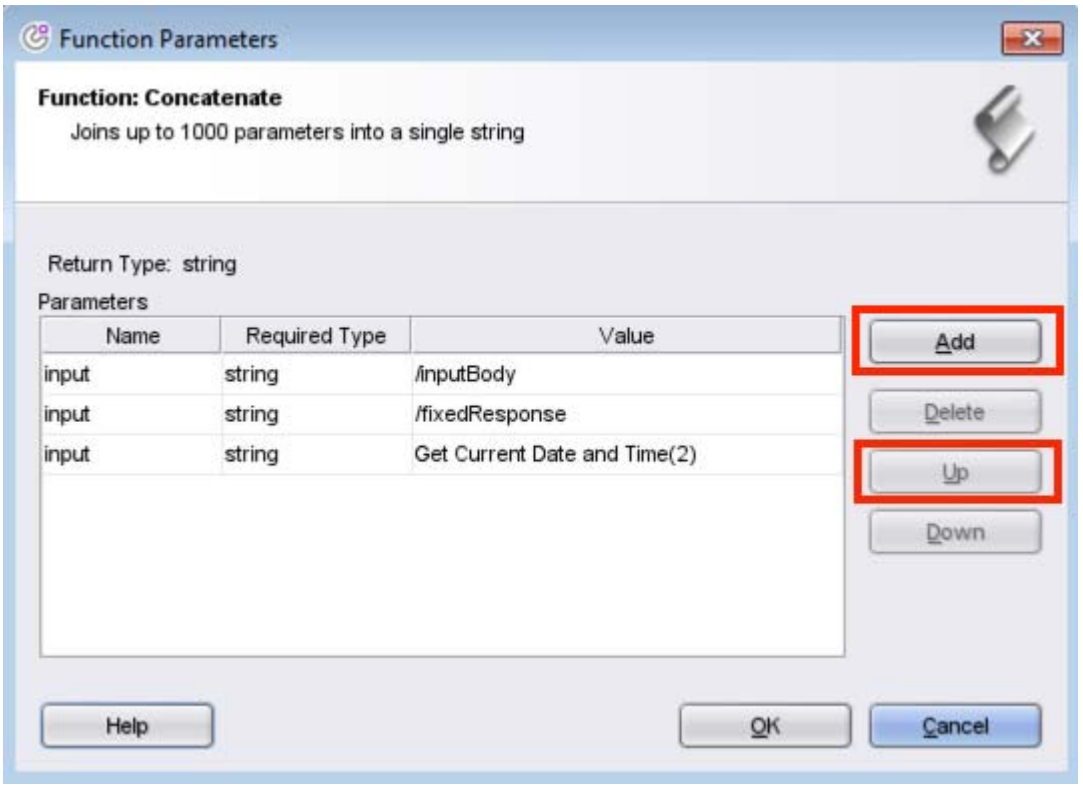
__ 14. Drag the **Get Current Date and Time** function to the function graph pane.



__ 15. Connect the input variables and the Date function to the Concatenate function.

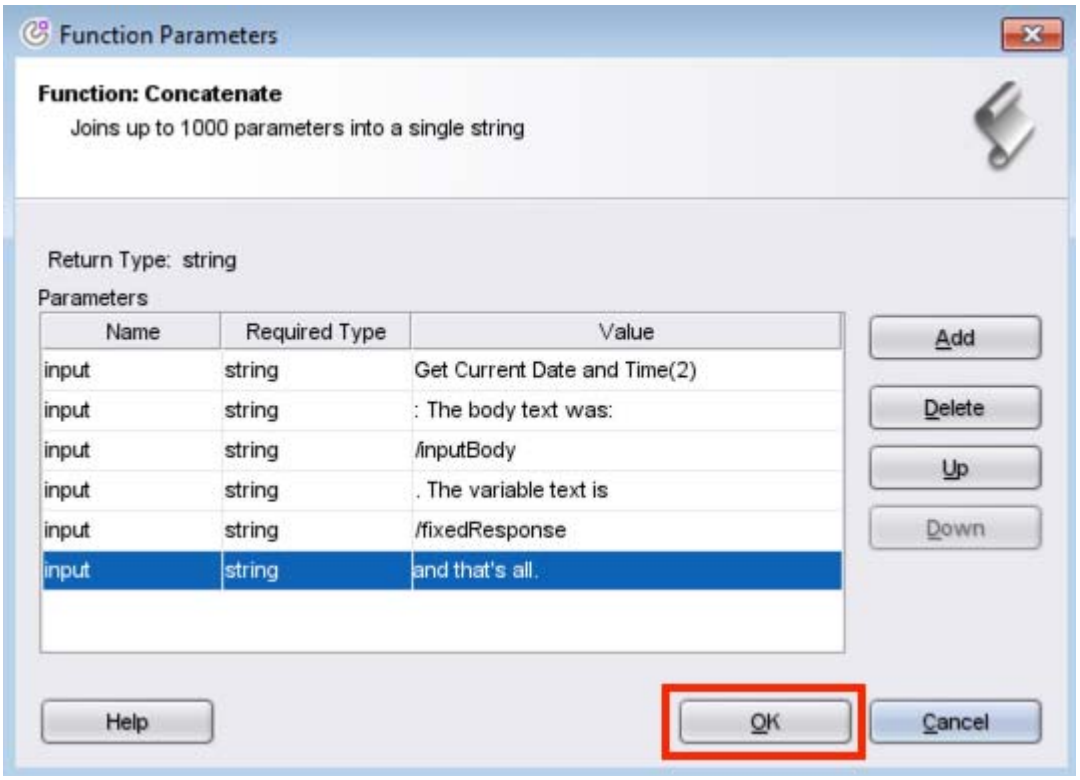


__ 16. Double-click the Concatenate function to open its parameters, and then add strings with the **Add** button.



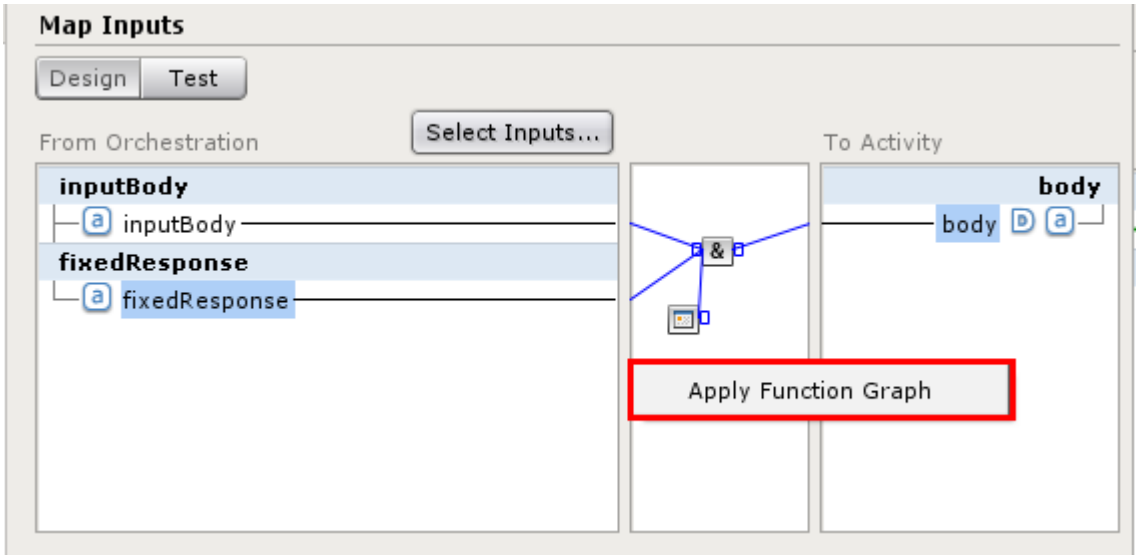
__ 17. Add the following text strings and reorder the parameters to form a coherent output string:

```
: The Body text was:
. The variable text is
  and that is all.
```

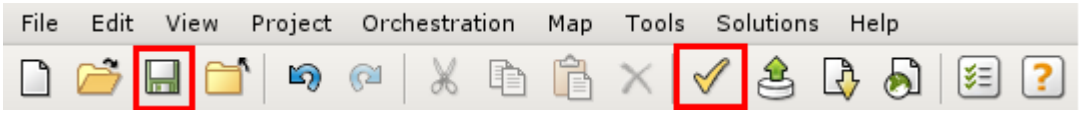


__ 18. Map the Concatenate function to the output body.

Right-click in the function graph pane to **Apply Function Graph**.

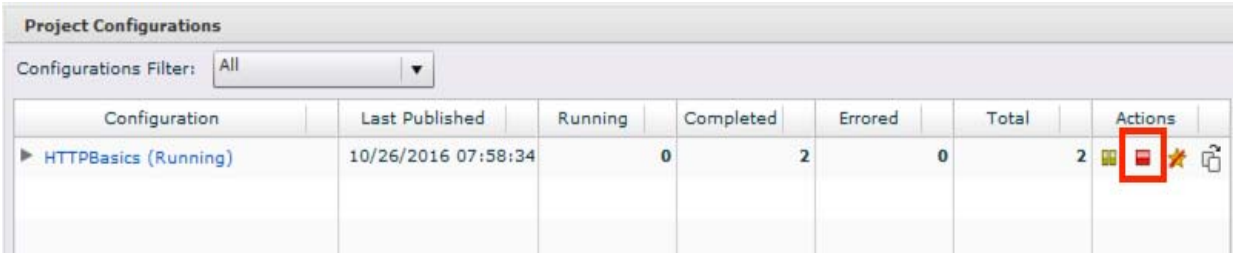


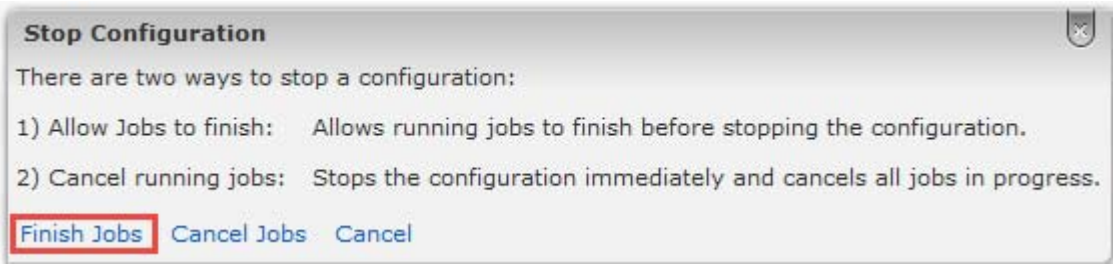
__ 19. Save and Verify the project.



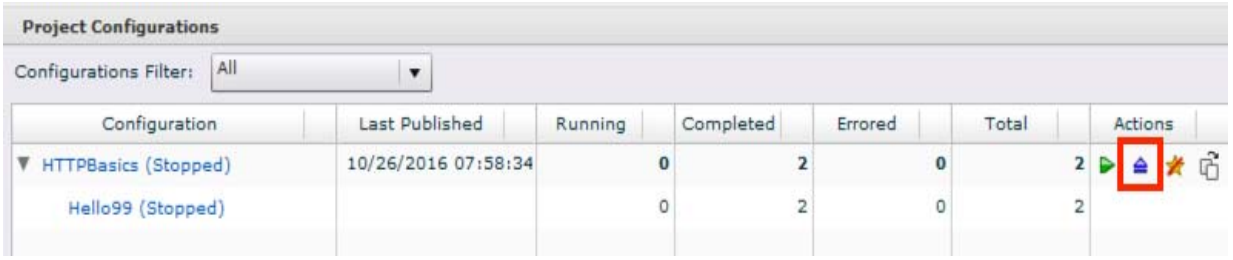
__ 20. Before you can deploy and test the updated project, you must go out to the WMC to Stop and Undeploy the current project.

__ a. Stop Configuration.

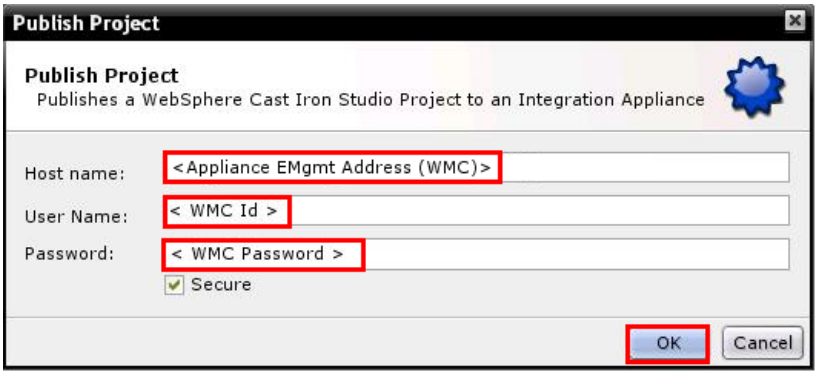




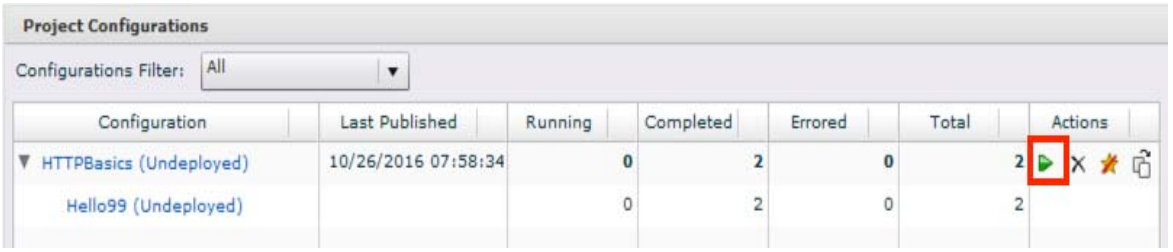
__ b. Undeploy Configuration.



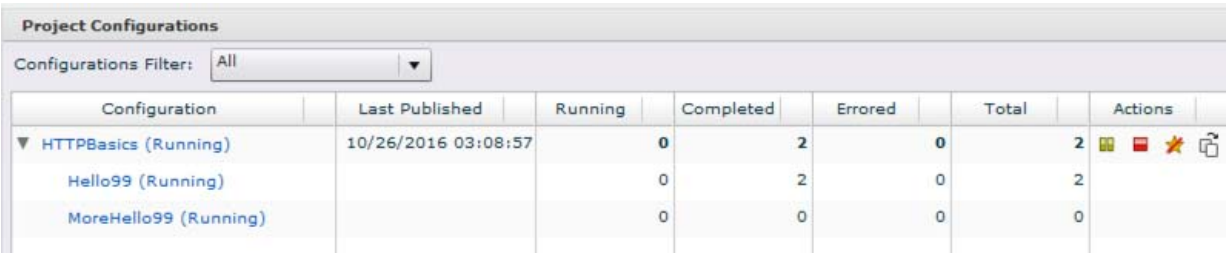
__ 21. Deploy and run the updated project by using the Run Configuration icon:



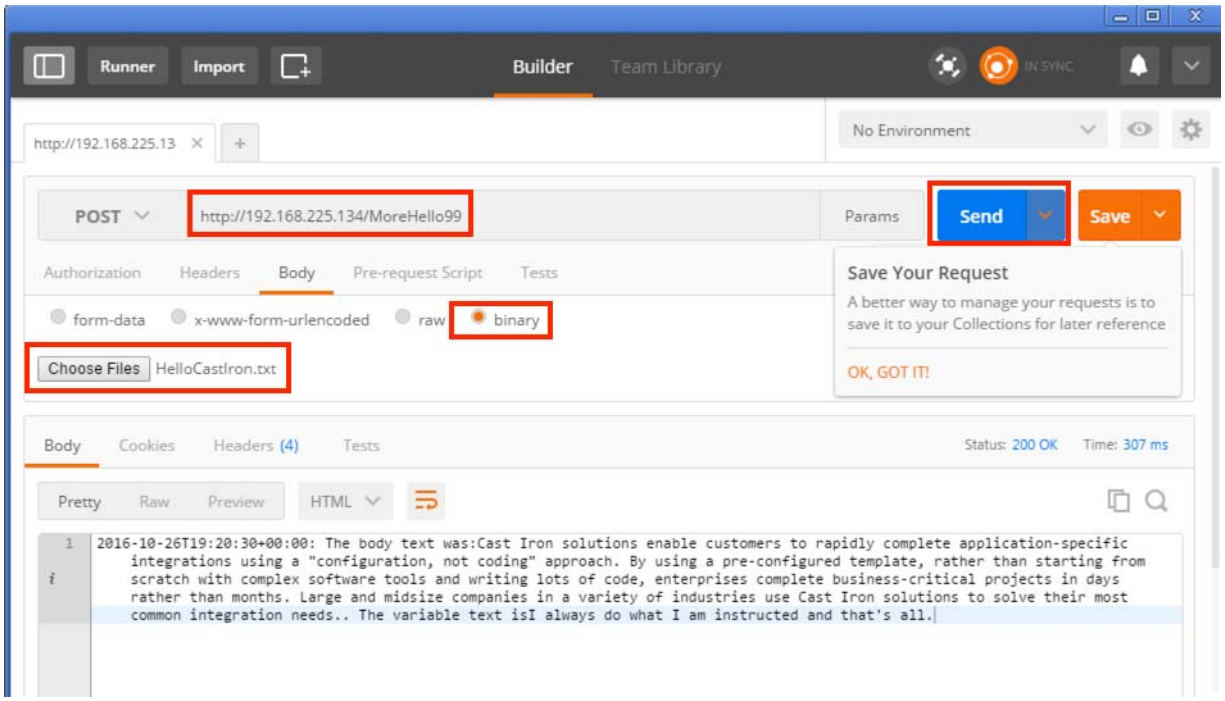
__ 22. Start Configuration.

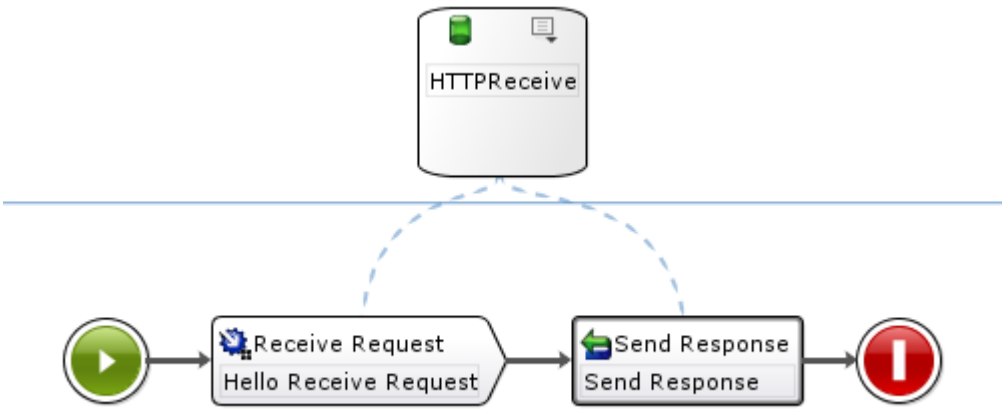


The configuration is now running.



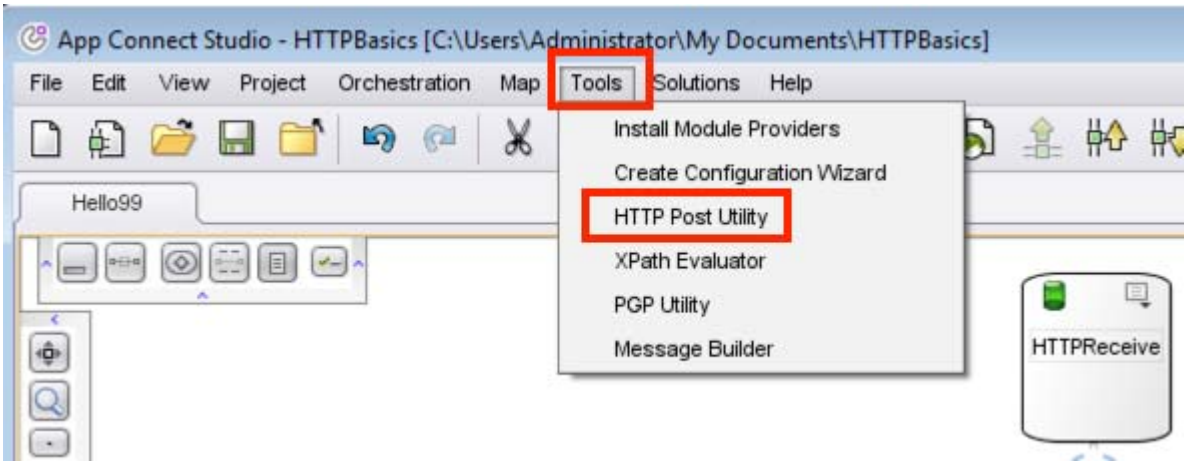
__ 23. Test the project. Use the Postman utility inside Google Chrome, which can be launched by going to the desktop and clicking the **Chrome App Launcher** on the desktop. Use the EData port of your appliance as the host name, and then the URL of your orchestration as the URI. Ensure also that the binary encoding type is selected and choose the requisite file, which is `c:\student\Lab Resources\Lab 1\HelloCastIron.txt`. If it runs correctly, you see the output as shown.



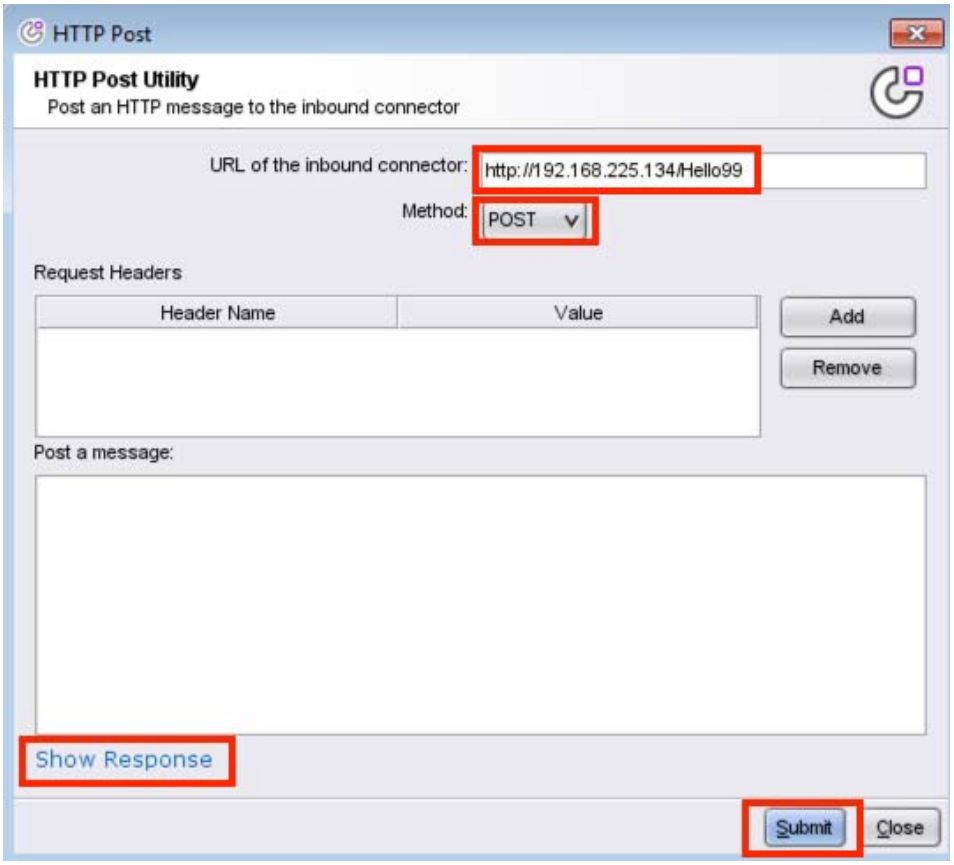


1.4. Testing project components

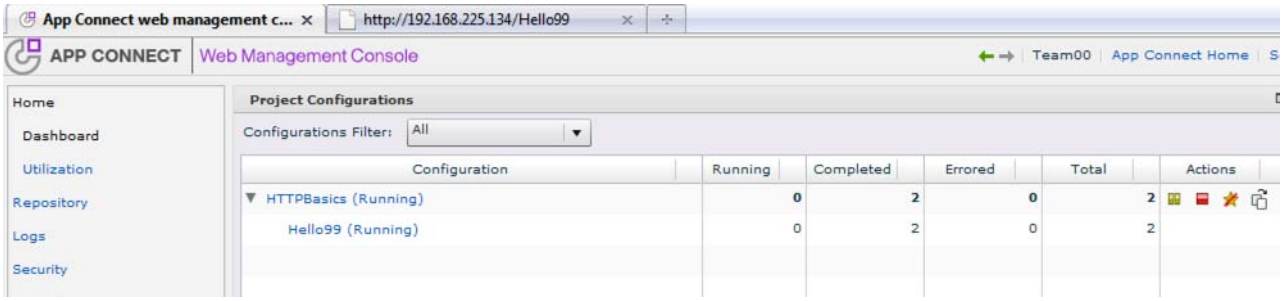
__ 1. Select **HTTP Post Utility** from the toolbar.

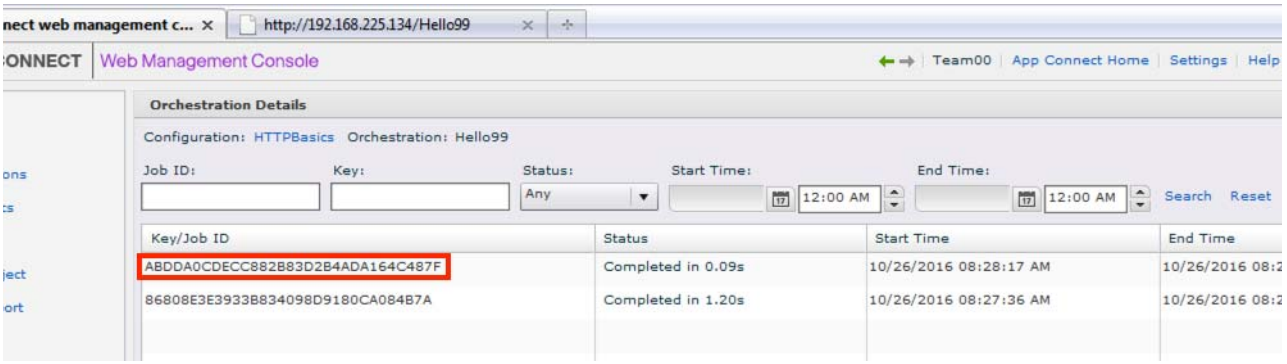


__ 2. Click **Show Response** to open the window that shows the output results. Click **Submit** to execute the orchestration.



__ 3. The WMC shows the results of each run. From the dashboard, expand the configuration and click the orchestration that is being tested to display the Orchestration Details page, which shows the results of each run of the orchestration.

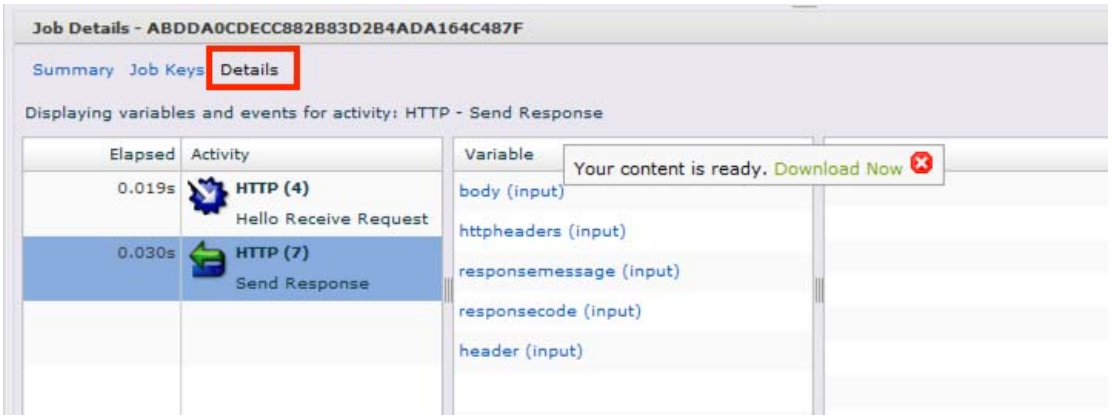




__ 4. Select the Job ID link for the POST run in the top pane. Click the **Details** link in the bottom pane and then the HTTP Send Response section. Click the **body (input)** link to dynamically generate a file with the variable's contents.

This action opens a file that contains the response that was displayed in the browser.

Subsequent labs present better opportunities for viewing input and output values.



End of exercise

Exercise review and wrap-up

In this exercise, you created a new project with an orchestration that receives an HTTP request and sends back a response.

Exercise 2. Loading accounts from a flat file

Estimated time

00:90

Overview

In this lab, you create a new project, FlatFileToSFDC with an orchestration that adds Accounts to Salesforce.com.

Objectives

After completing this exercise, you should be able to:

- Configure the HTTP Receive Request
- Configure the Transform Read Flat File activity
- Configure the For Each statement that is used to loop through multiple records that come from a source of data
- Configure the If Then…Else statement that is used to add Business Rules or Business Logic
- Create objects within Salesforce
- Create log messages that are used to create internal log messages

Introduction

In many cases, legacy applications need to be accommodated. Many of the applications use flat files to exchange data with other applications. Most ERP applications also support flat file imports and exports. To use flat files in App Connect Professional integrations, the file must be described in a flat file schema. The **Transform Read Flat File** activity parses the schema to transform data from string to a structured representation. The **Transform Write Flat File** activity transforms data from structured data to string.

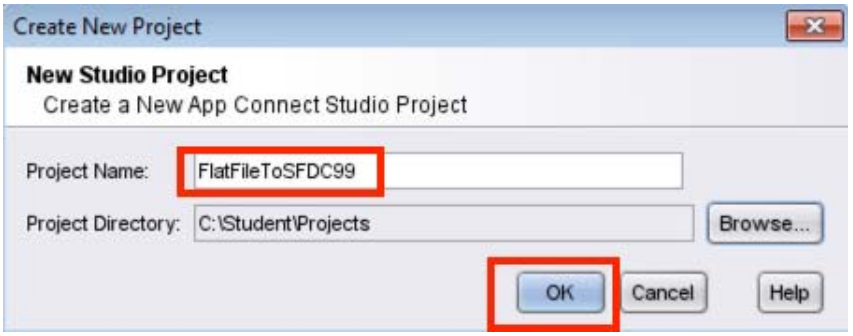
The **Transform Read/Write Flat File** activities are similar to the **Transform Read/Write XML** activities. One significant difference is that a flat file schema defines one document type, whereas an XML schema can define a collection of document types.

Exercise instructions

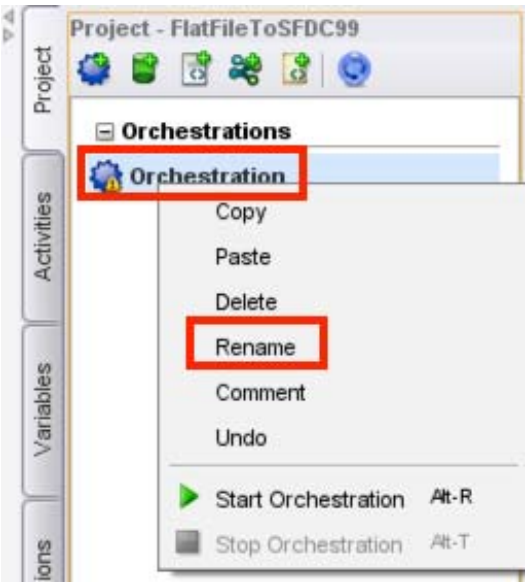
2.1. Flat File to Salesforce.com

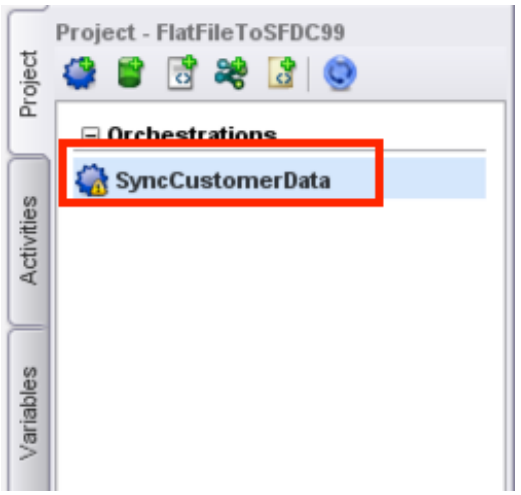
Before you begin this step, follow the prework steps in creating a Saleforce.com account. See the instructor for these steps.

__ 1. In App Connect Studio, create a project FlatFileToSFDC99. Click **OK**.

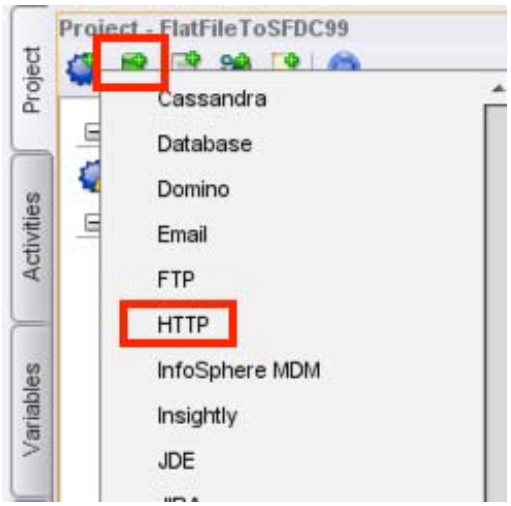


__ 2. Rename the default Orchestration to `SyncCustomerData` by right-clicking **Orchestration** and selecting **Rename** from the list:

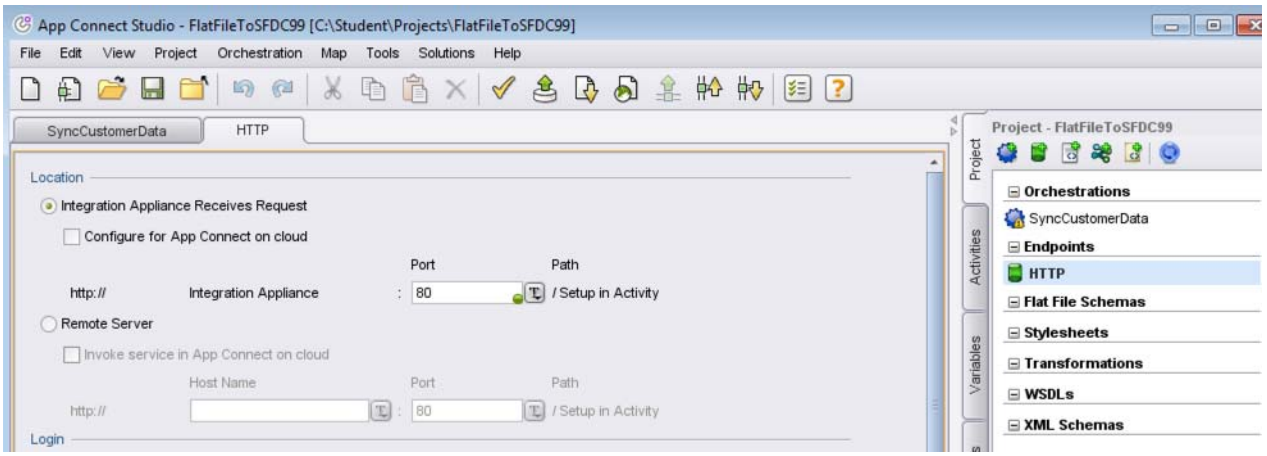




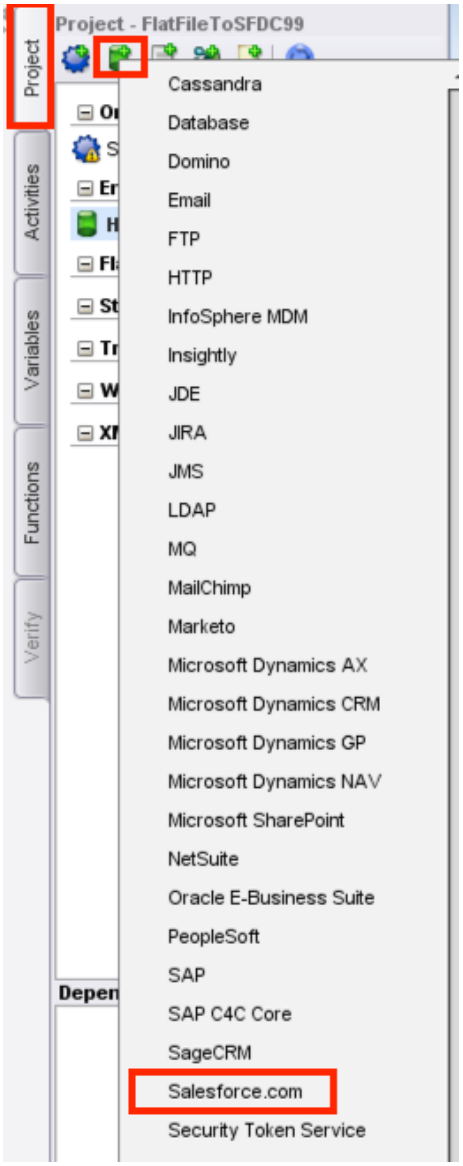
__ 3. Create an HTTP endpoint by clicking the New Endpoint icon and selecting **HTTP** from the list:



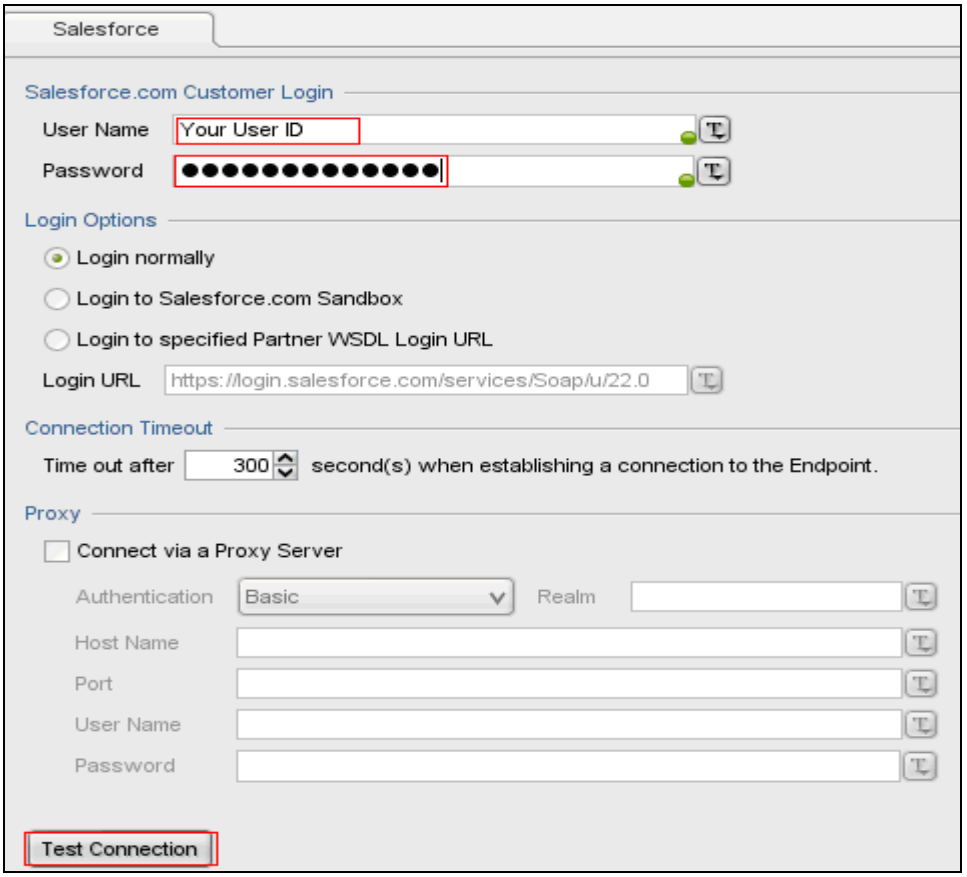
__ 4. This action generates an HTTP endpoint with default settings, which you ***do not*** need to ***change***.

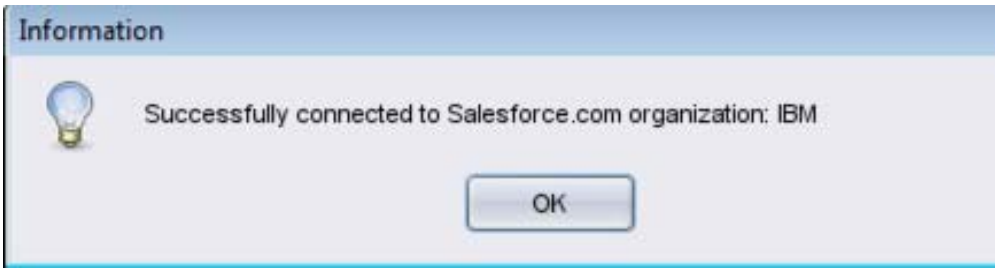


__ 5. Repeat the process to create a Salesforce.com endpoint.



__ 6. Enter your Salesforce.com User Name and Password. Then, click **Test Connection** to validate access to your Salesforce.com instance:

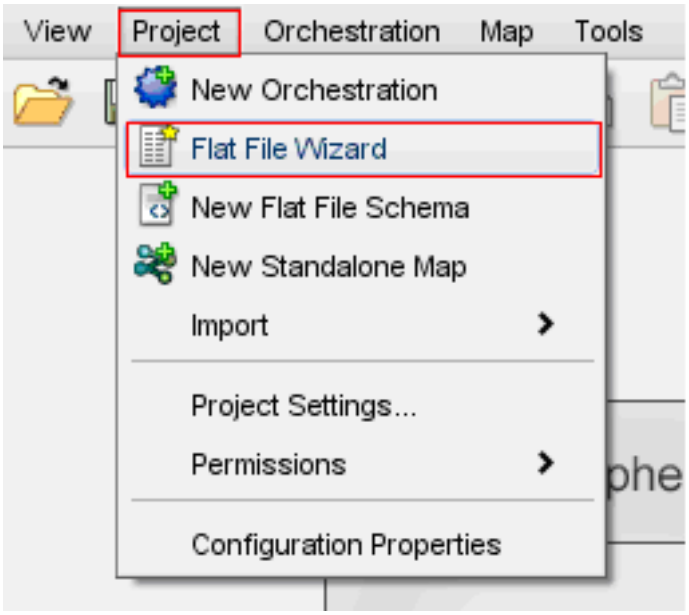




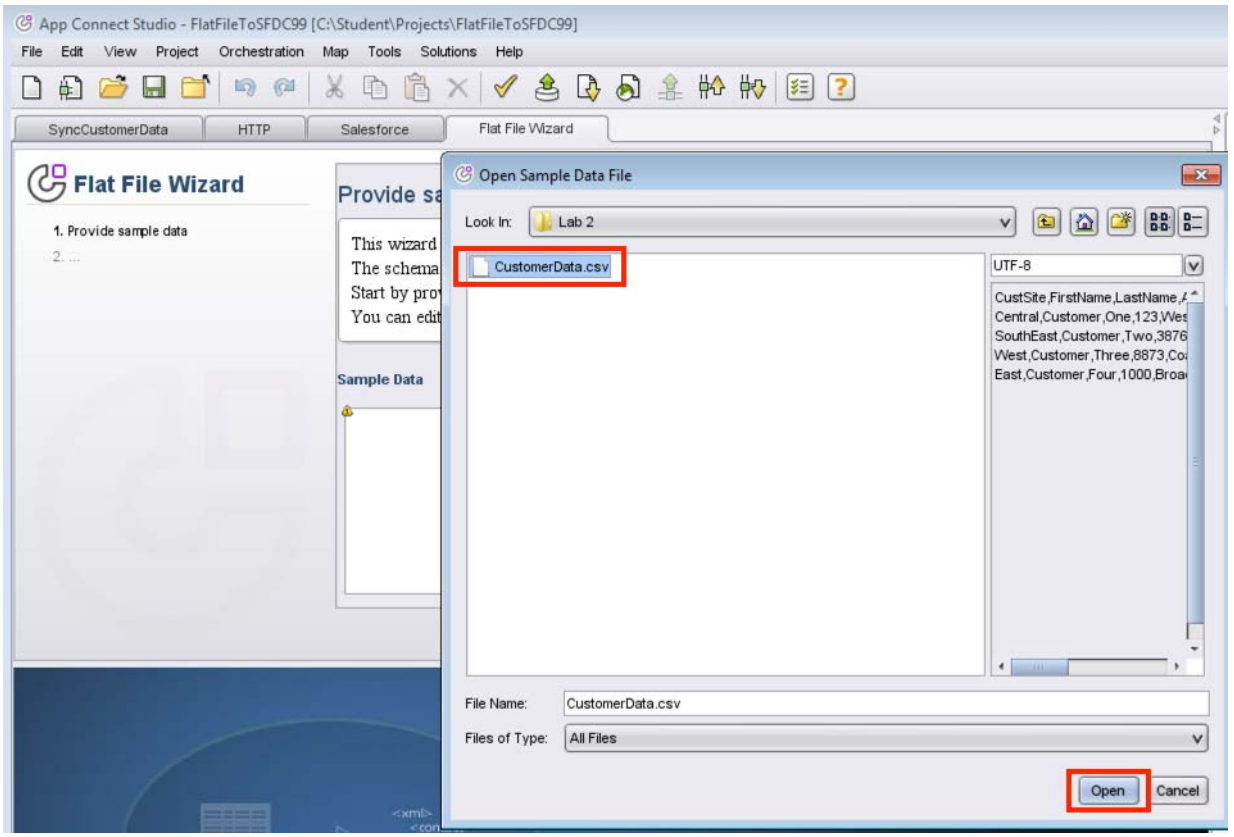
2.2. Creating a flat file schema by using the Flat File Wizard

The schema can be generated by importing a sample data file. The file that is to be imported contains two types of records: Header and Data. The Header indicates the field names. The header and data rows have the same structure of nine fields.

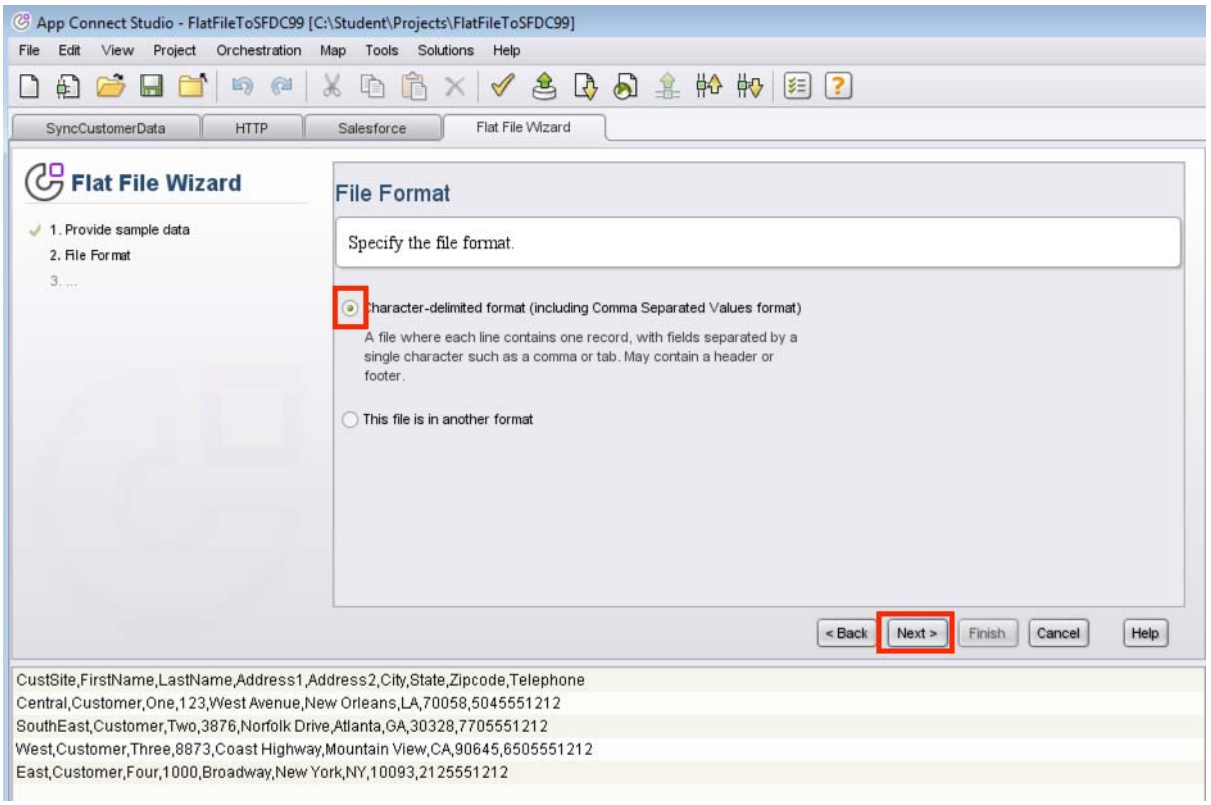
__ 1. Open the Flat File Wizard by using the Project menu: **Project > Flat File Wizard**.



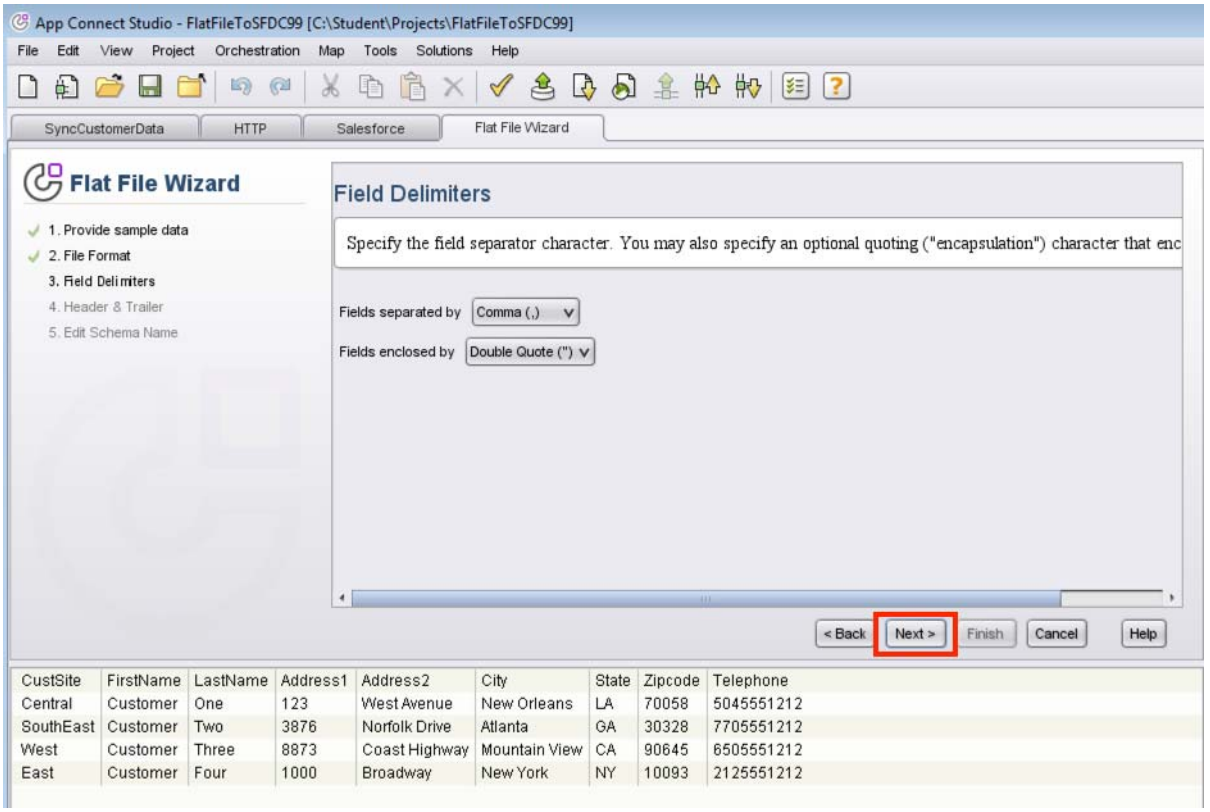
__ 2. Provide sample data by browsing for the input file CustomerData.csv in folder `C:\Student\Lab Resources\Lab 2`. Click **Open**. Click **Next**.



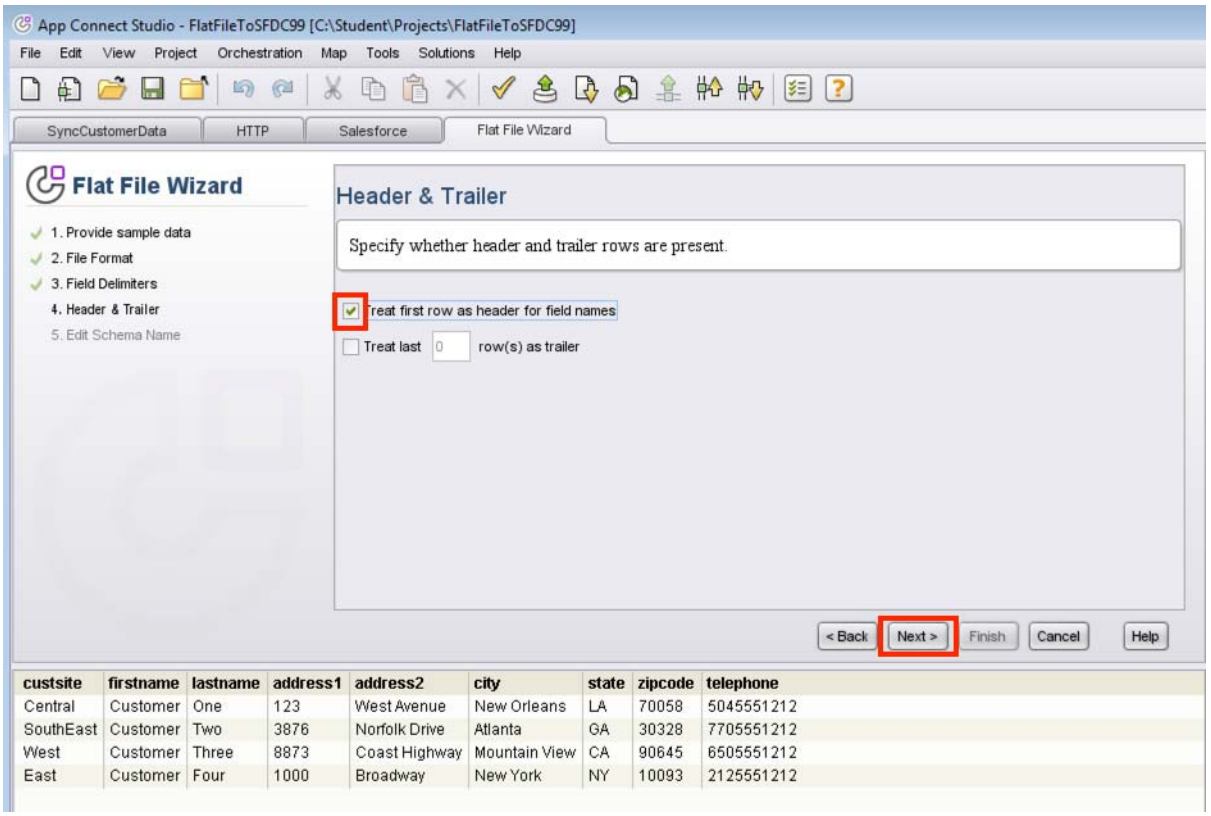
__ 3. **Specify File Format** - The Character-delimited format (default) applies to your data. Click **Next**.



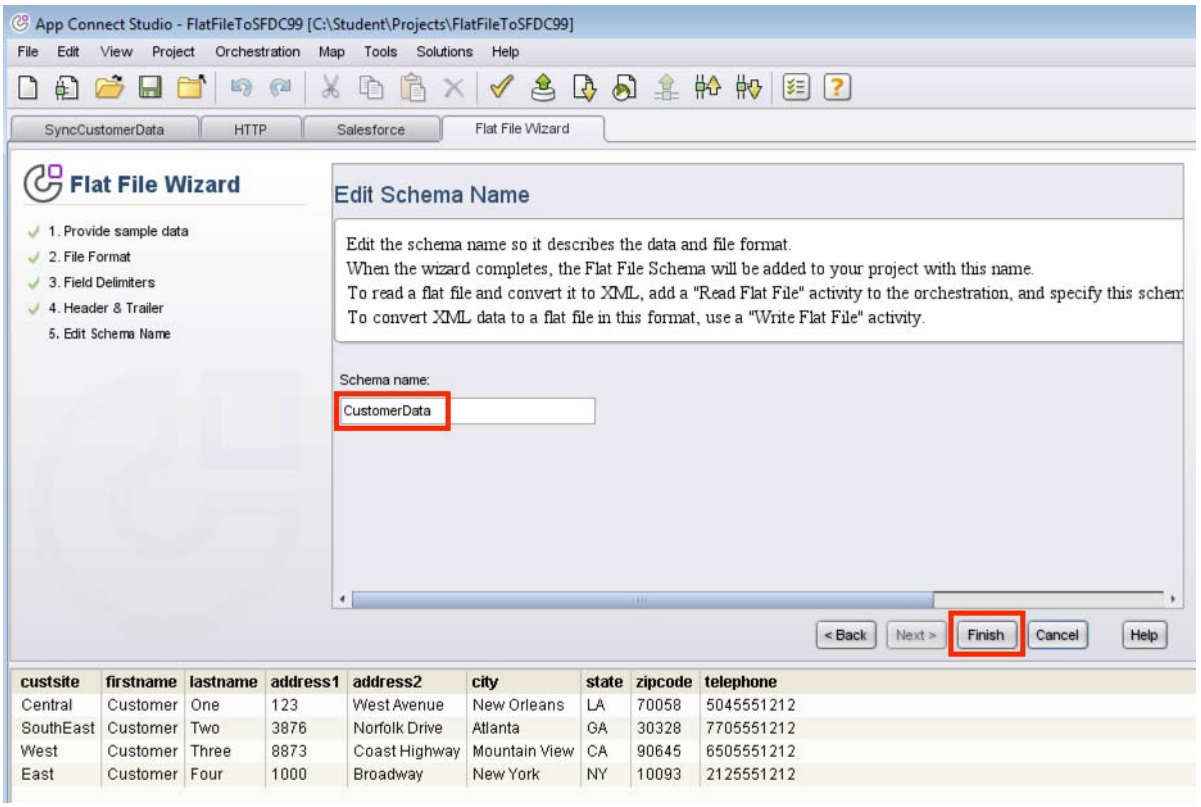
__ 4. **Specify Field delimiters** - Accept the defaults. Click **Next**.



__ 5. **Specify Header and Trailer** - The sample data is constructed to assist the definition of the schema, with a header record, so select just the **Header** option. Click **Next**.



__ 6. **Edit Schema Name**: Taken from the input file name, take the default name of **CustomerData**. Click **Finish**.

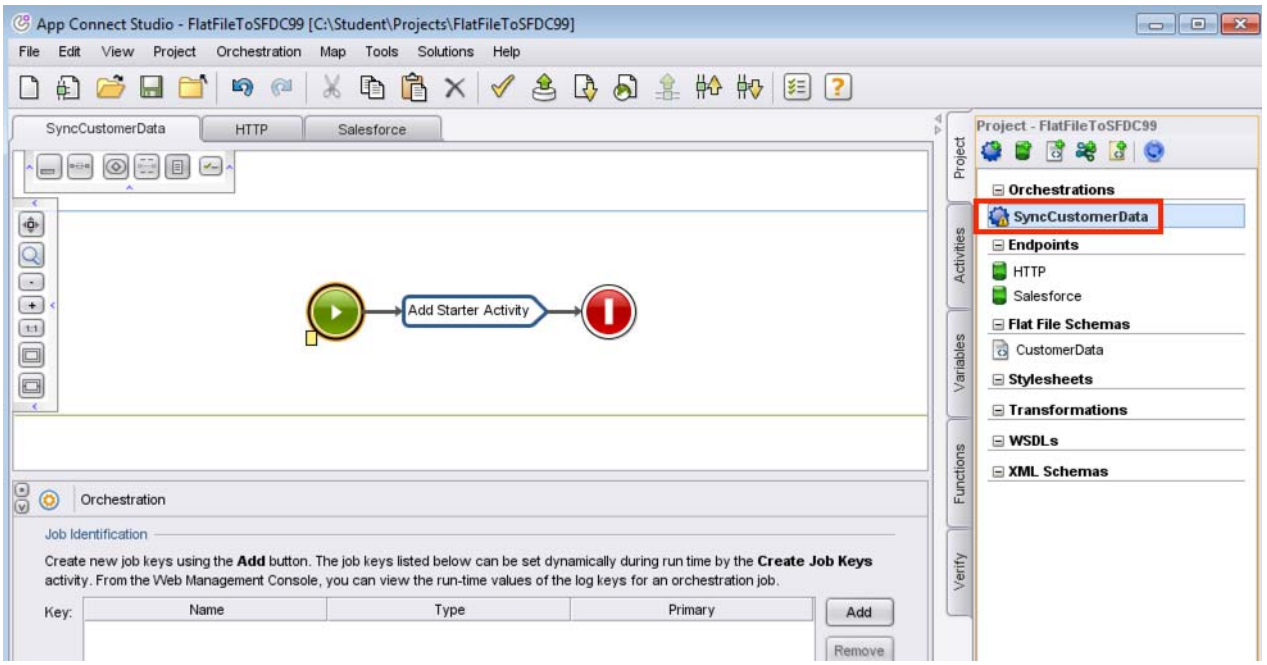


CustomerData schema is added to the **Flat File Schemas** folder.

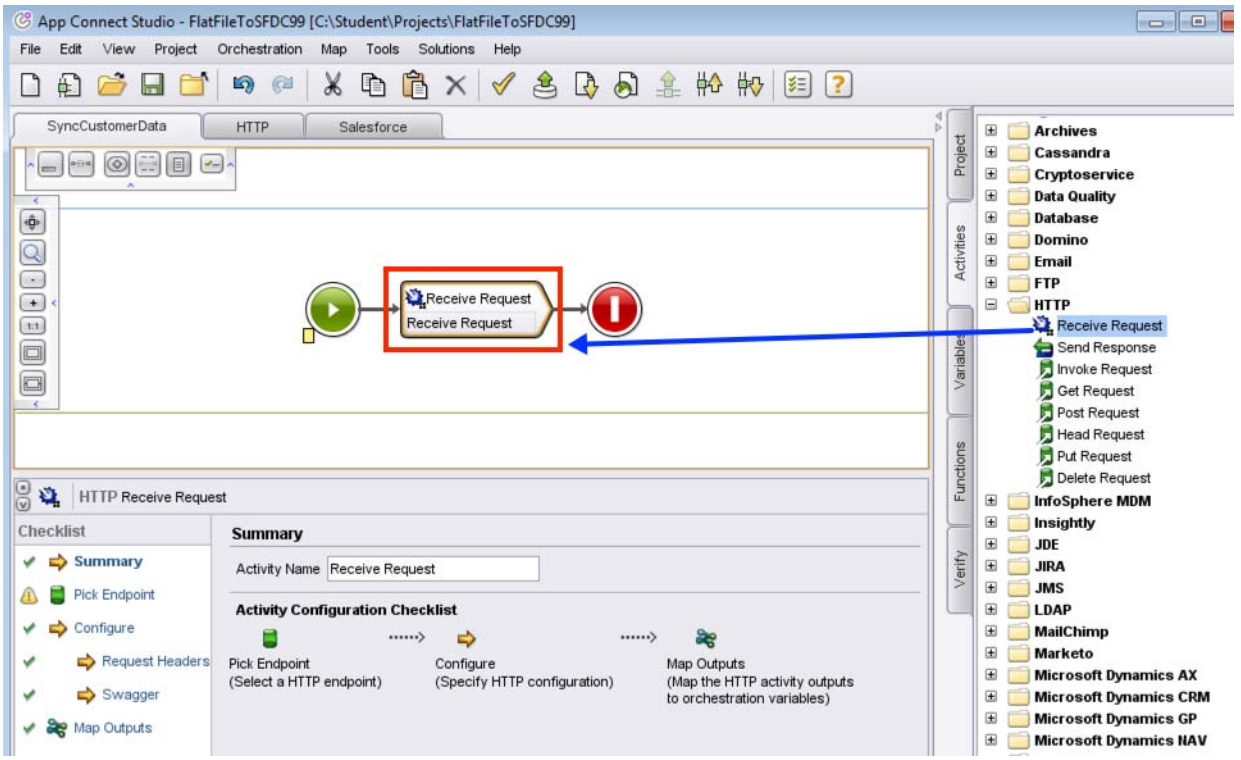
2.3. Creating an orchestration

Build the orchestration

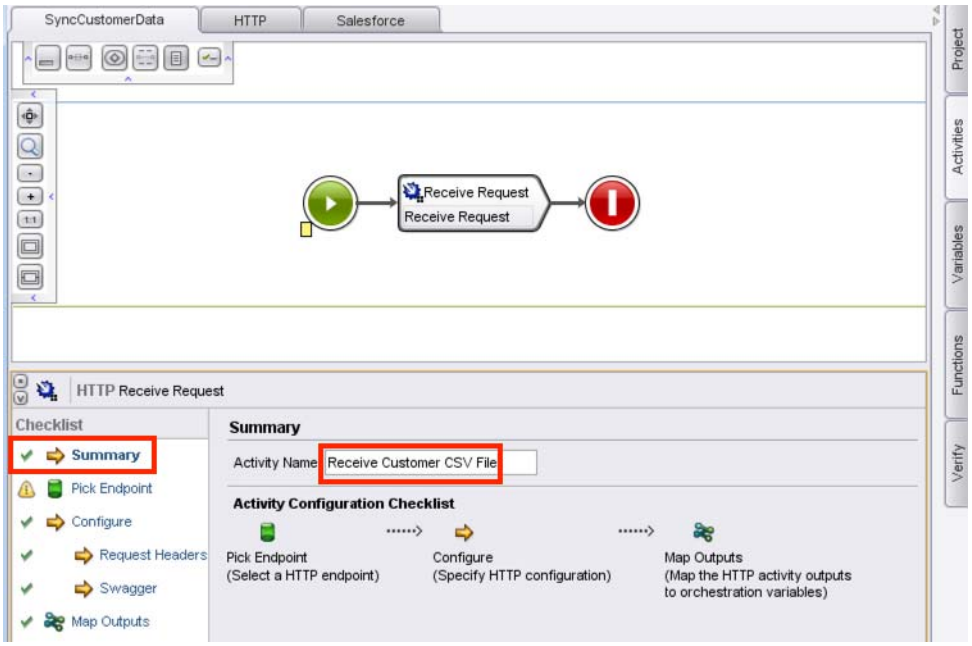
__ 1. Double-click the Orchestration name: **SyncCustomerData**.



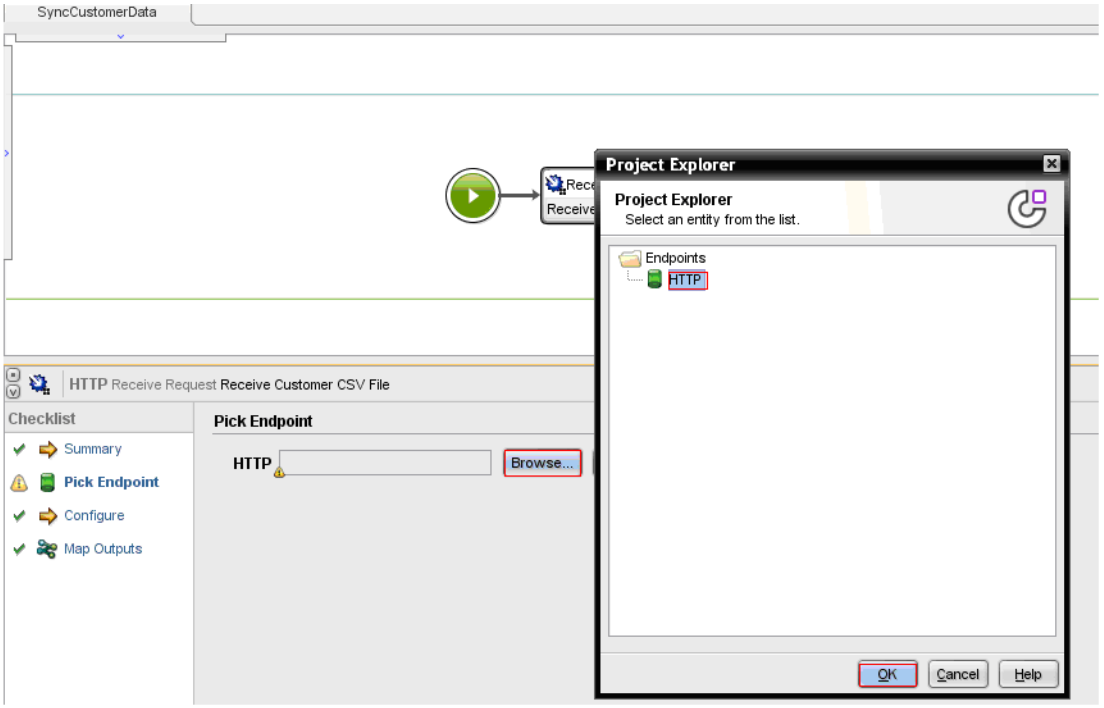
__ 2. Add an **HTTP Receive Request** activity to the orchestration. Left-click and hold on **Receive Request** from the **HTTP** folder on the **Activities Tab** and drag it to the workspace between the green and red icons.



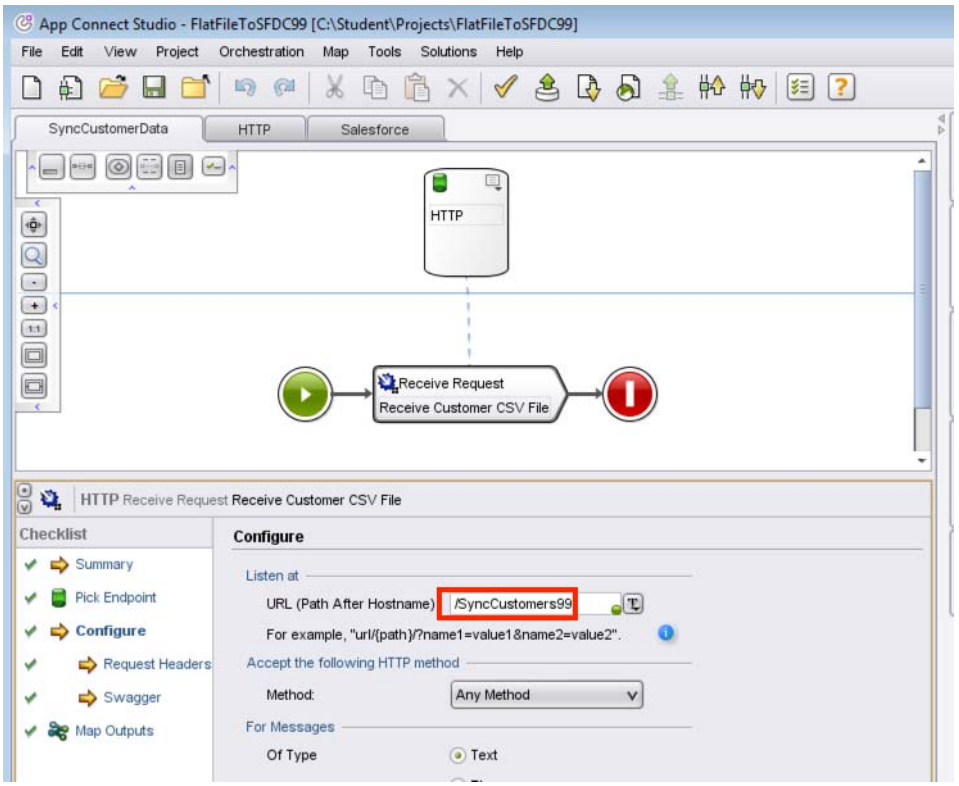
__ 3. Click **Summary** and change the Activity Name to: `Receive Customer CSV File`



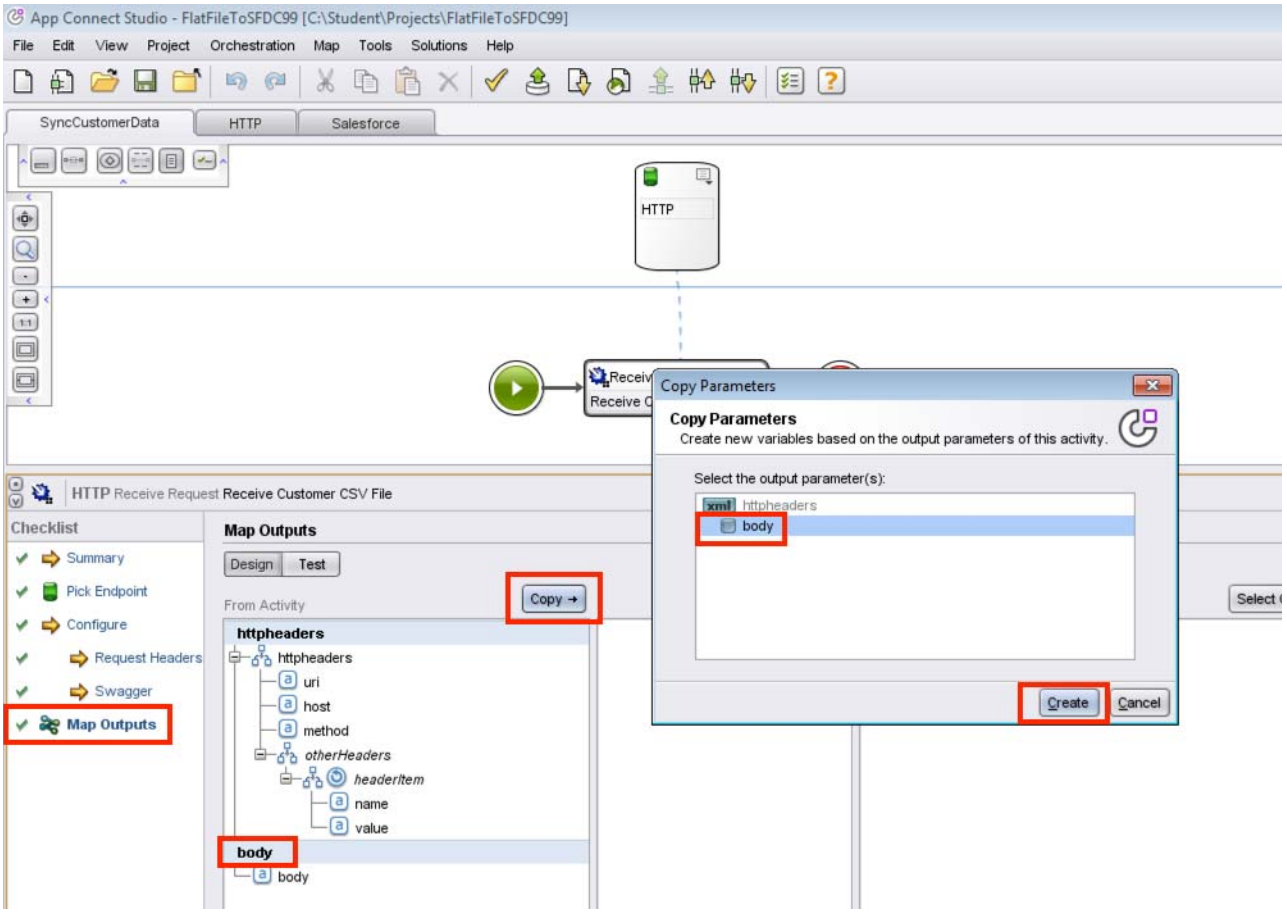
__ 4. Click **Pick Endpoint > Browse**, select **HTTP**, and then click **OK**:



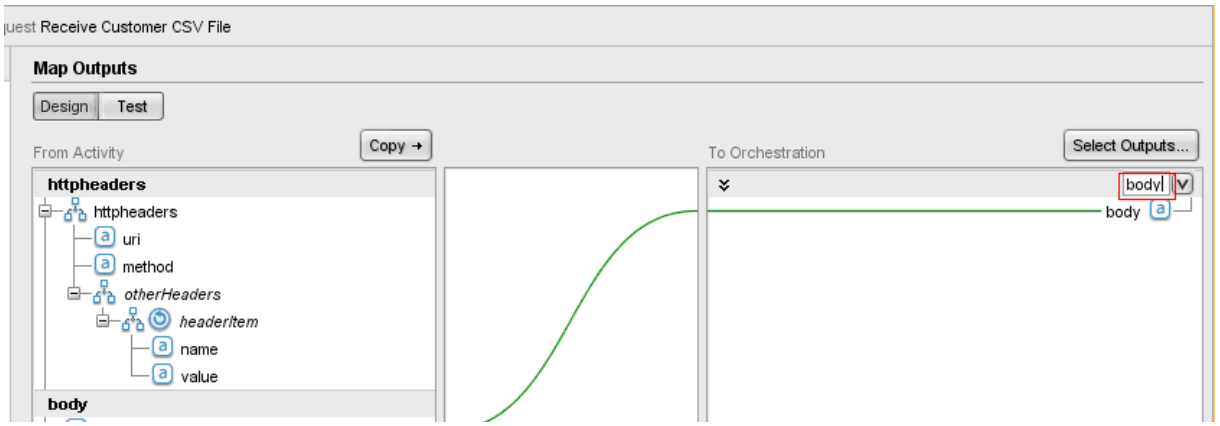
__ 5. Click **Configure** – URL set to: `SyncCustomers99`



__ 6. **Click Map Outputs**: Click **Copy**, and then select body and click **Create**:

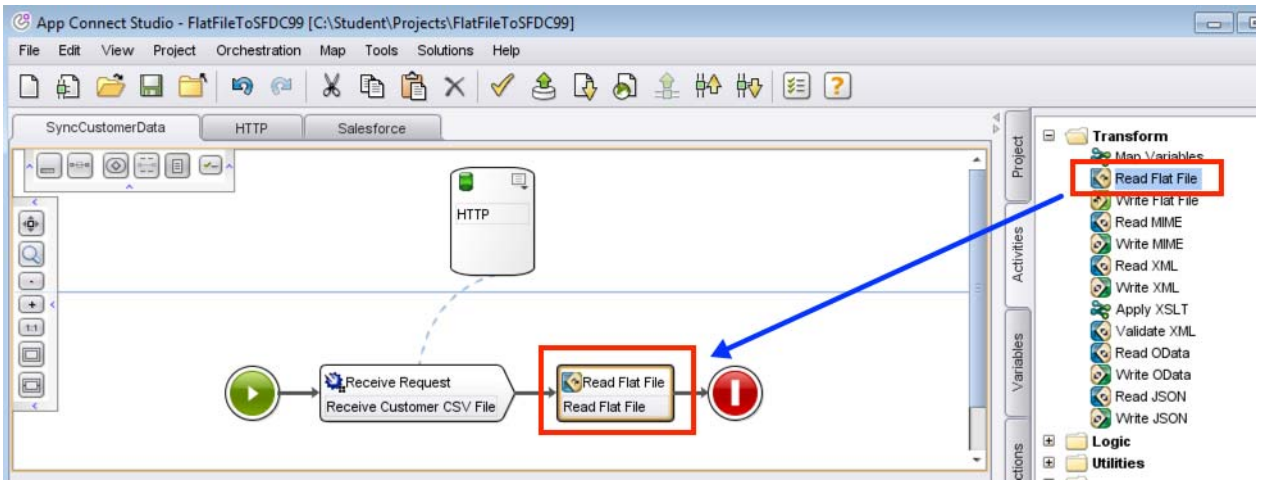


__ 7. To identify the Customer CSV data in the next Activity. Rename the body variable to `csv` by double-clicking the body column name and typing: `csv`
Press Enter.

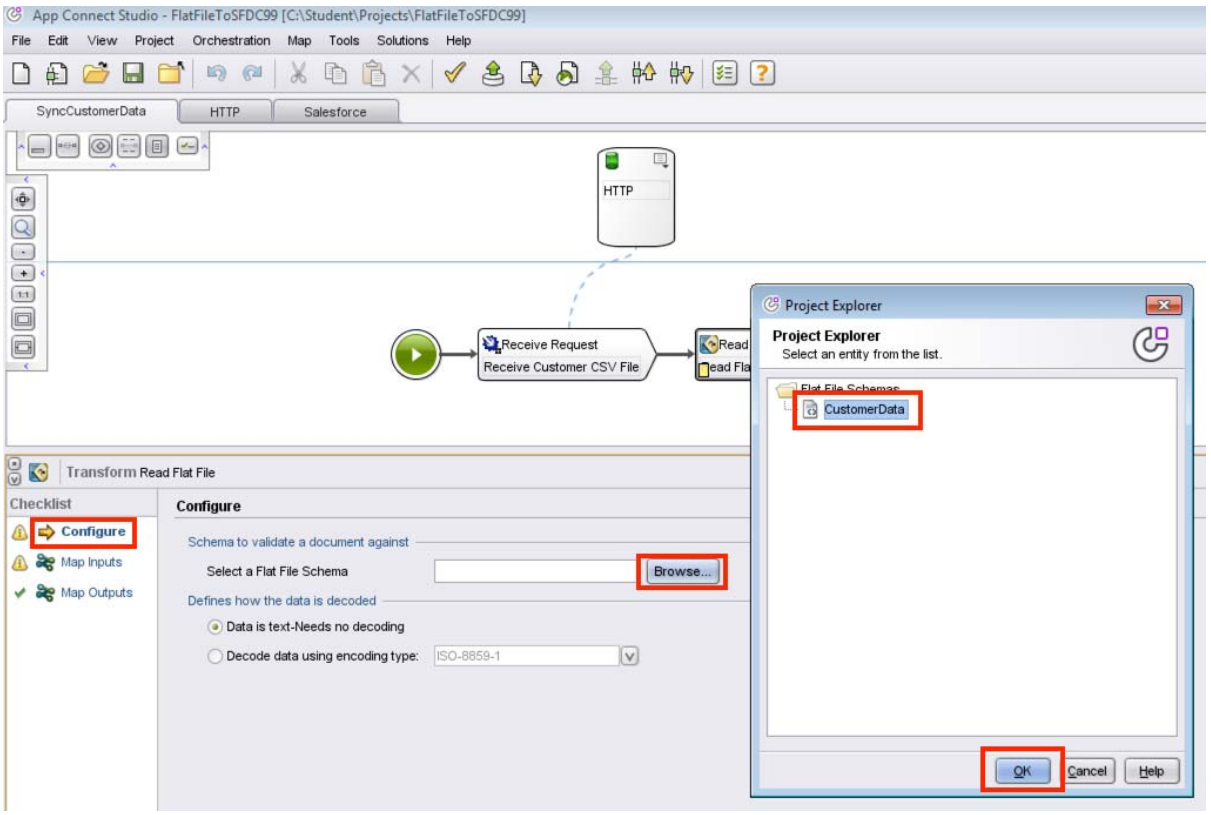


The configuration for the HTTP Receive Request Activity is complete.

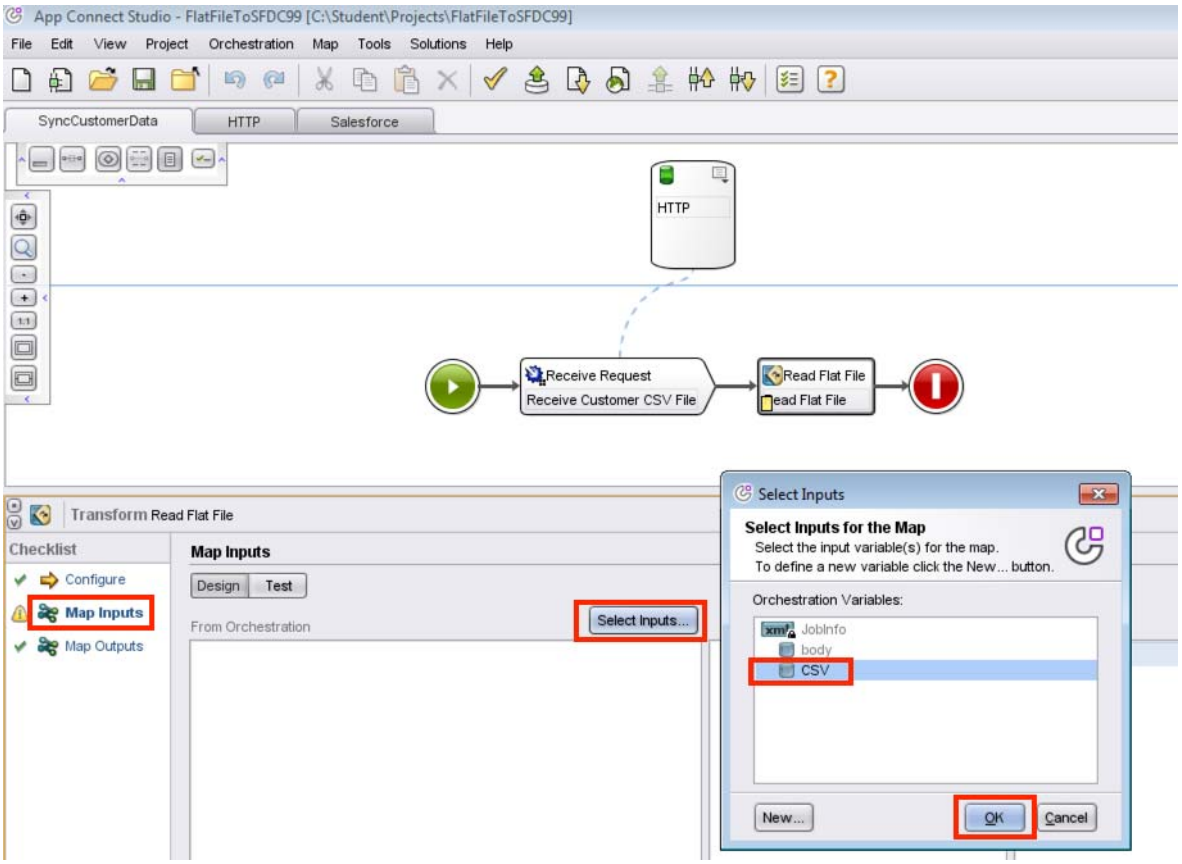
__ 8. Next, add a **Read Flat File** activity to the orchestration from the **Transform** folder on the **Activities Tab** and drag it to the right of the **Receive Request** activity.



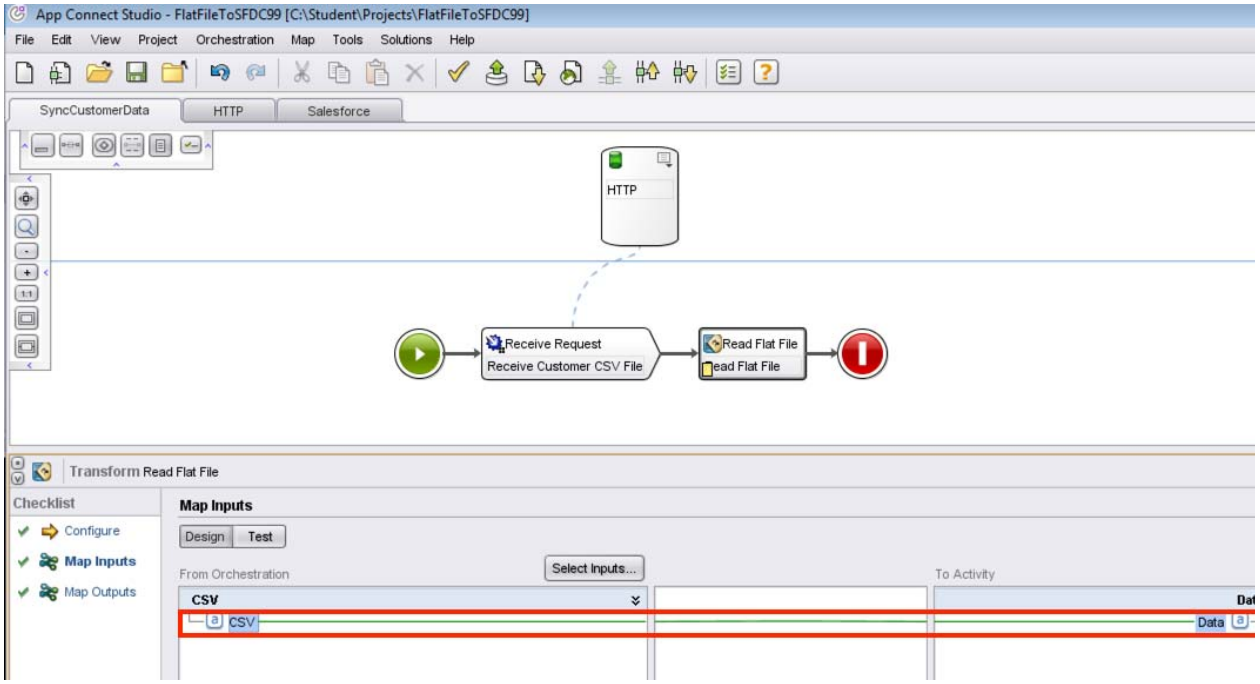
__ 9. Click **Configure**, and then click **Browse** to locate the flat file schema **CustomerData** that was created previously. Click **OK**.



__ 10. Click **Map Inputs**: The input to the **Read Flat File Activity** is the output from the previous **Receive Request Activity**. Click **Select Inputs** and then choose **CSV** and then **OK**.



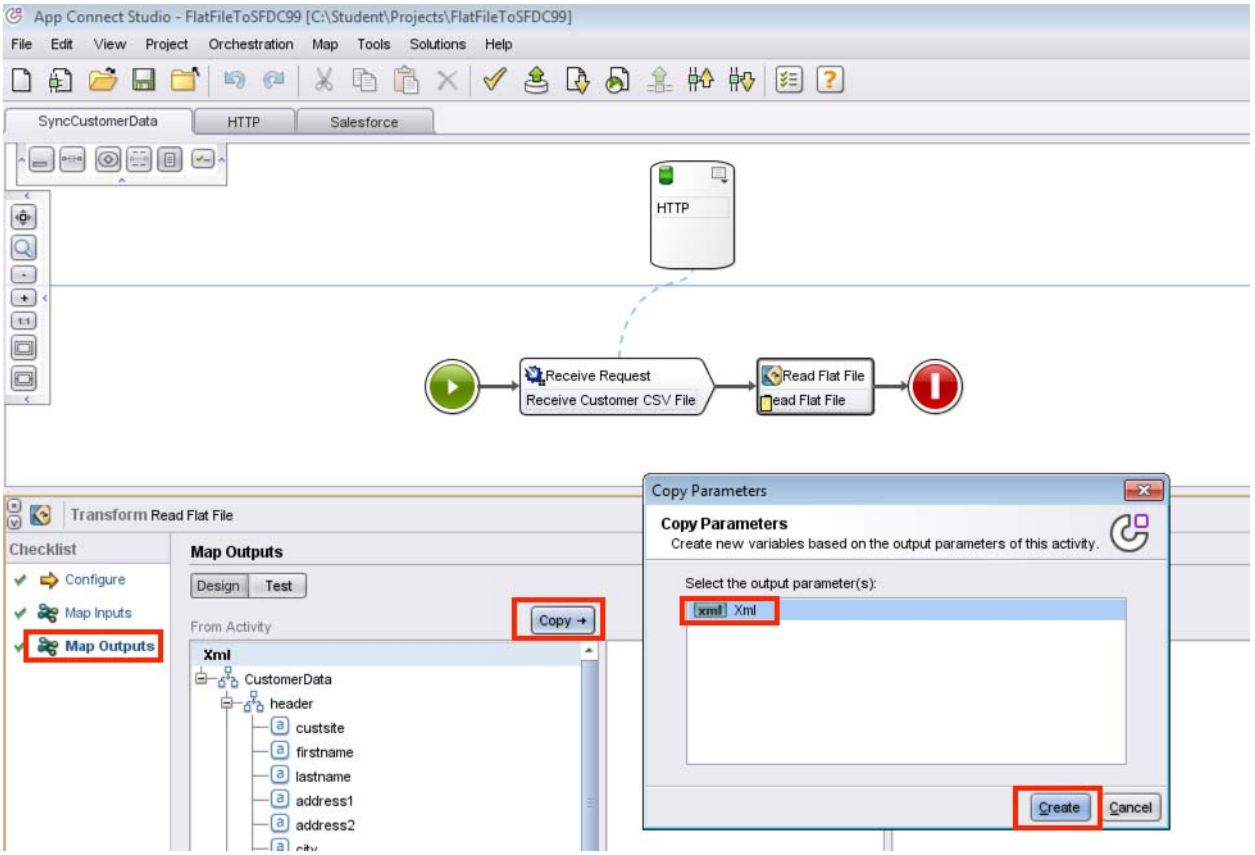
__ 11. Next, position your cursor on the CSV field under the **From Orchestration** column, left-click and hold, then drag your cursor on top of the Data field under the **To Activity** column, and release. This action maps the CSV field to the Data field.

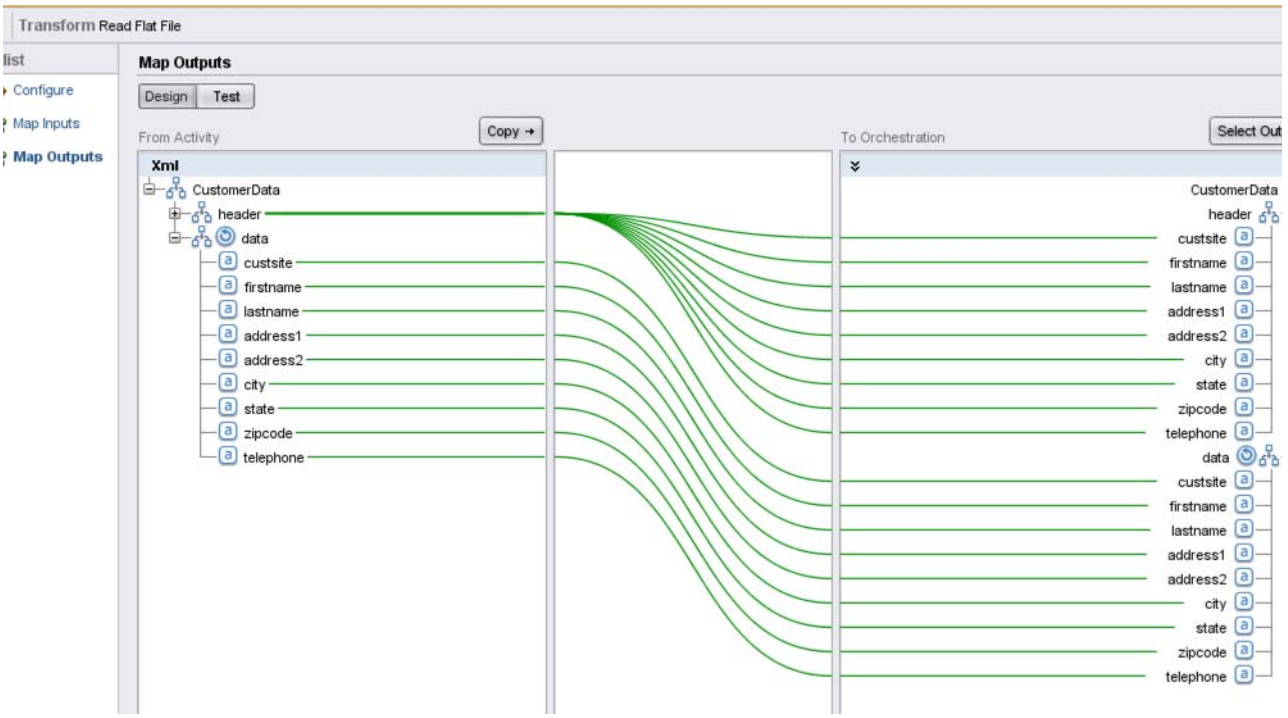


__ 12. Click **Map Outputs**: The output is an XML representation of the input flat file. As described by the schema, the first record of the input is header information and non-data.

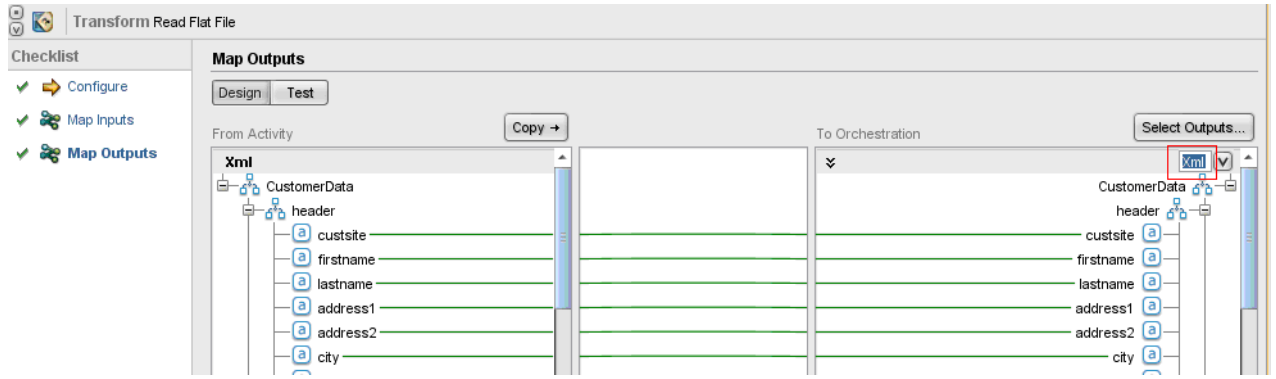
The output structure shows that the data section is a recurring node, with zero or more entries.

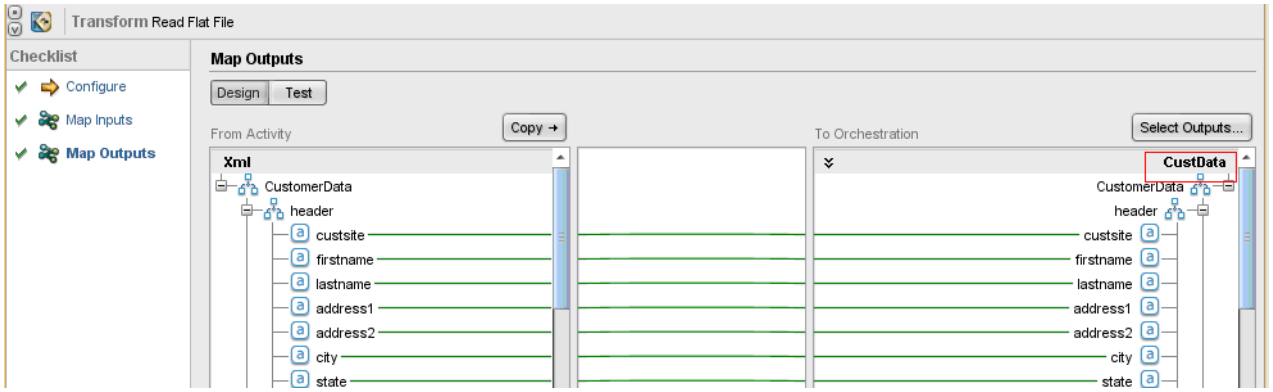
__ 13. Click **Copy**, select **Xml**, and then **Create**:





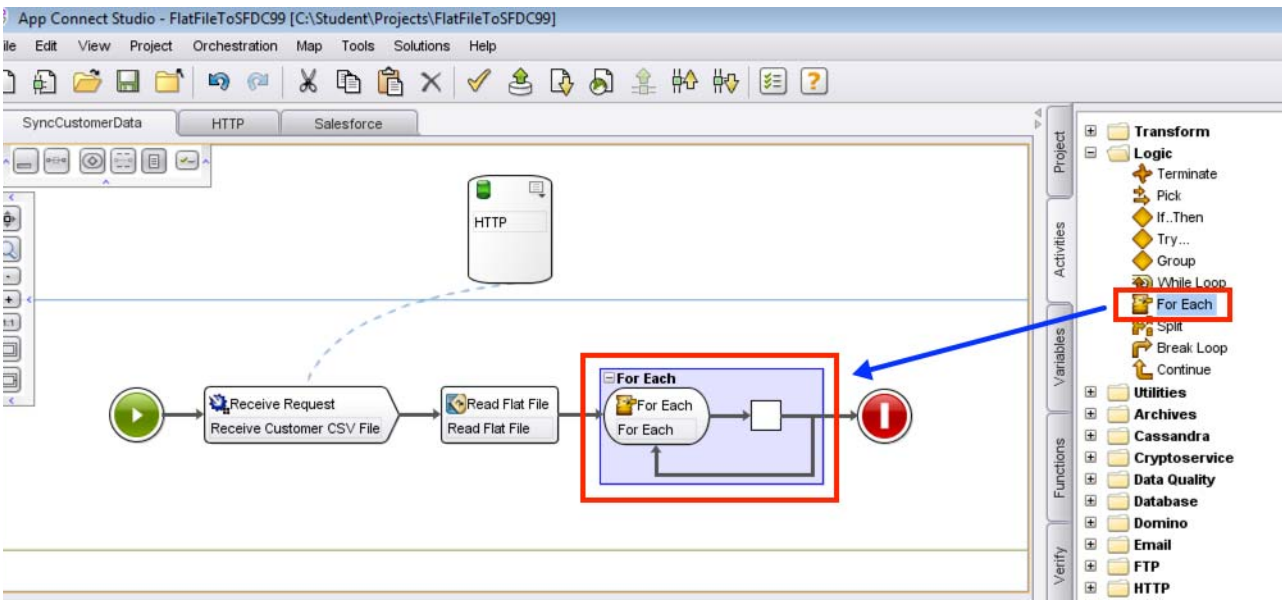
__ 14. Rename **Xml** under the **To Orchestration** column to `CustData` to identify it in the next step:





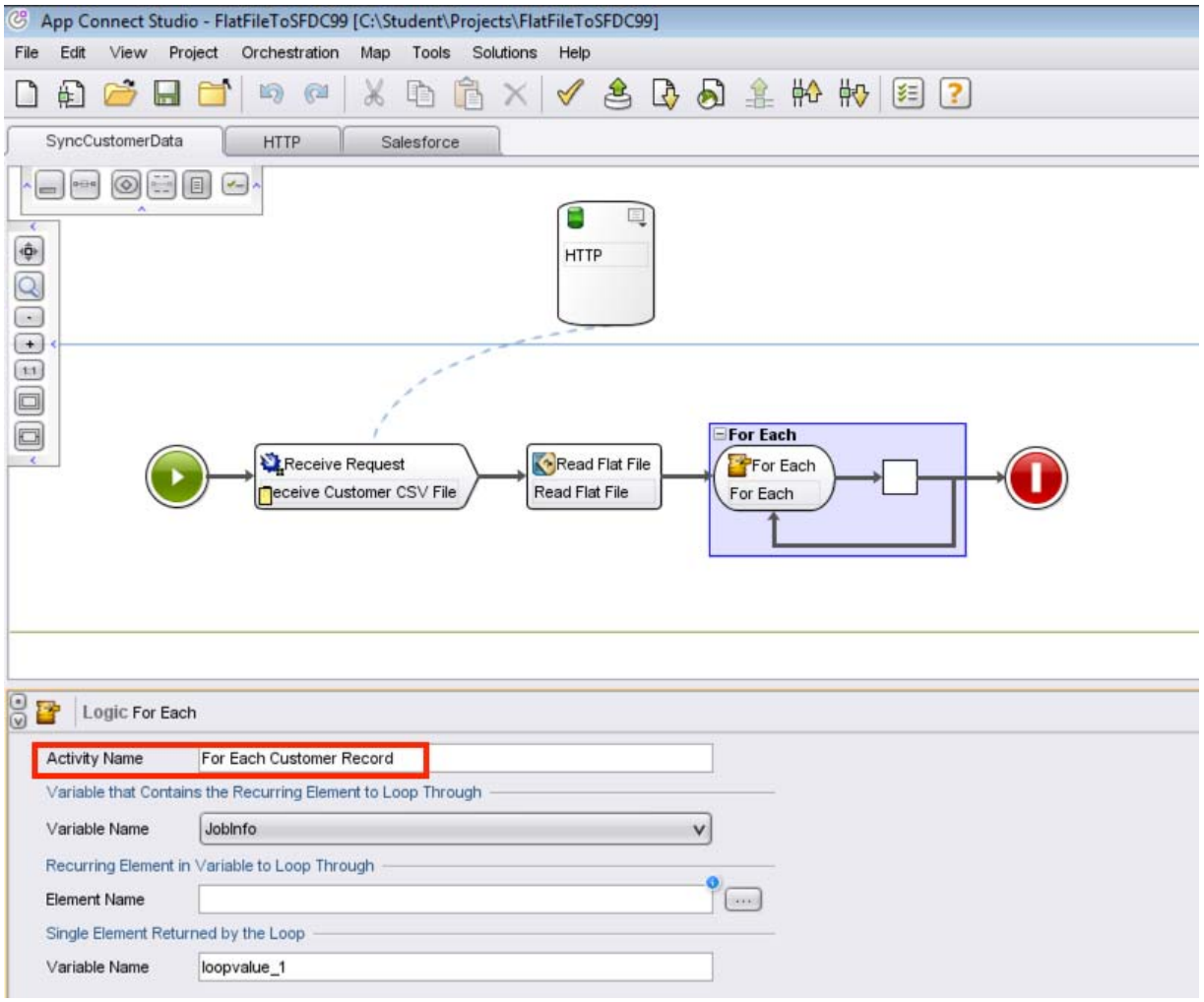
The configuration for the Read Flat File Activity is complete.

__ 15. Next, to accommodate recurring records that are coming from the CSV file, add a **For Each** activity to the right of the **Read Flat File** activity from the **Logic** folder on the **Activities** tab.

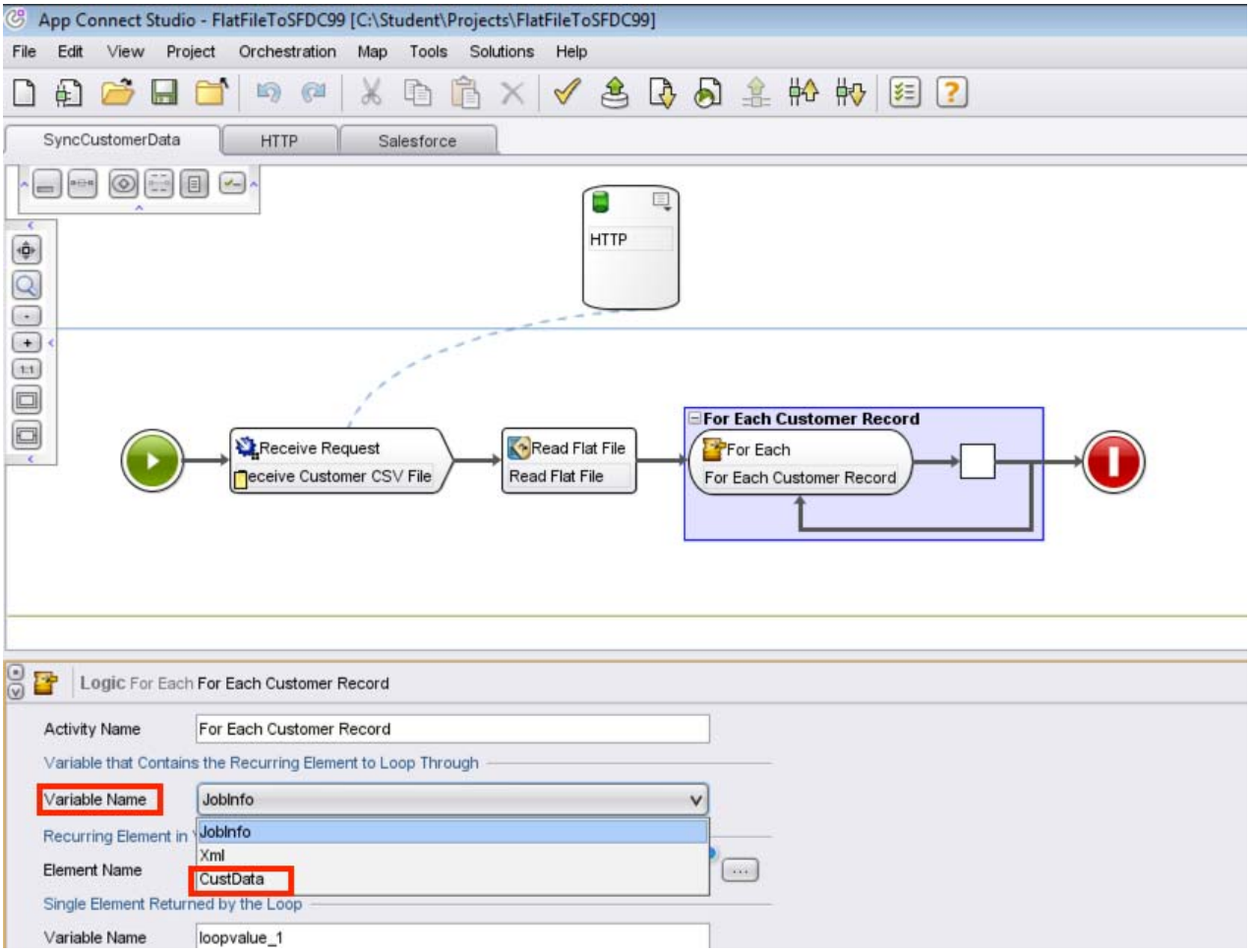


Configure the For Each activity

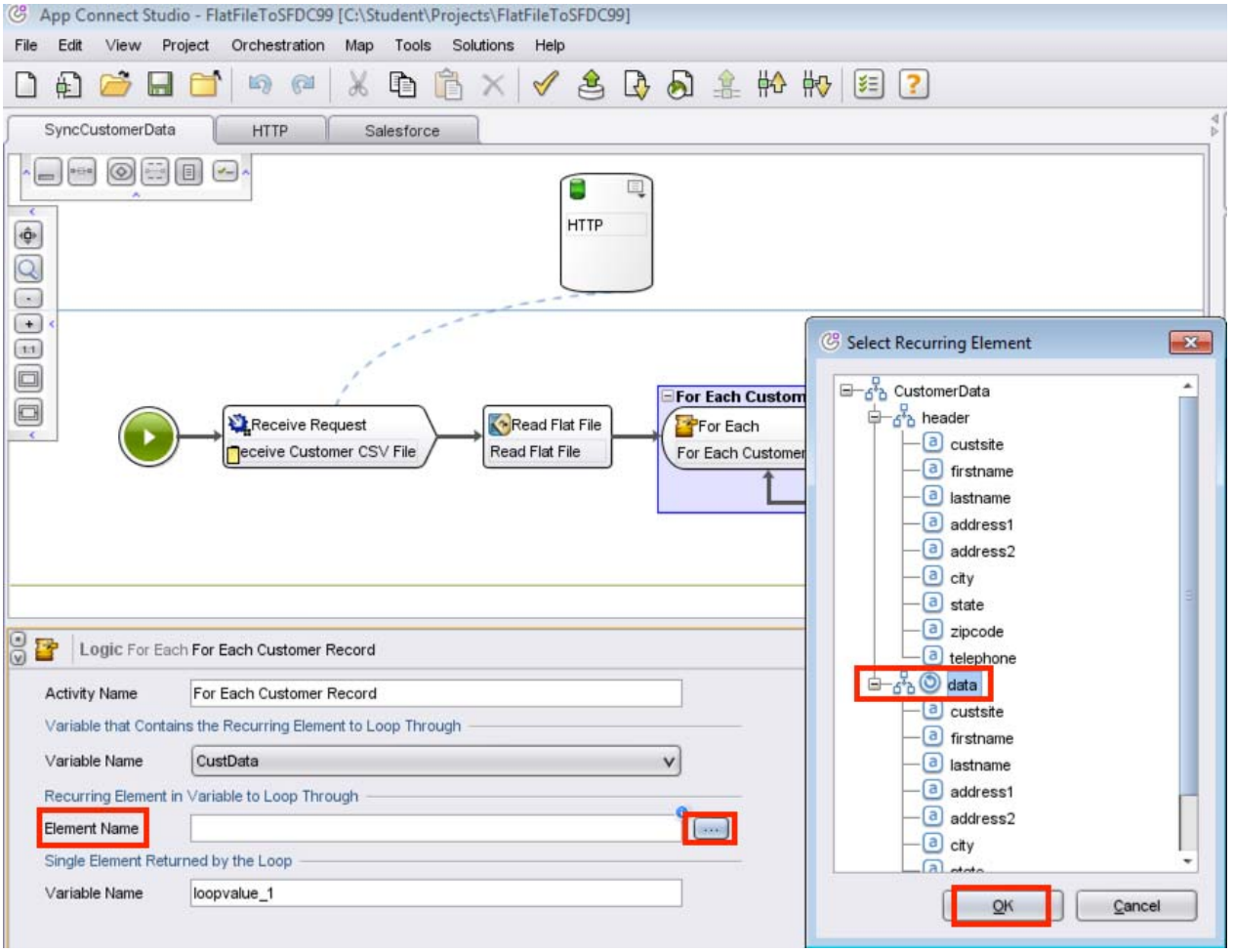
__ 16. Activity Name. Change to: `For Each Customer Record`



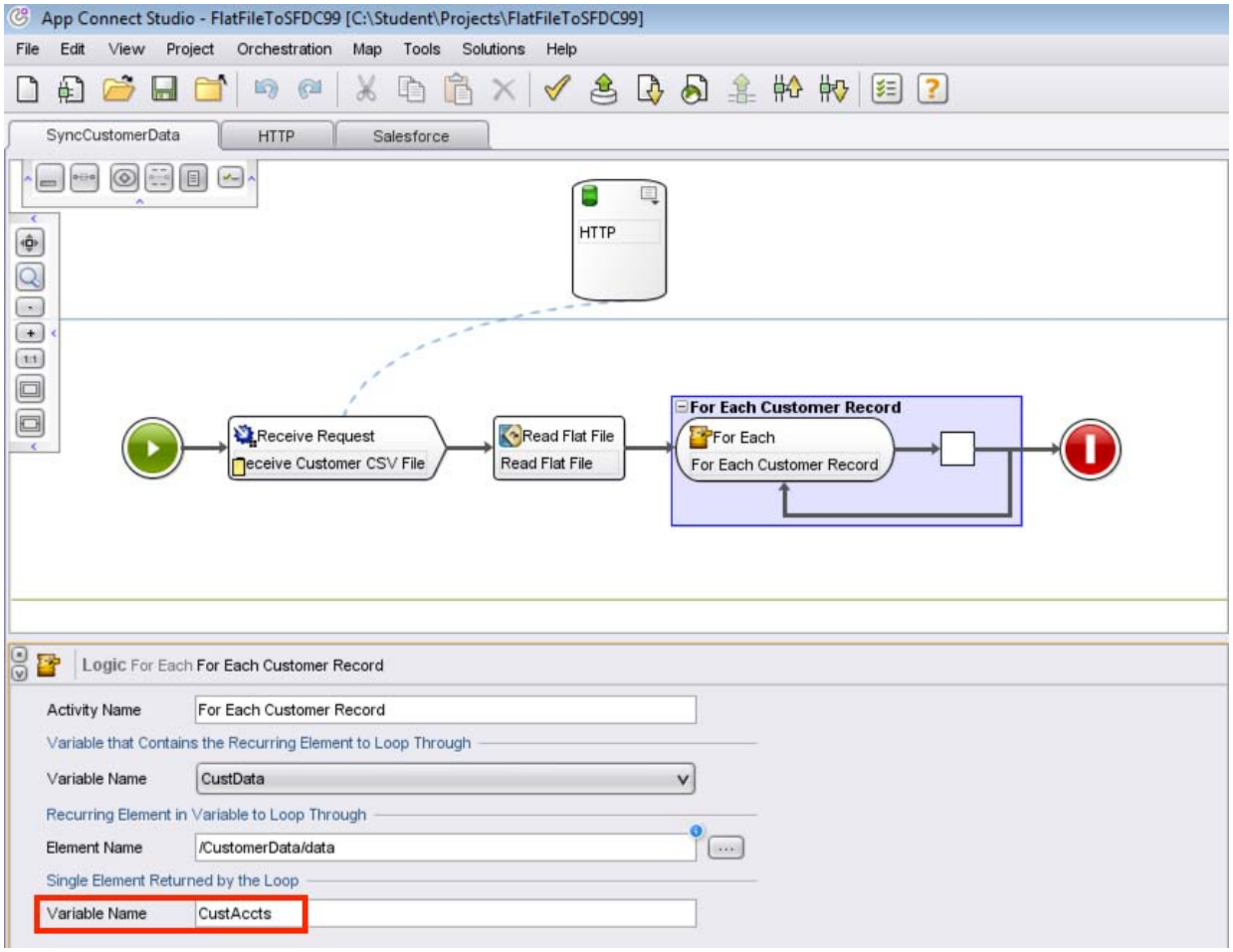
__ 17. **Variable Name**: Select **CustData** from the list.



__ 18. **Element Name**: Click the **…** box next to the **Element Name** field, select the recurring **data** field from the **CustData** list, and then click **OK**.

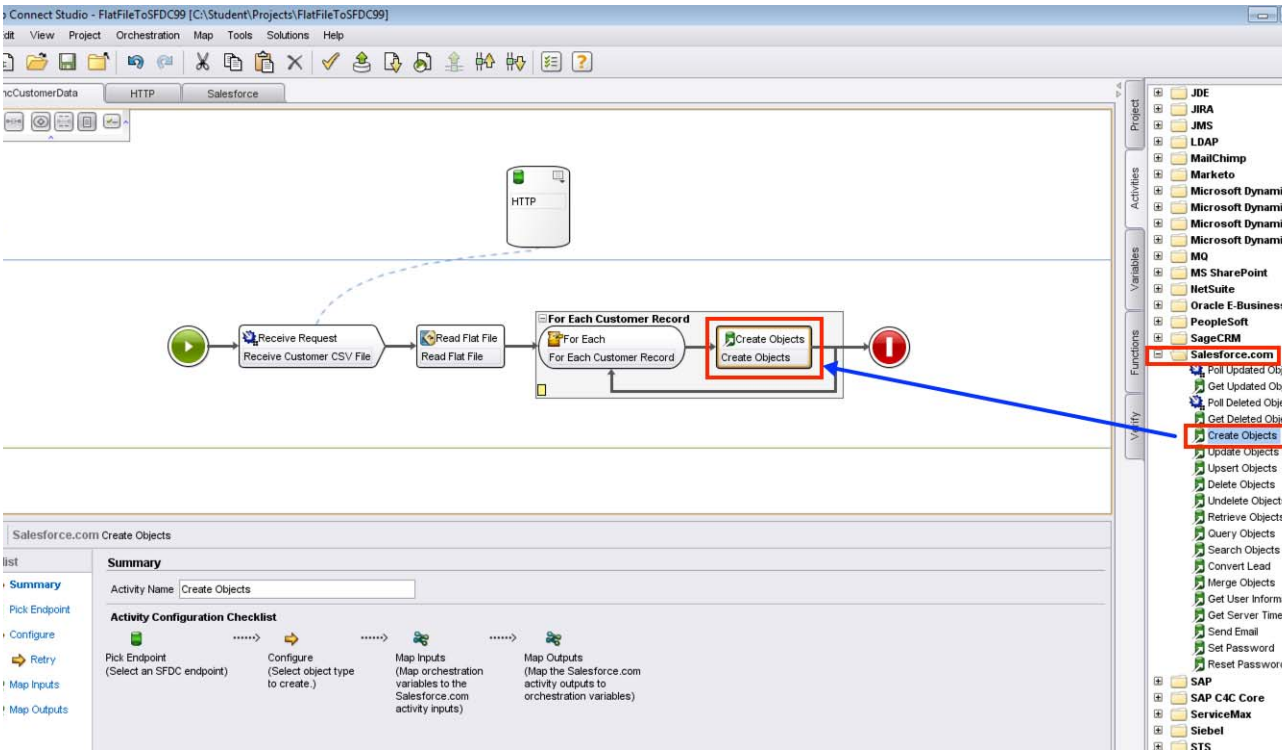


__ 19. **Variable Name**: Change the **Variable Name** parameter to: `CustAccts`



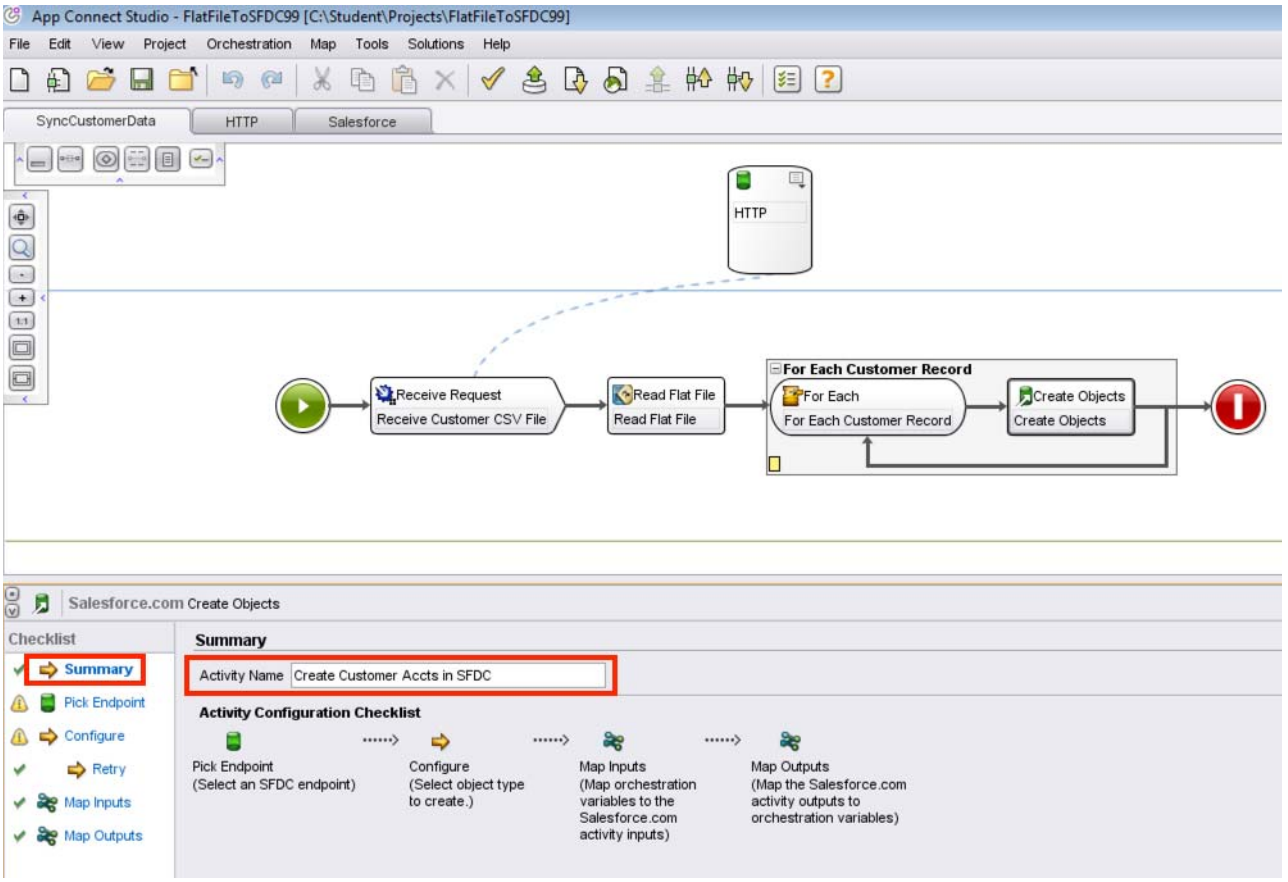
The configuration for the For Each activity is complete.

__ 20. Add a **Create Objects** activity from the **Salesforce.com** folder on the **Activities Tab** next to the **For Each** activity.

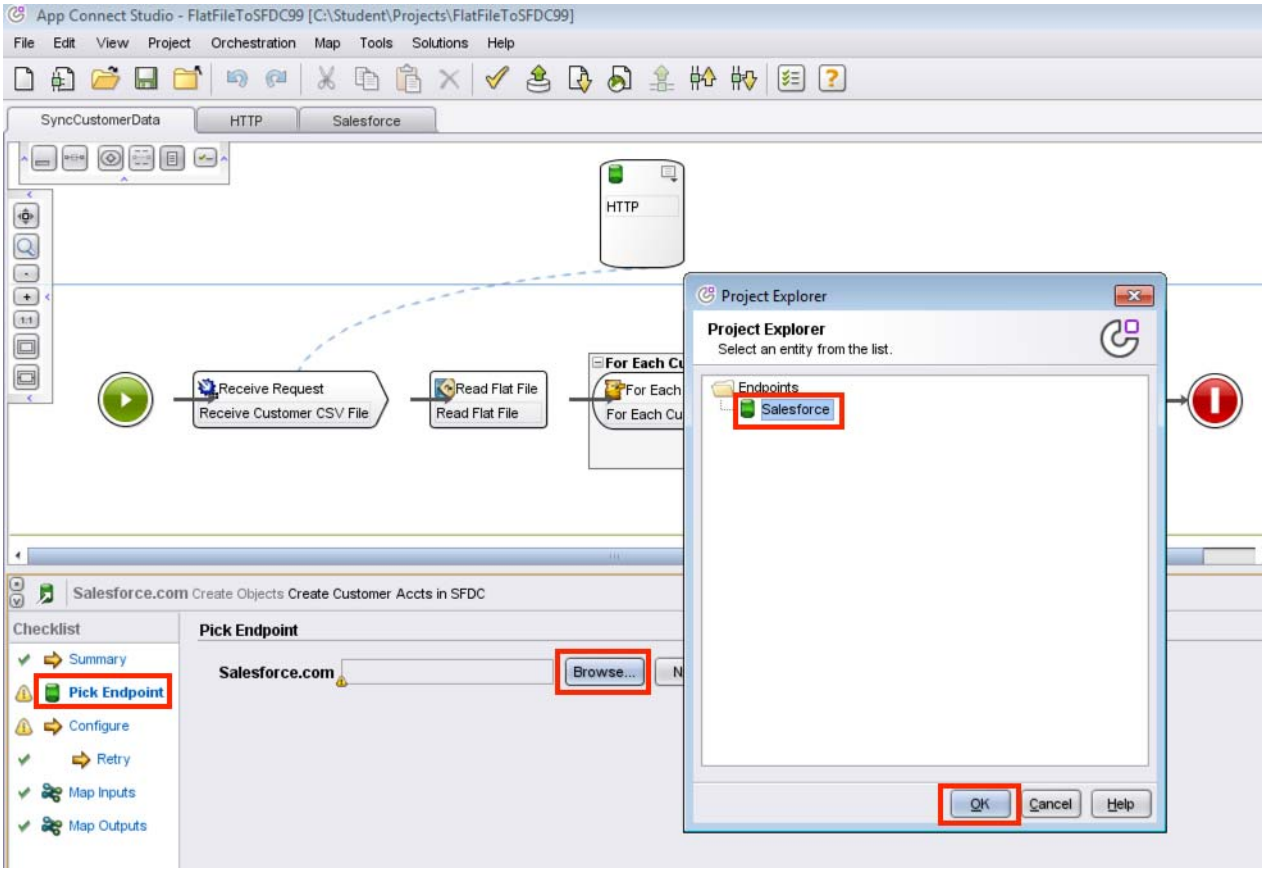


Configure the Salesforce Create Objects Activity by selecting the Checklist items

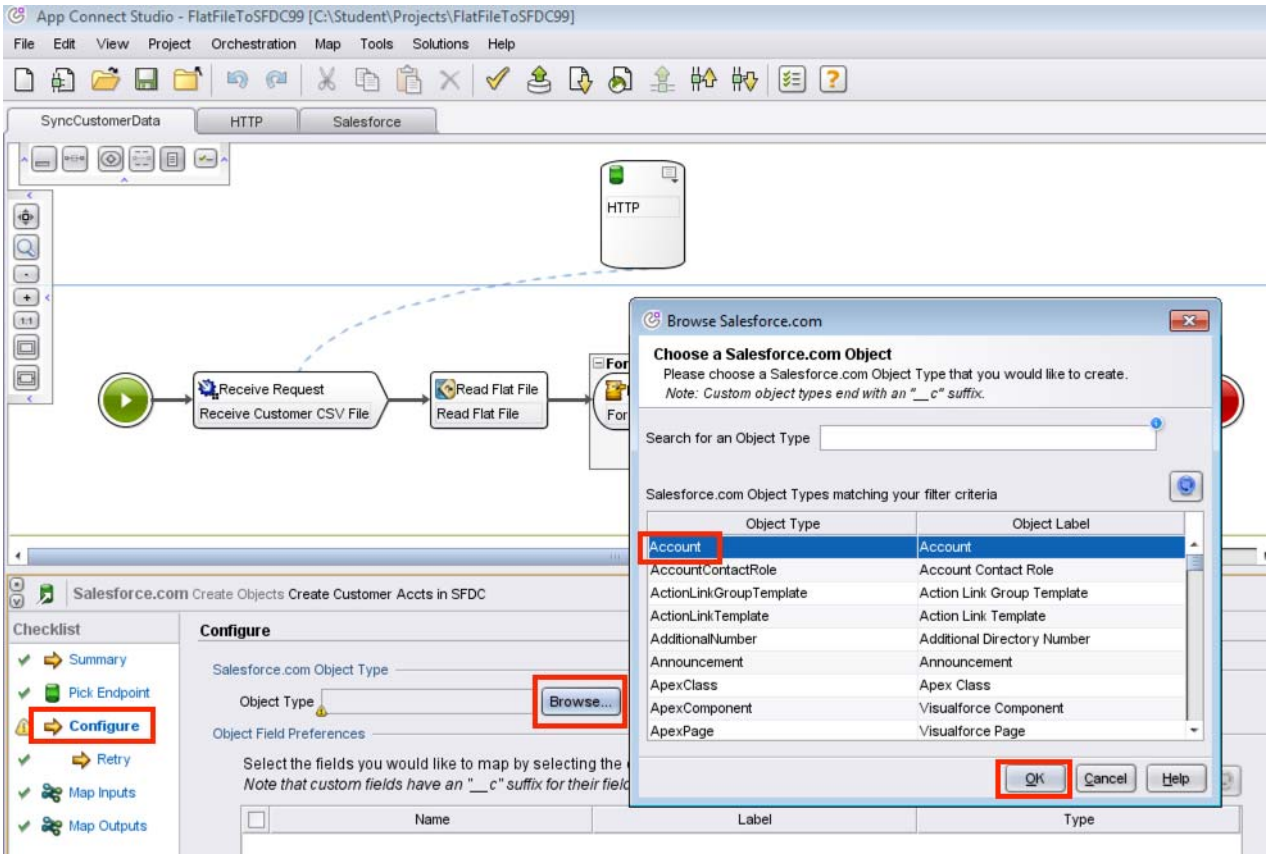
__ 21. **Summary**. Change the **Activity Name** to: `Create Customer Accts in SFDC`

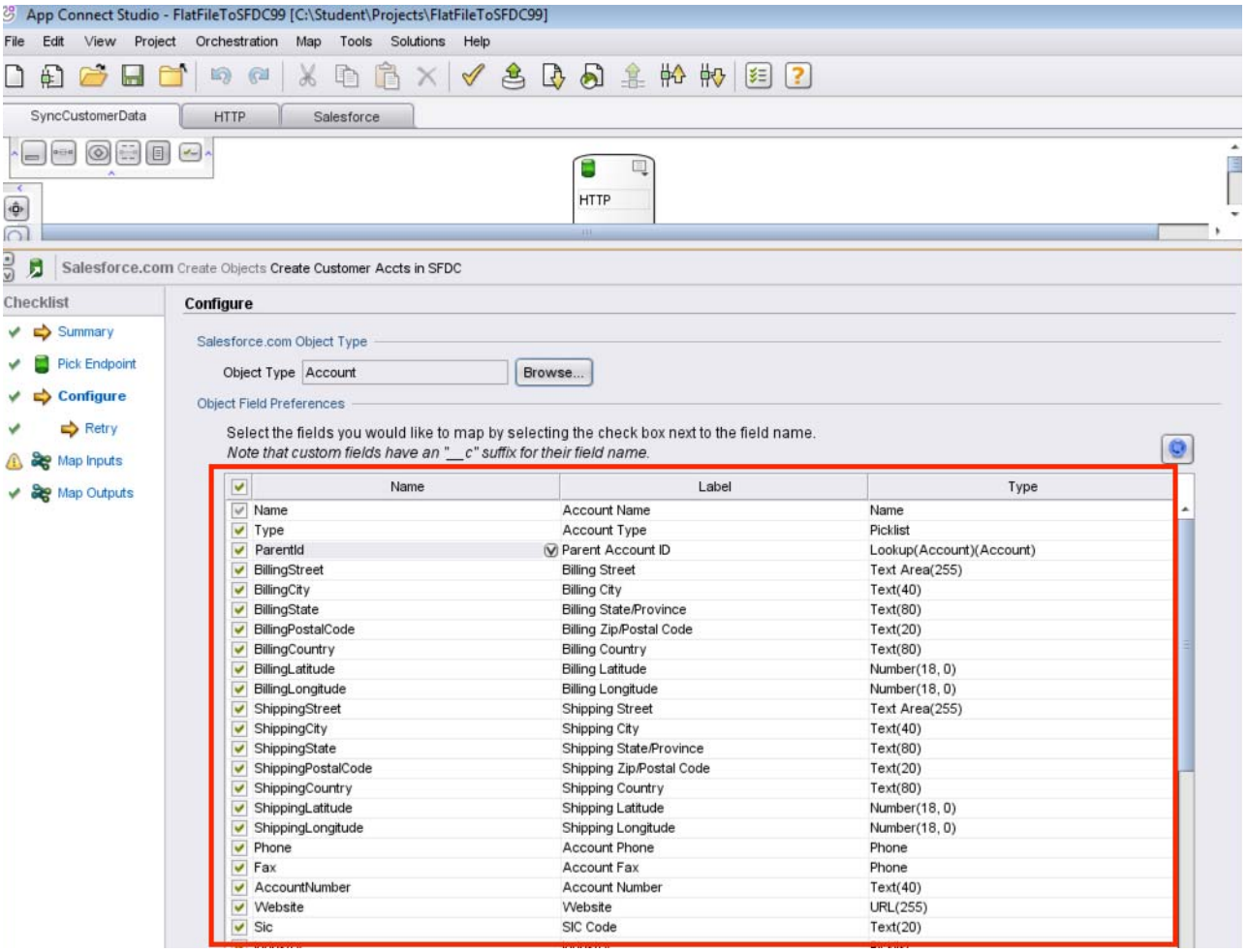


__ 22. Pick **Endpoint**. Click **Browse** and select **Salesforce** as the endpoint. Click **OK**.

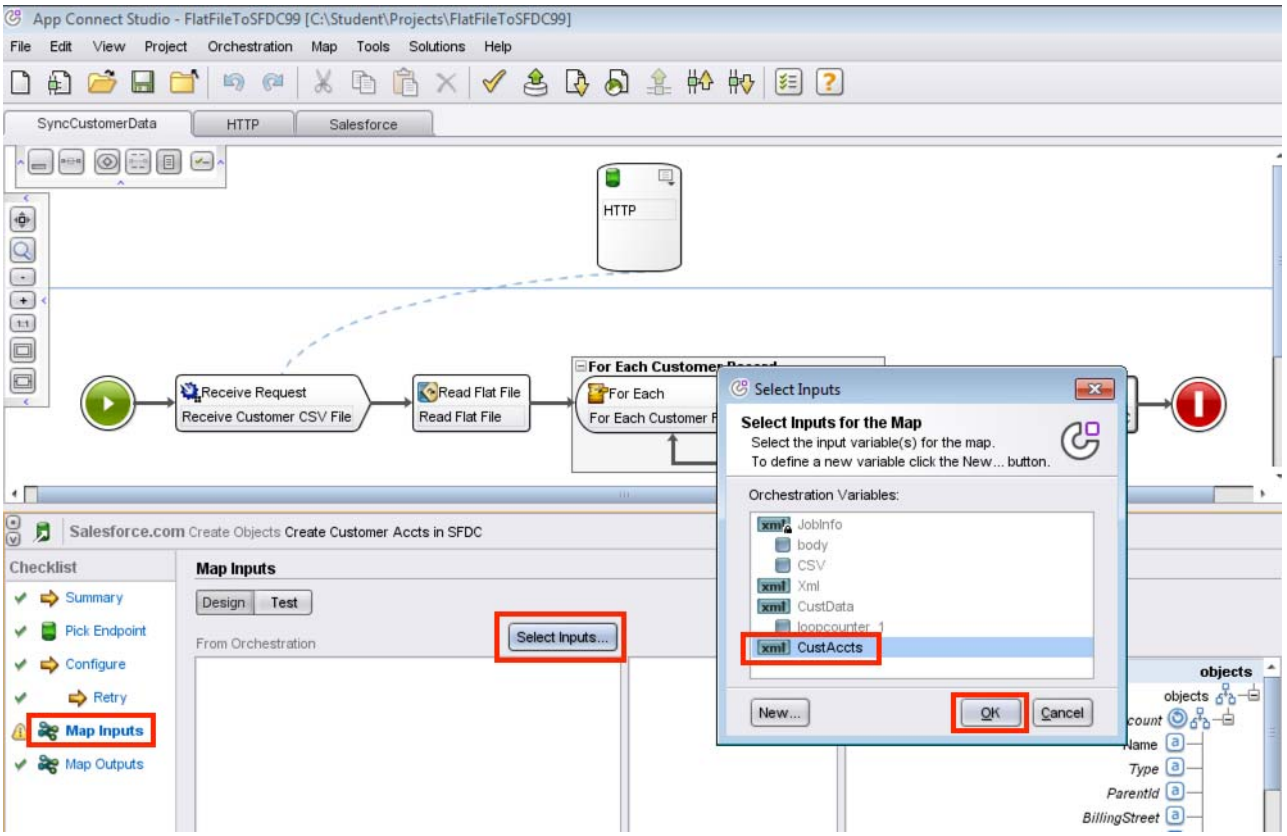


__ 23. **Configure**: Click **Browse**, select the **Account Object** from the **Salesforce** list, and then click **OK**. All of the available **Account Object** metadata fields are then displayed, including any customized fields that were added to the **Salesforce Account Object**.

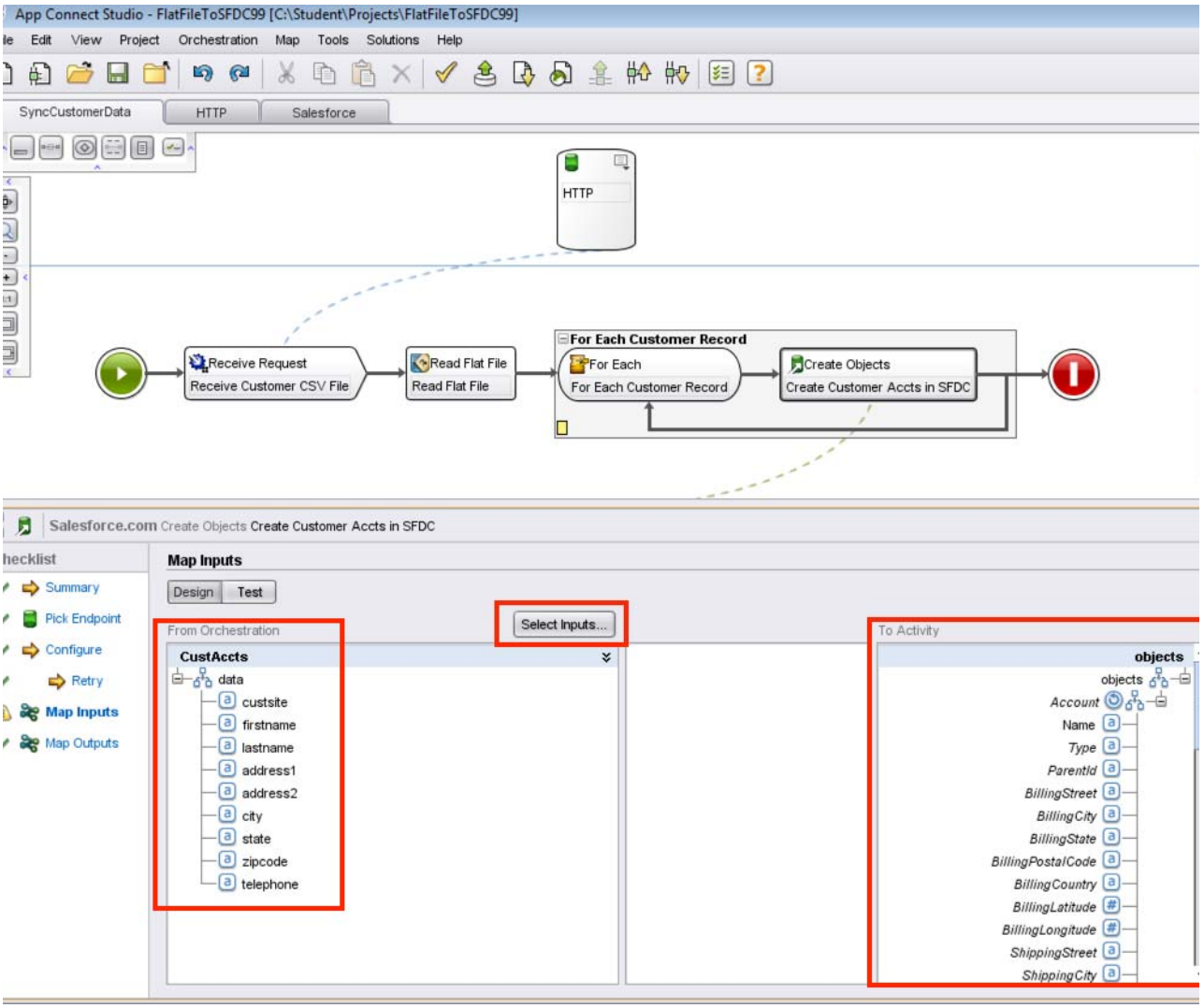




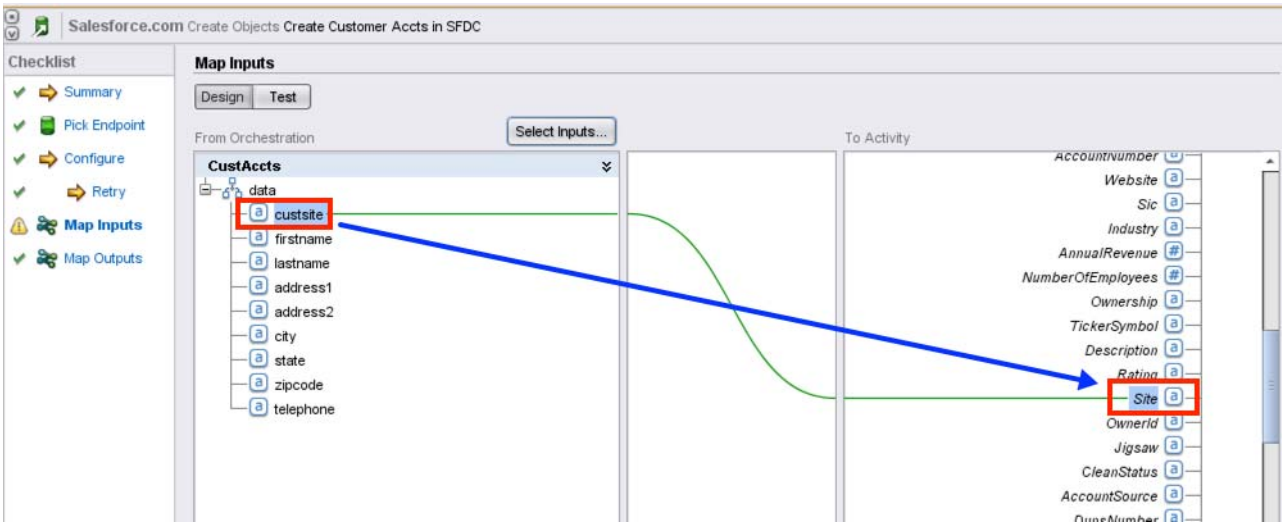
__ 24. **Map Inputs**: The input to this activity is the output from your previous activity, the **For Each** activity. Click **Select Inputs**, choose **CustAccts**, and then click **OK**.



The representation is then the source of data, which is located under the **From Orchestration** column on the left and the target of the data on the right under the **To Activity** column.

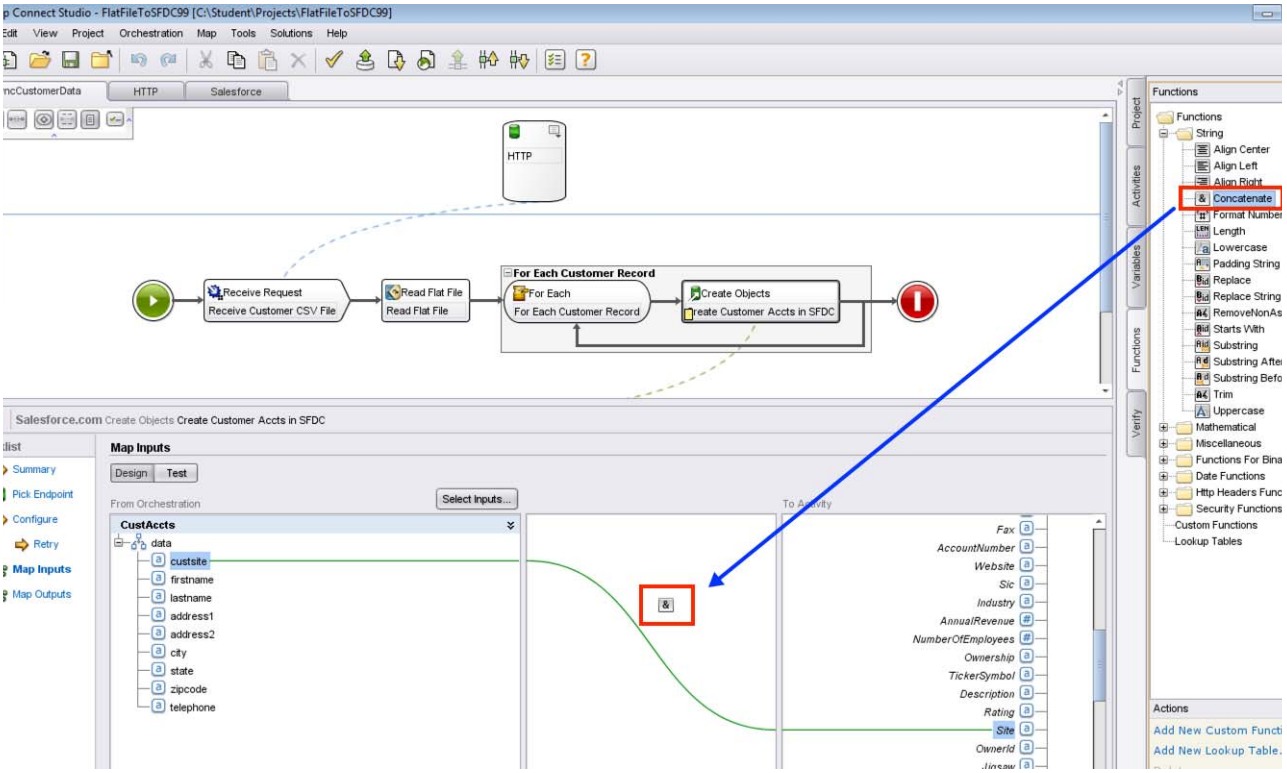


__ 25. Place your cursor on the **custsite** field that is located under the source of data column **From Orchestration**. Left-click and hold down, and then drag that field to the **Site** field that is located under the target of data column **To Activity**.



Notice that the source of data contains a **firstname** field and a **lastname** field. However, the target of data has only a **Name** field, which requires a manipulation of the data between the source and target systems.

__ 26. Next, select the **Functions** tab on the right side of App Connect Studio. You see the available functions that are used to transform data that is being sent from a source of data to a target of data. Select the **Concatenate** function from the String folder and drag it to the area between the **From Orchestration** column and the **To Activity** column.

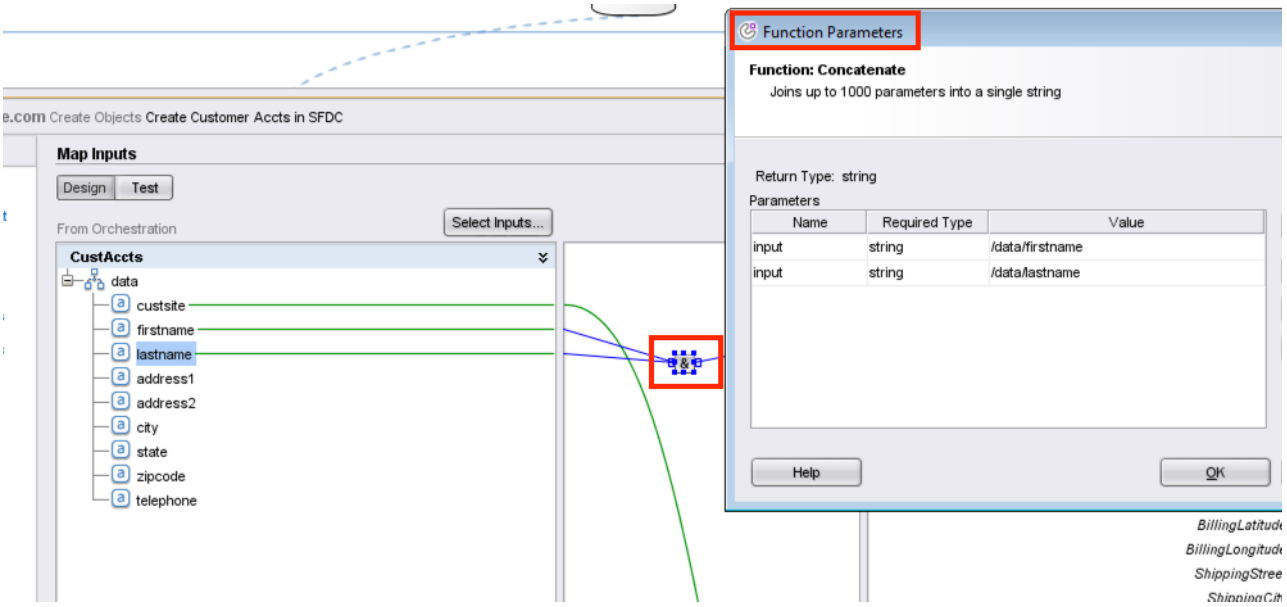


__ 27. Select the **firstname** field from the **From Orchestration** column and drag it to the Concatenate (**&**) icon.

__ 28. Select the **lastname** field from the **From Orchestration** column and drag it to the **Concatenate** icon.

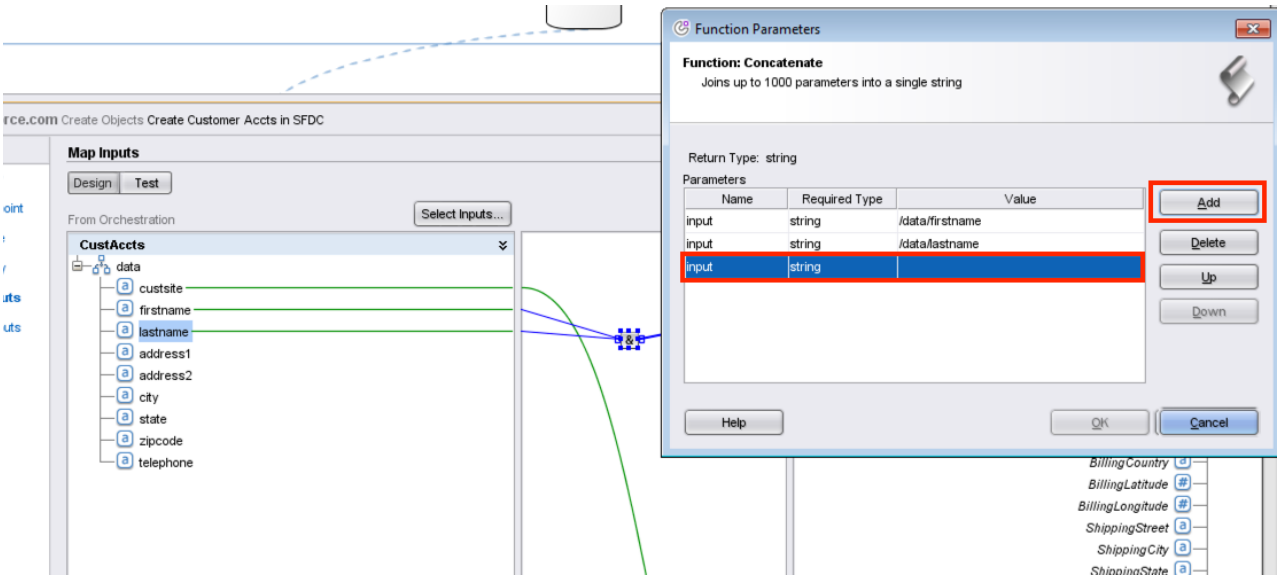
__ 29. Left-click and hold the **Concatenate** icon and drag it to the **Name** field located under the **To Activity** column.

__ 30. **Double-click** the **Concatenate** icon to open the **Function Parameters** dialog box.
*

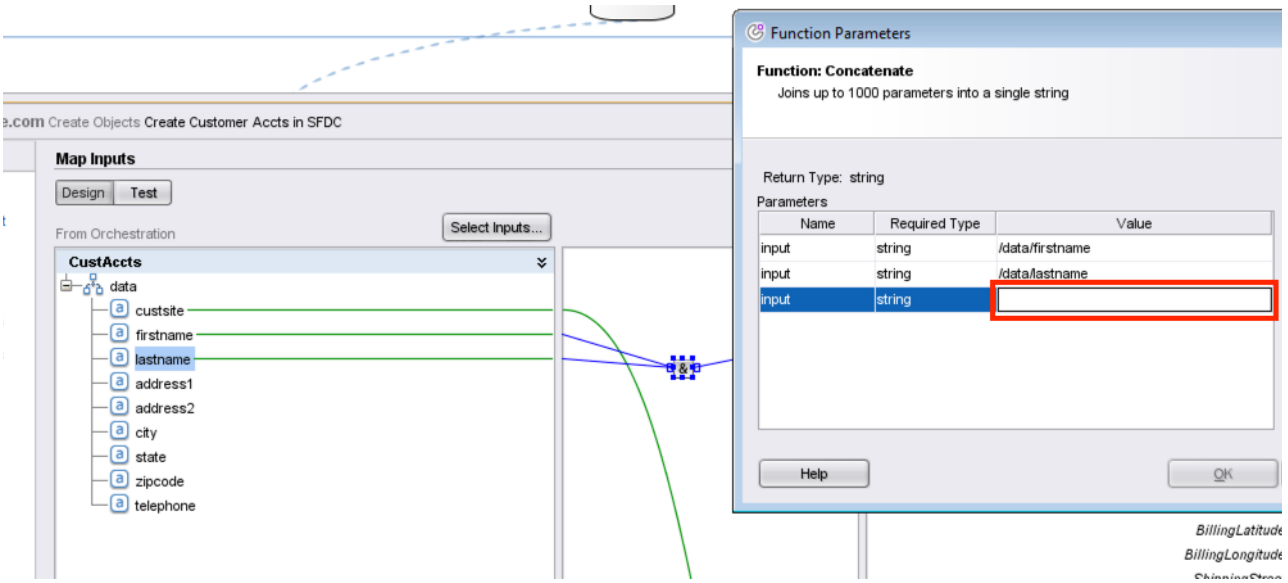


Now separate the **firstname** and **lastname** with a space so that they appear correctly in your target system.

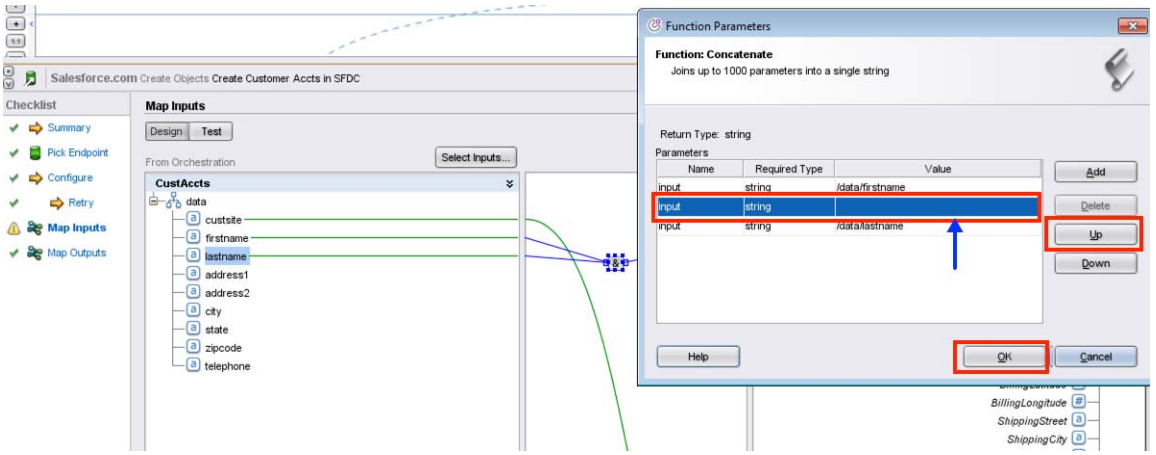
__ 31. Click **Add**, which then shows a new **input string** field.



__ 32. Double-click in the blank area under the **Value** column for the new input string and then press the space bar once to add a space and click the Enter key.

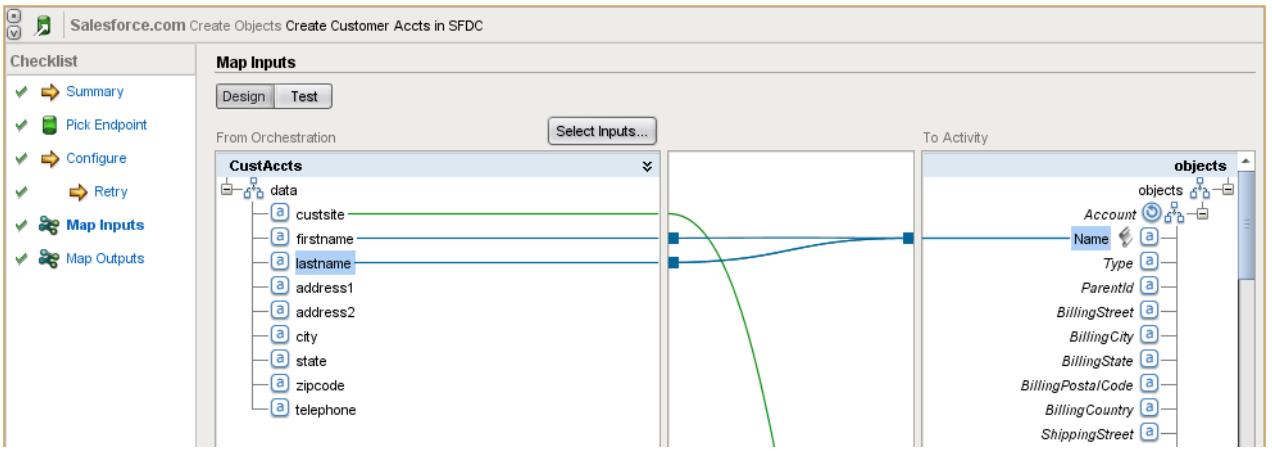


__ 33. Click **Up** so that you can move the space between the **firstname** and **lastname** fields. Click **OK**.

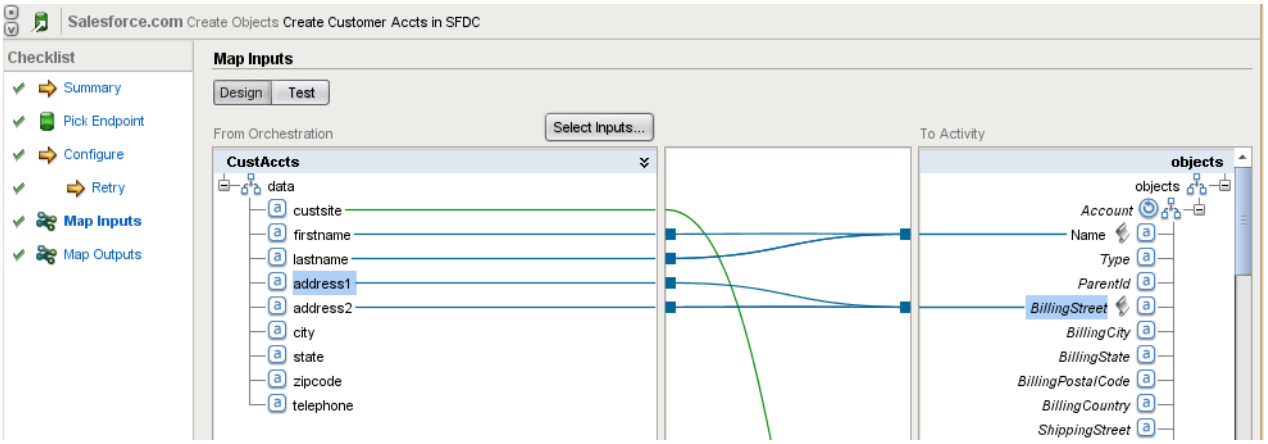


__ 34. The Function Parameters dialog box disappears. Now place your cursor anywhere in the area below the **Concatenate** icon and right-click to apply the **Function Graph** to the activity. Click **Apply Function Graph**.

The result is as follows:

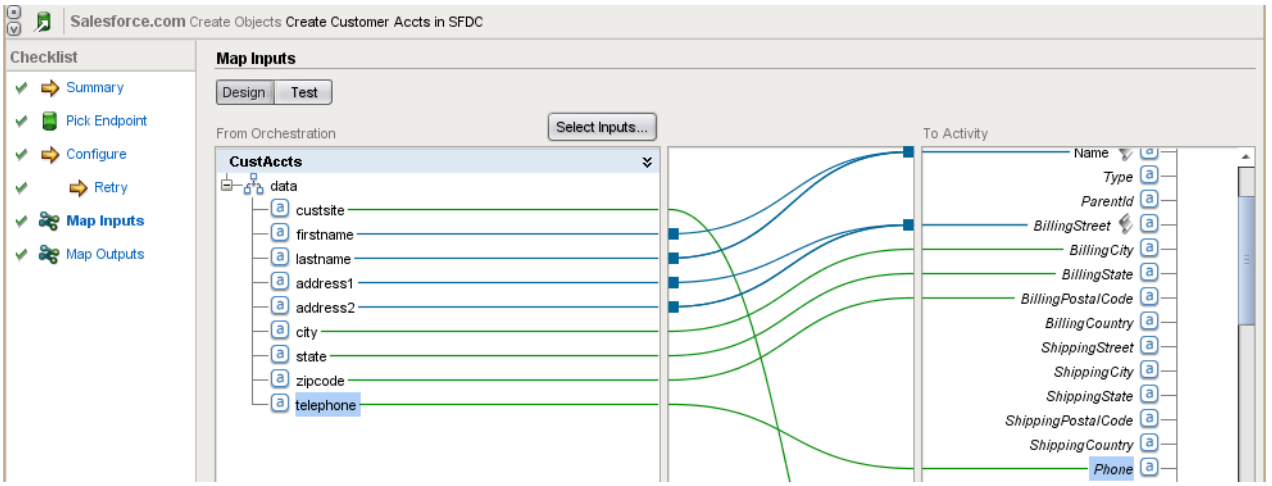


__ 35. Repeat the previous process to concatenate the **address1** and **address2** fields that are located under the source **From Orchestration** column to the **Billing Street** field, which is located under the target **To Activity** column. The result should look as follows:



__ 36. Now map the remaining fields under the source **From Orchestration** column to the corresponding columns that are located under the target **To Activity** column, as follows:

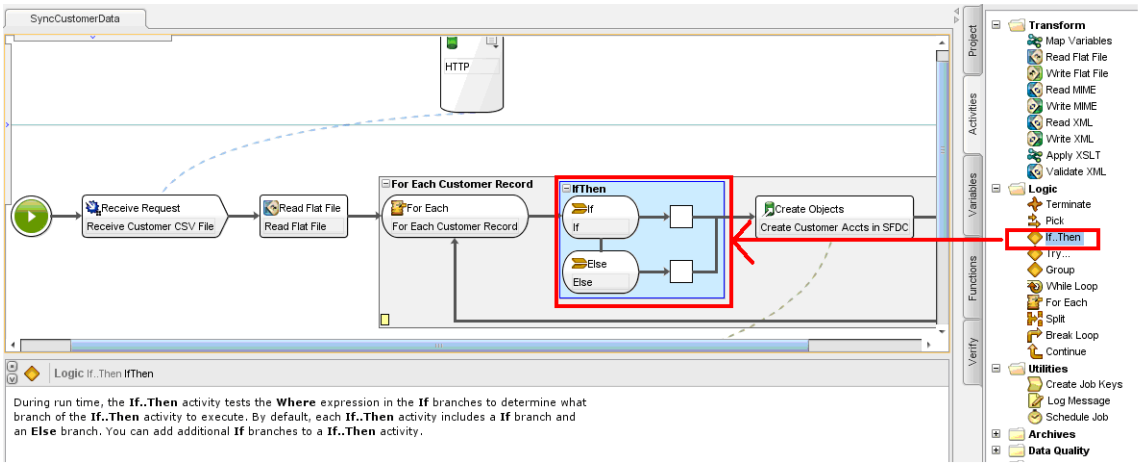
- city – Billing City
- state – Billing State
- zipcode – Billing Postal Code
- telephone – Phone



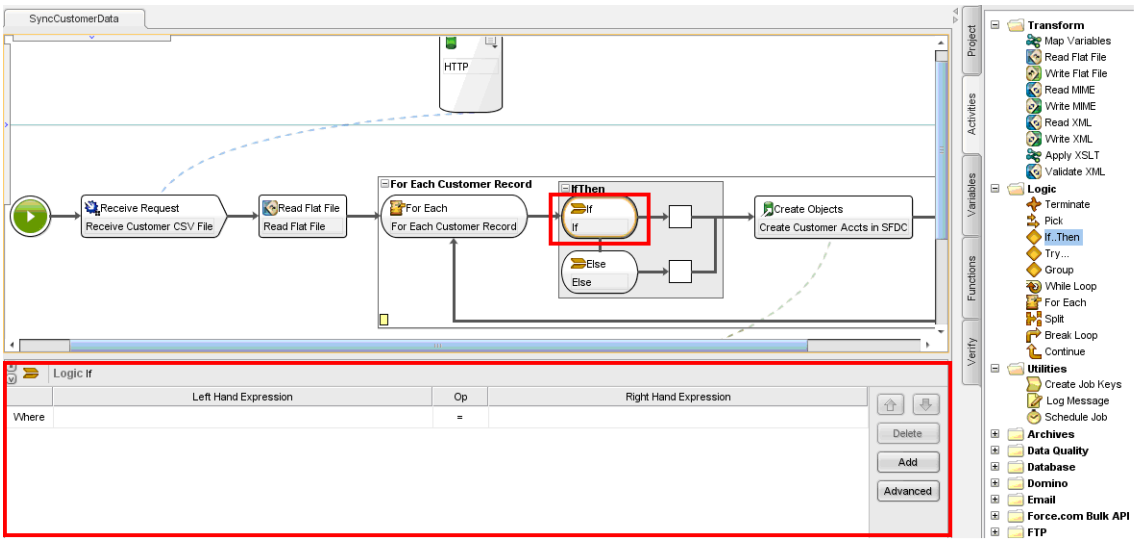
The configuration for the Salesforce.com Create Objects Activity is complete.

Add some business logic to the orchestration

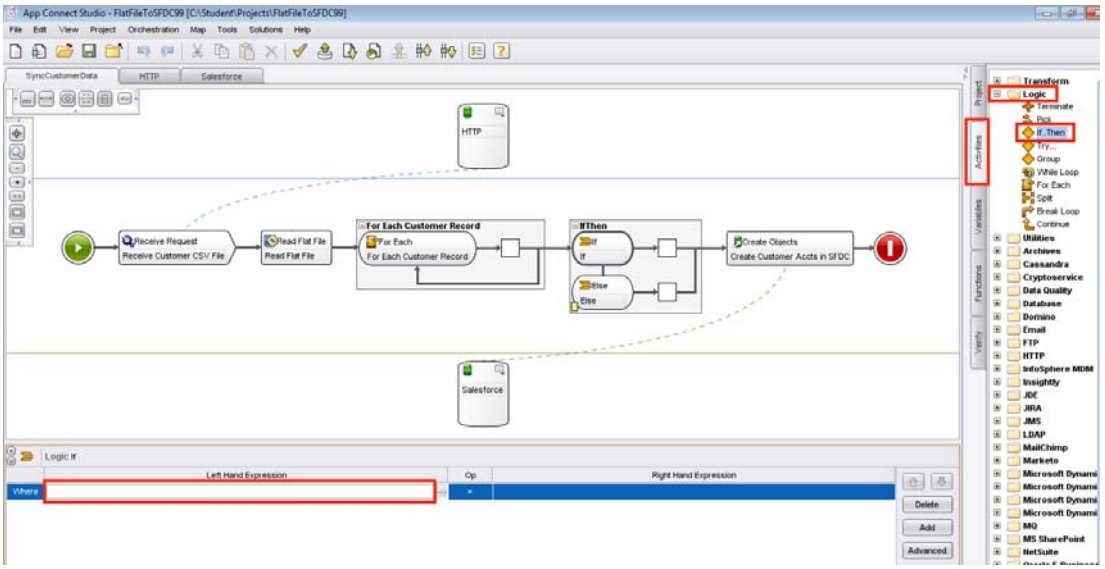
__ 37. Drag the **If Then Else Activity** on the **Activities** tab under the **Logic** folder to the area between the **For Each** activity and the Salesforce **Create Objects** activity.



__ 38. Click the top **If** input box, and notice that it opens the **If-Then** area below the orchestration workflow.



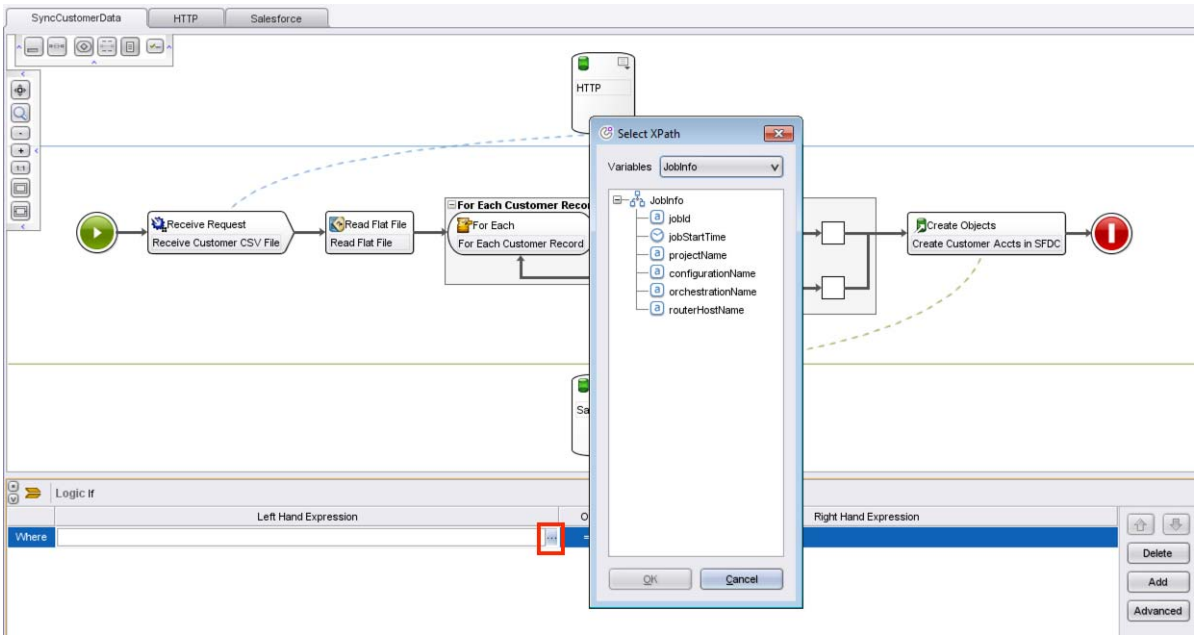
__ 39. Place your cursor in the blank area next to the **Where** parameter and click to open the dialog box under the **Left Hand Expression**.



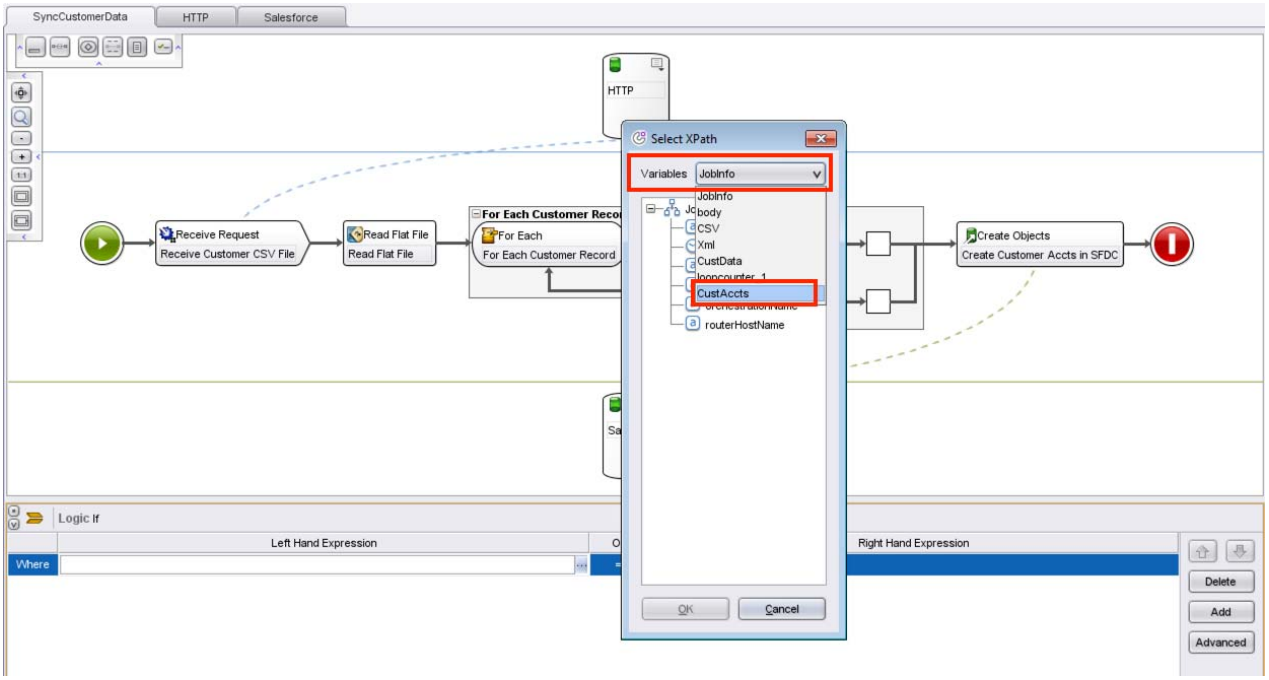
You can type an expression, or you can use the Logic Wizard. For this lab, you use the **Logic Wizard** to create the business logic expression.

__ 40. If the Customer Site is not equal to the value `East`, then create the new record in Salesforce.com. If it is equal to the value `East`, then disregard and log an error message to be viewed later in the lab.

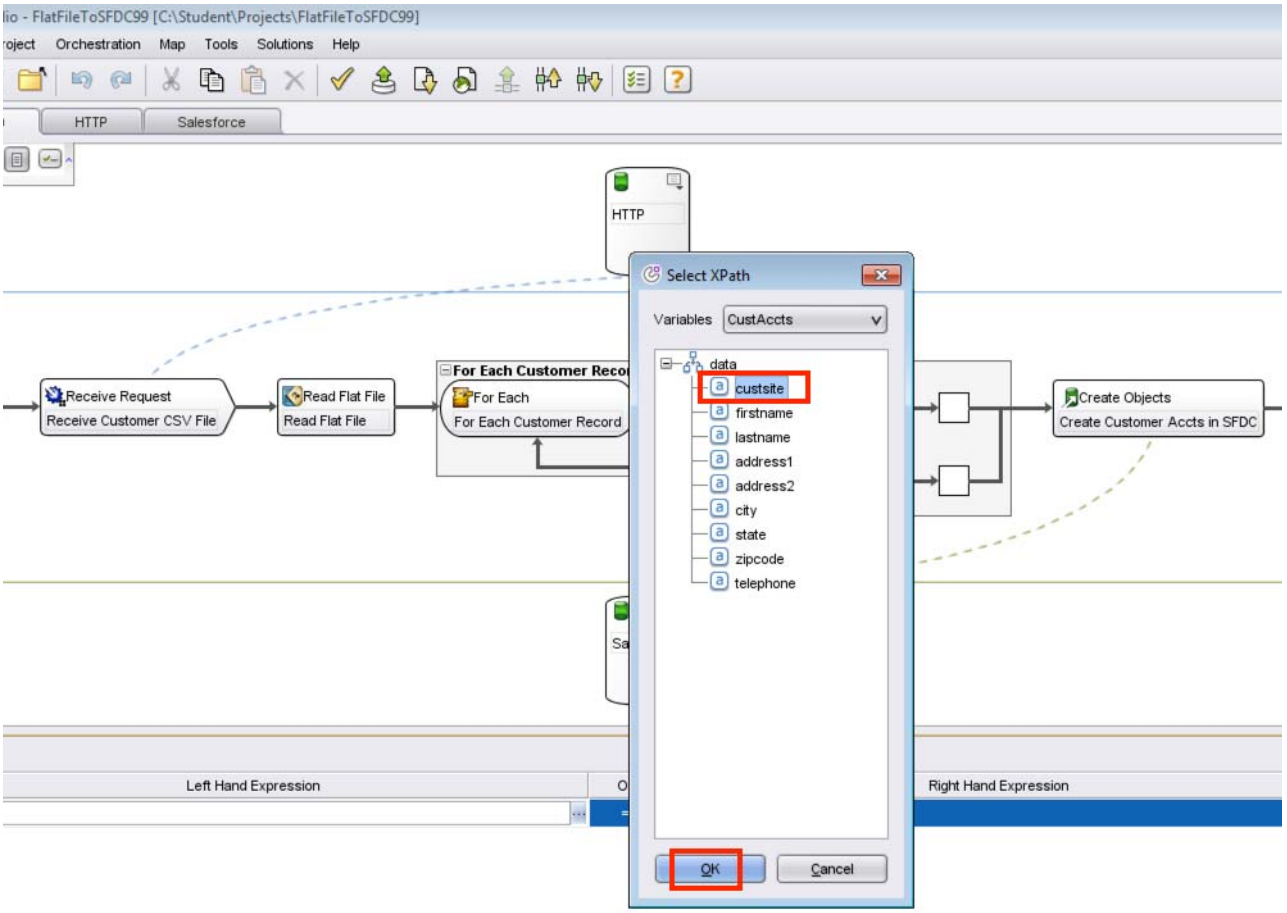
__ 41. Click the **...** box next to the blank **Left Hand Expression** field to invoke the Logic Wizard.

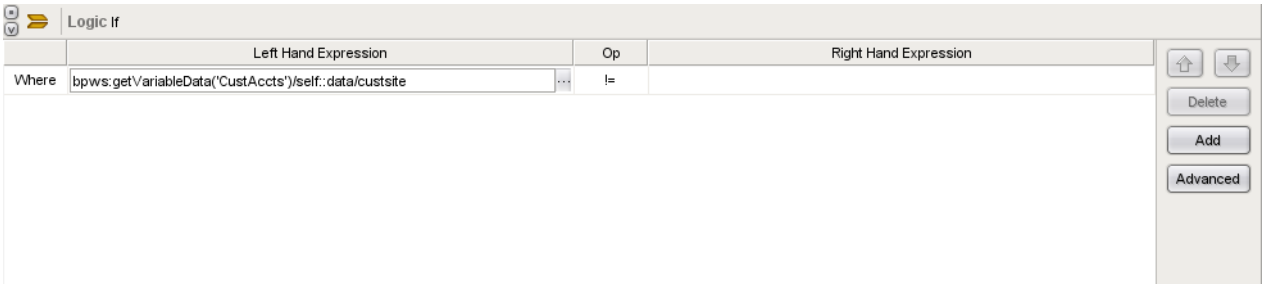


__ 42. Click the **Variables** list to select the variable **CustAccts** that is to be used as input. As soon as you select **CustAccts**, you see a list of the fields available for the business logic.

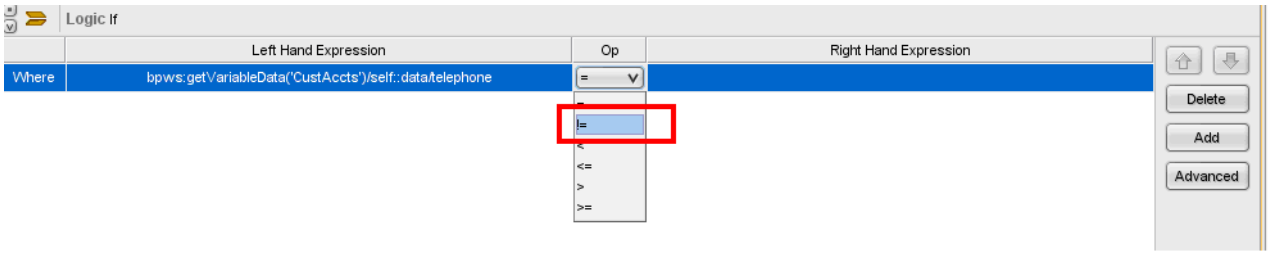


__ 43. Click the variable **custsite** and then **OK**.

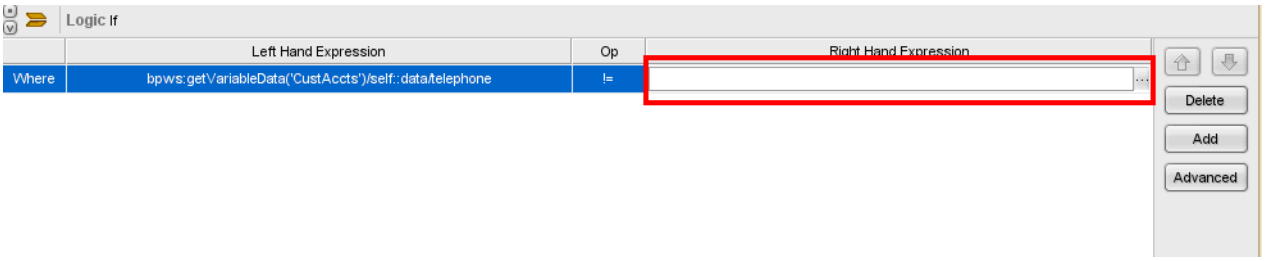




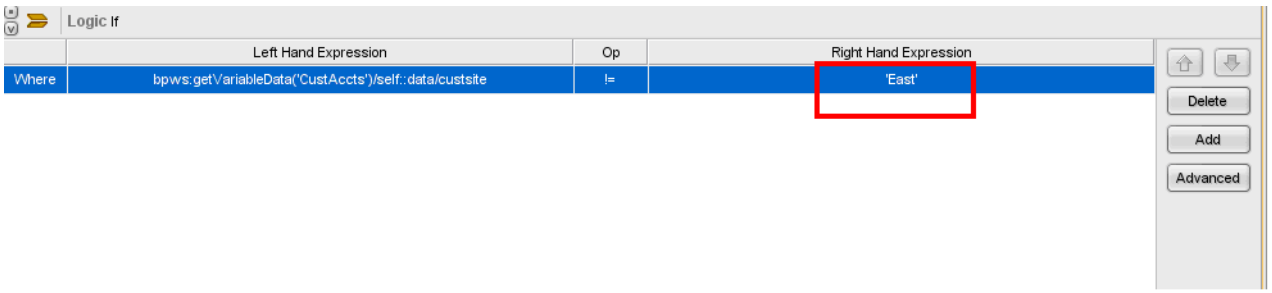
__ 44. Click the **operator** field that is located under the **Op** column to select the “Not Equal” (`!=`) parameter.



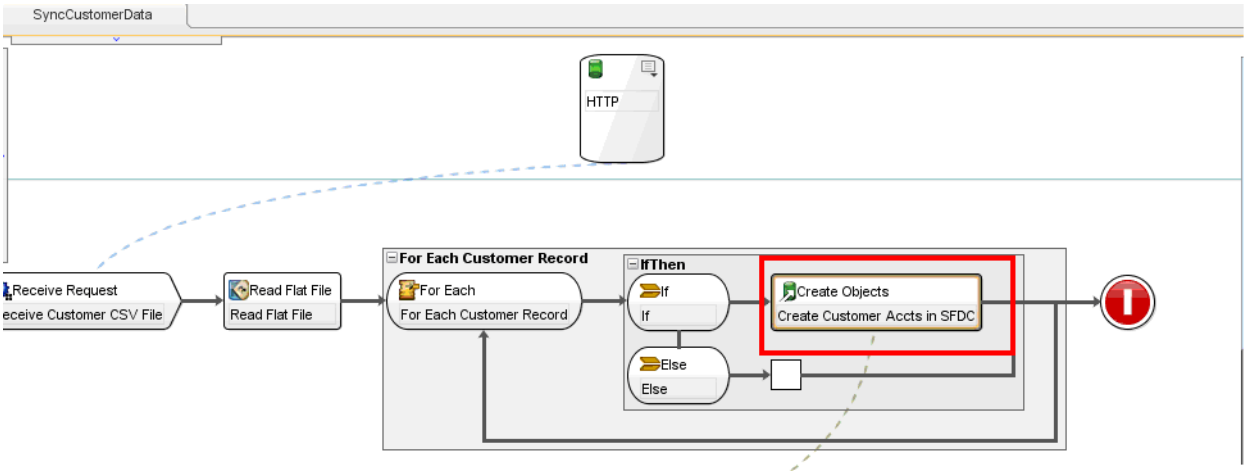
__ 45. Now click the blank area under the **Right Hand Expression** column to add the value.



__ 46. Enter the value as typed here and press the Enter key: `'East'` (make sure to use single quotation marks).

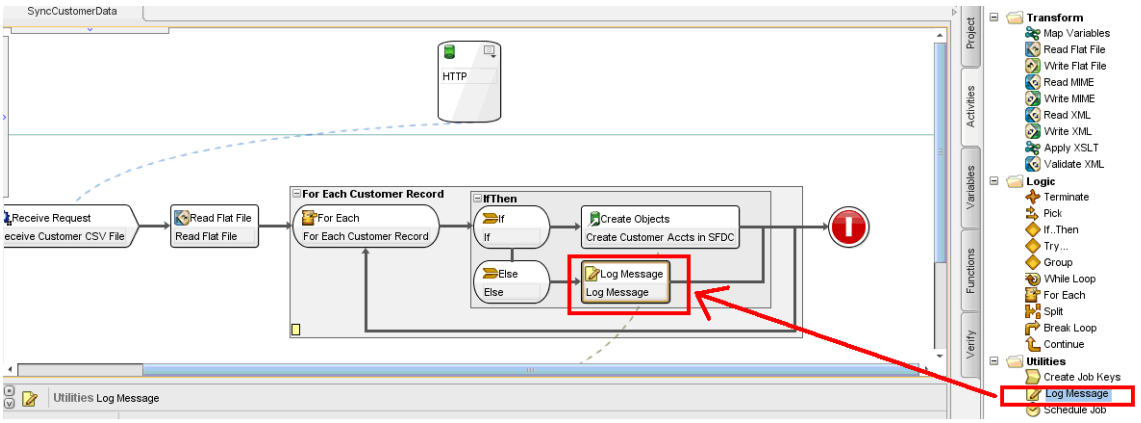


__ 47. Next, you click the Salesforce.com **Create Objects** activity and drag it to the empty box to the right of the **If** box. In this way, any records that pass the business logic test are created in Salesforce.com.

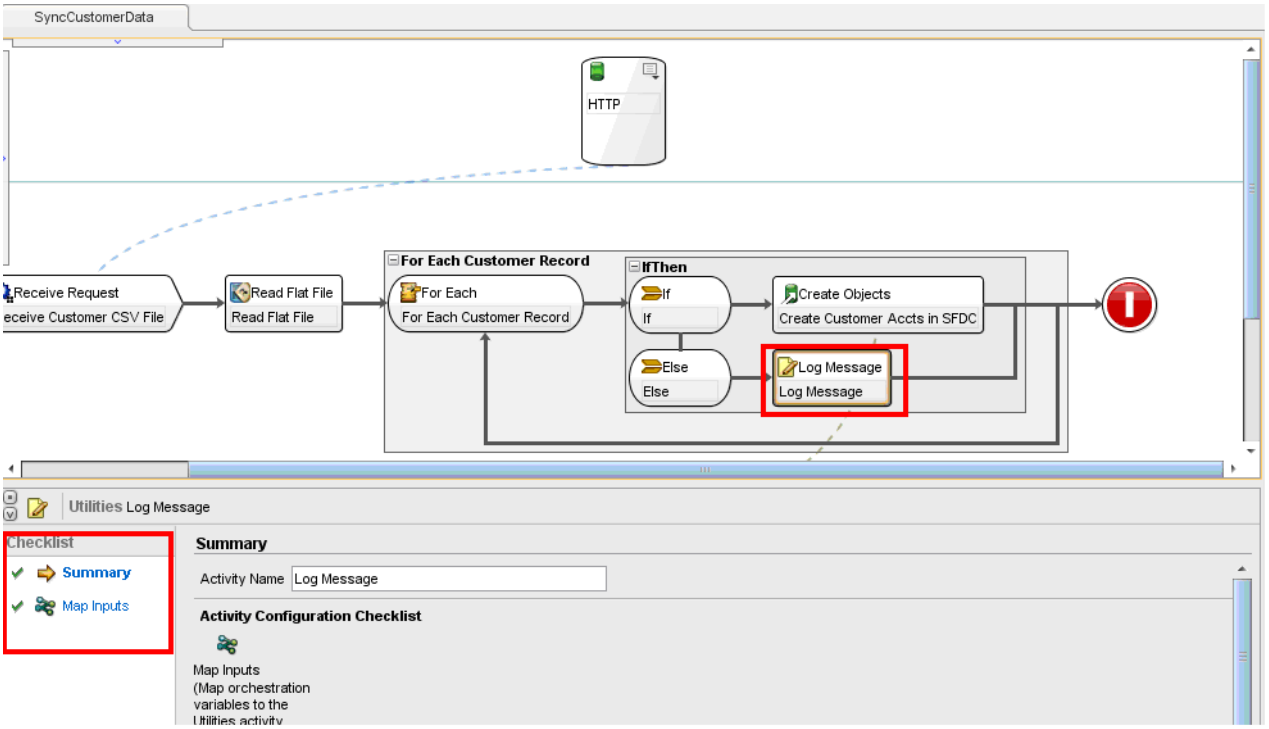


Create an error message for those records that do not meet the business logic validation

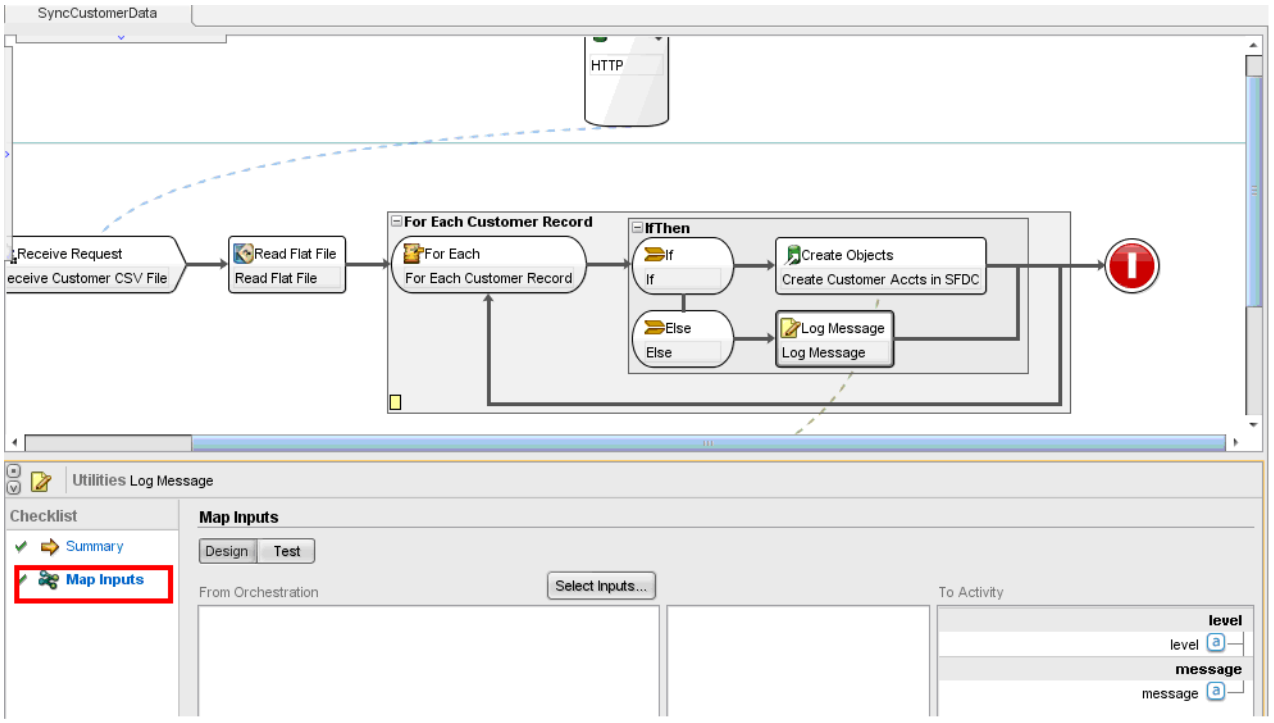
__ 48. From the **Activities** tab **Utilities** folder, select the **Log Message** activity and drag it to the blank box next to the Else box.



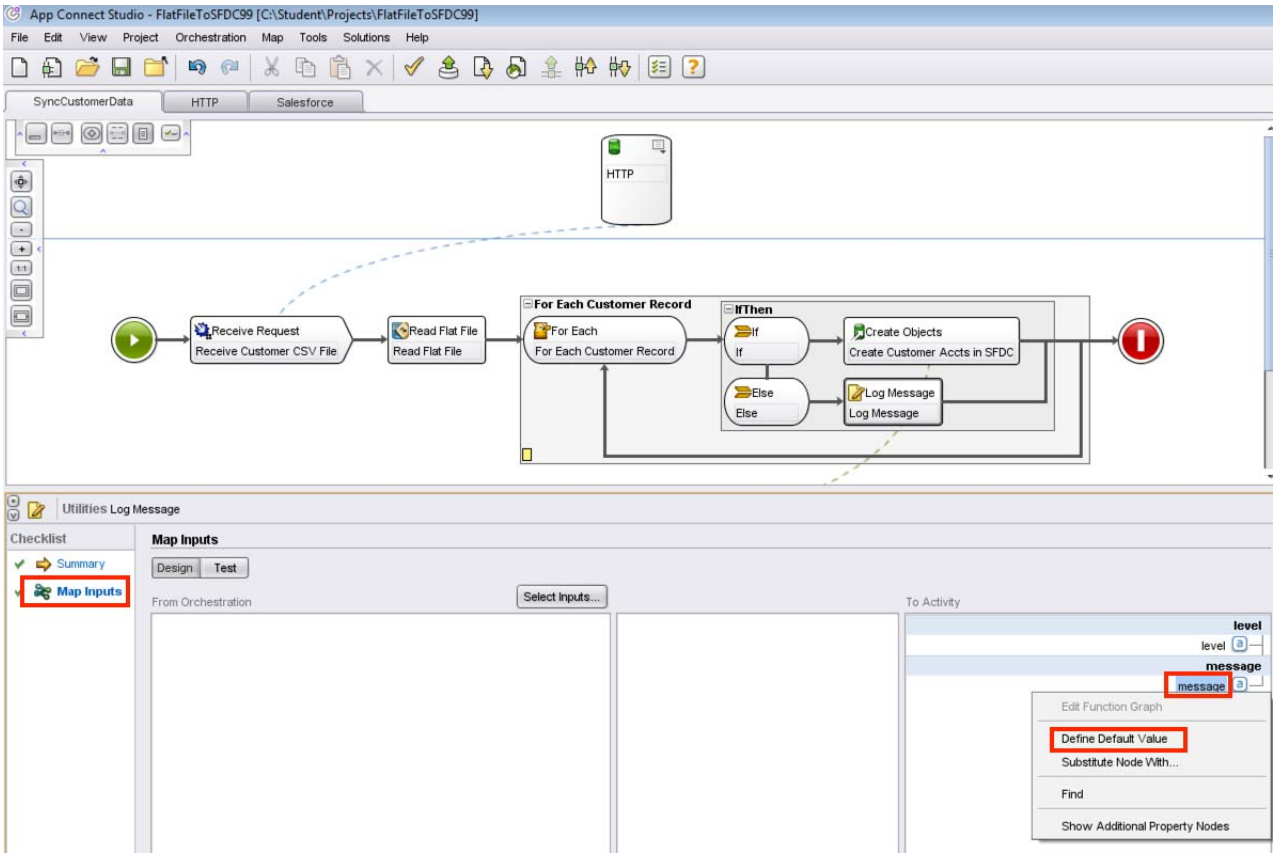
__ 49. Click the **Log Message** activity to display the **Checklist**.



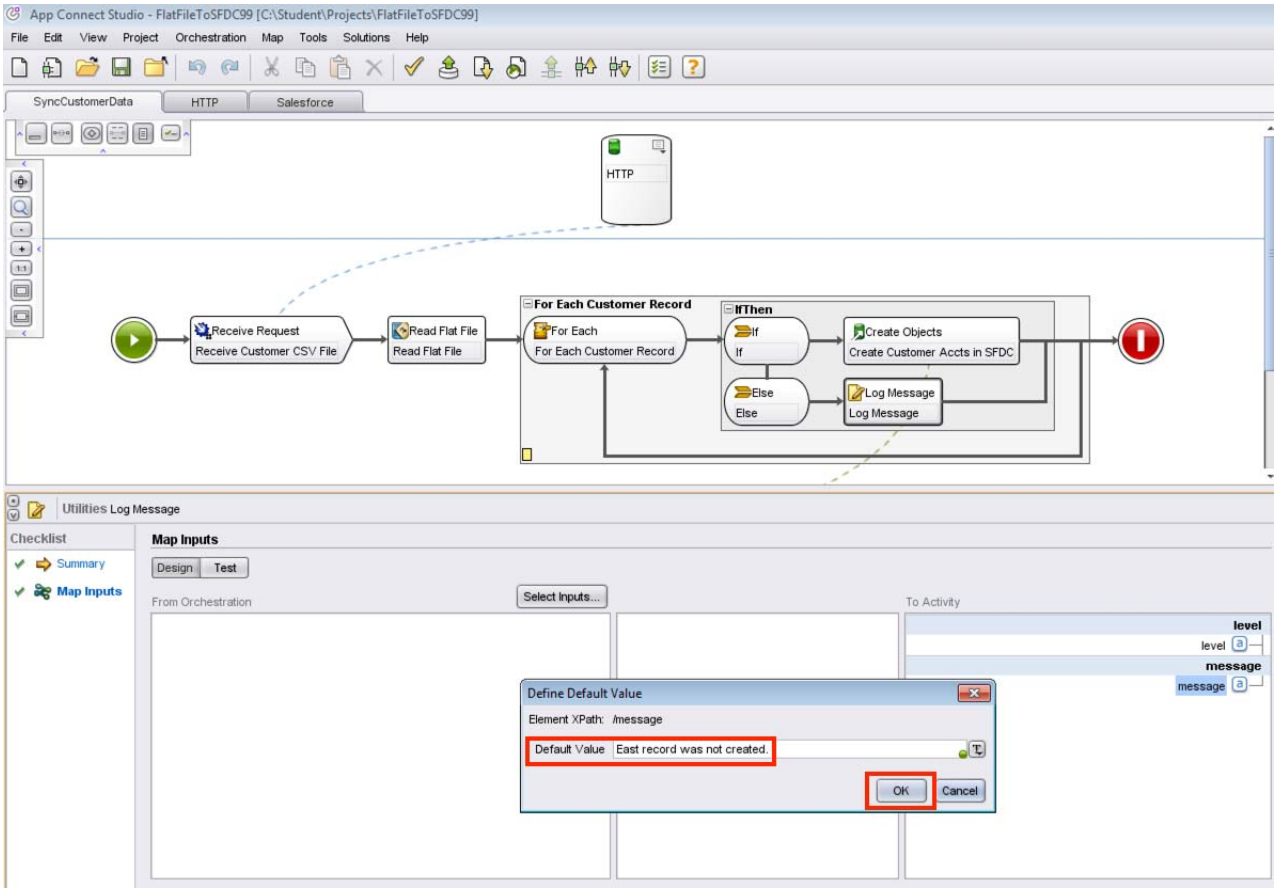
__ 50. Click the **Checklist** item **Map Inputs**.



__ 51. Right-click the message field to assign a default message.



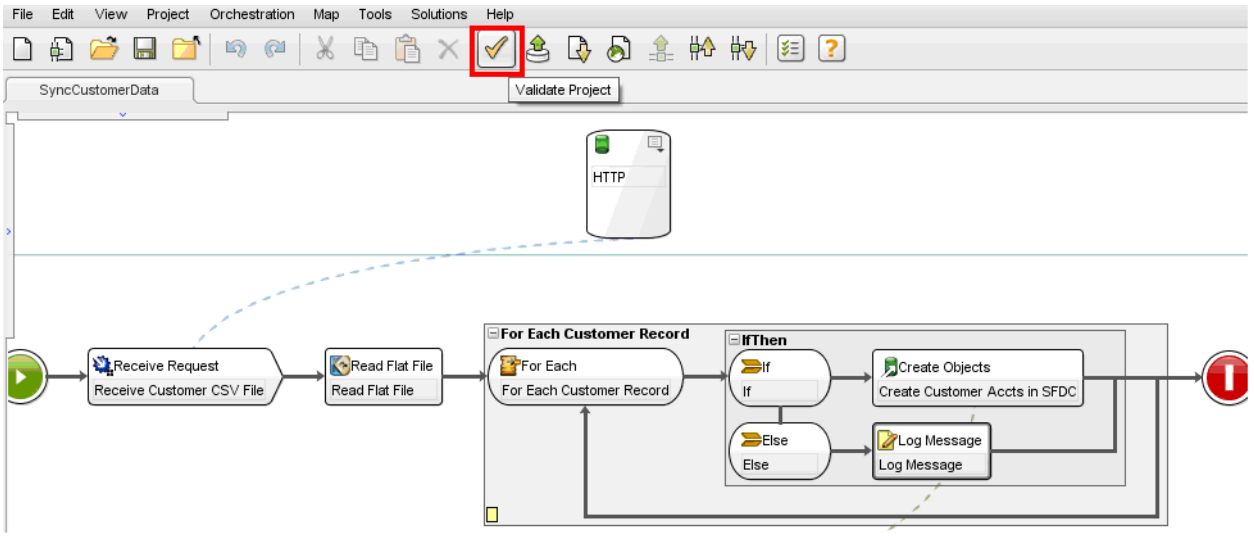
__ 52. Type in the following text: `East record was not created.`
Click **OK**.



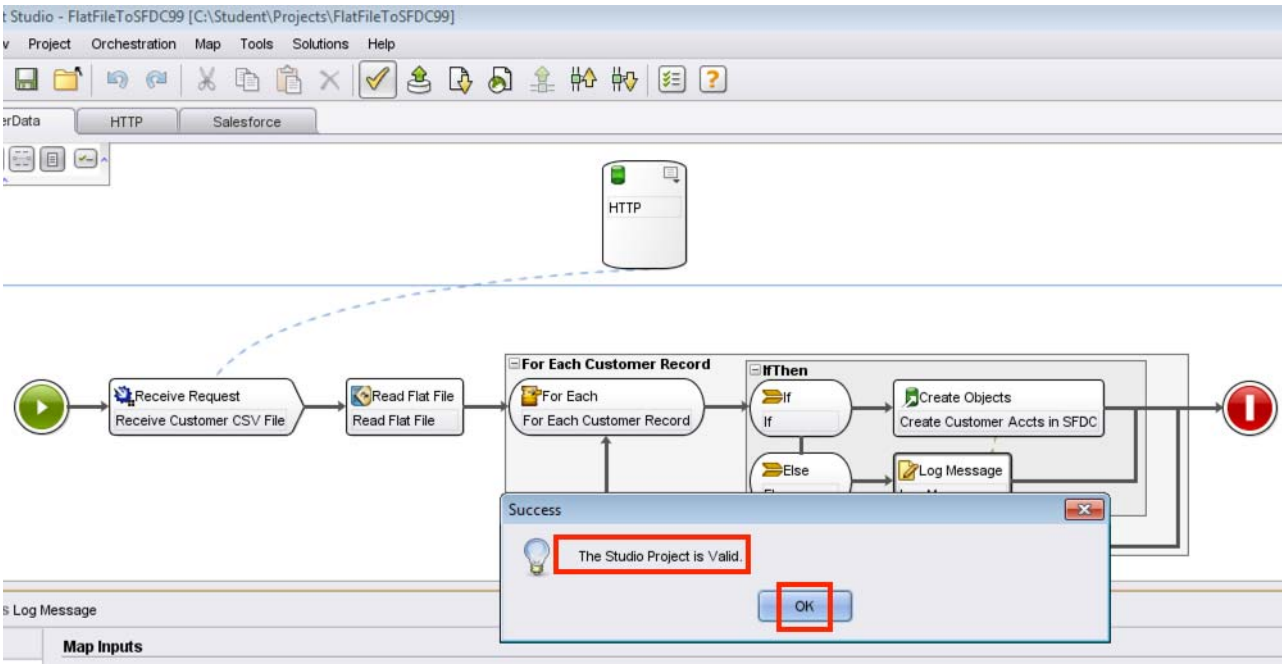
Notice that a **D** is now placed next to the message field, which indicates that a default value was created.

Validate your orchestration to make sure that all of your checklist items are answered correctly

__ 53. Click the **Validate Project** icon at the top of App Connect Studio.

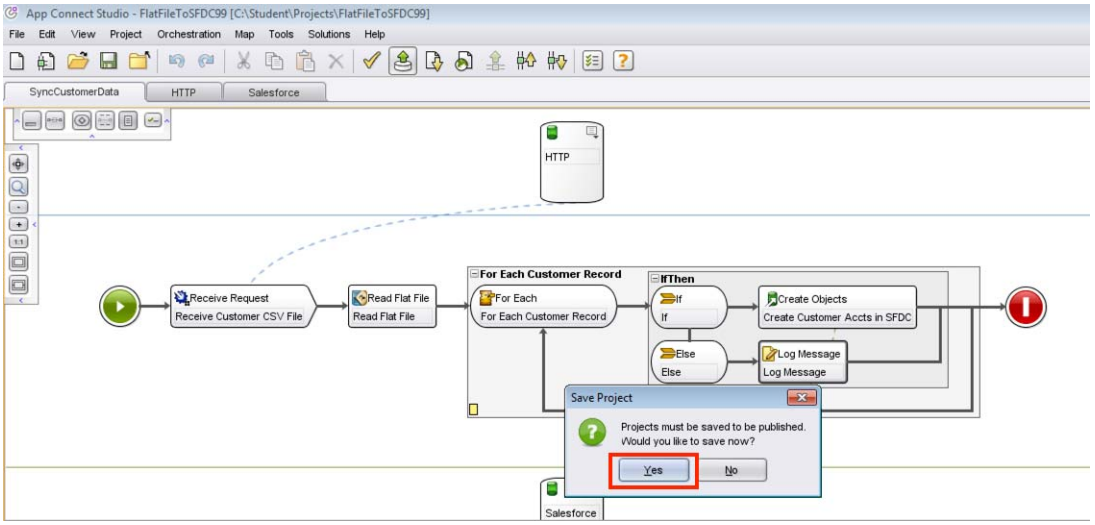


If the orchestration validates correctly, you should see a **Success** message box. Click **OK**.

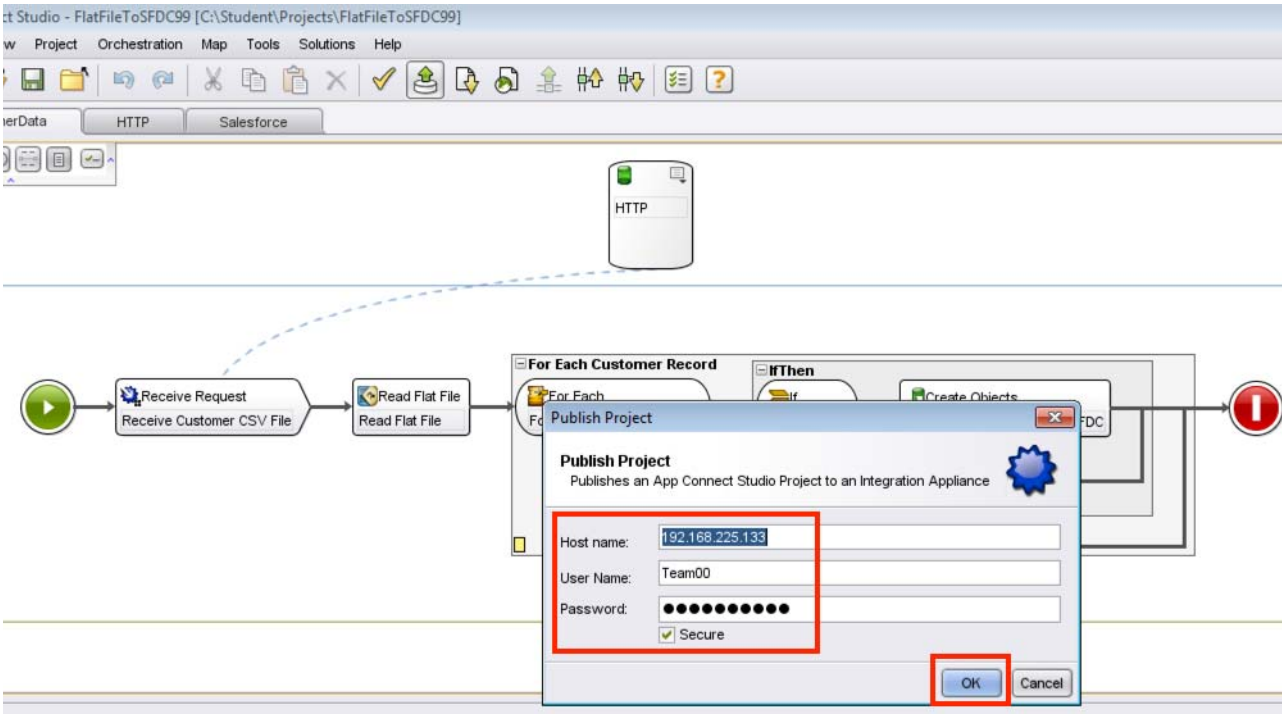


__ 54. To publish the project, click the **Publish** icon on the toolbar.

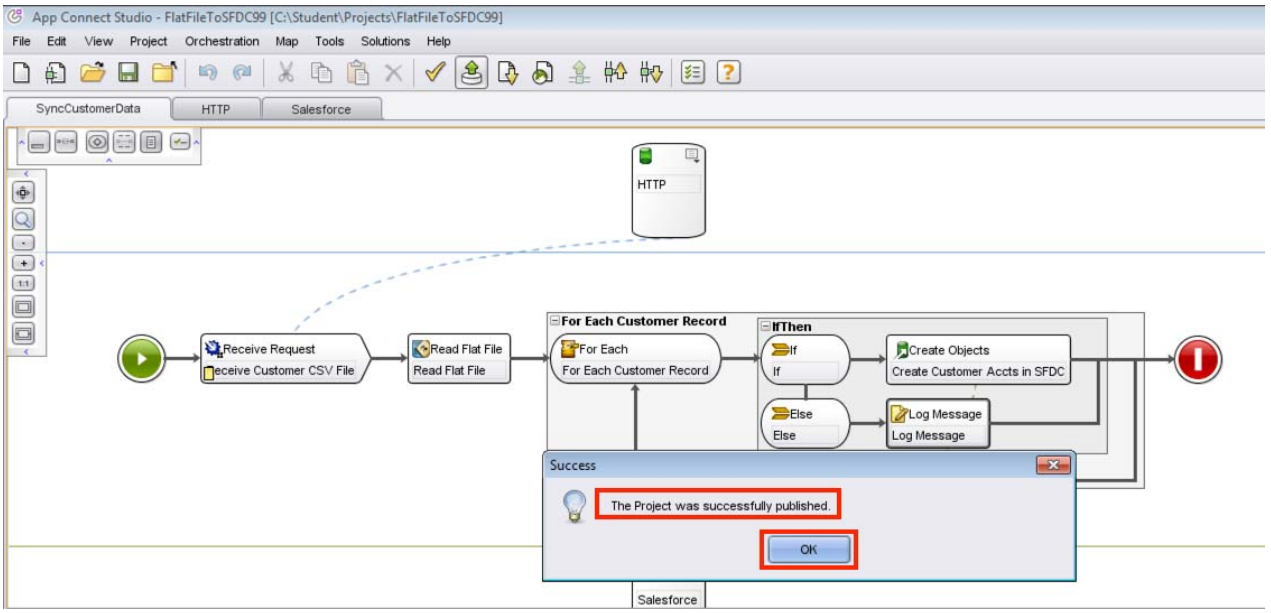
__ 55. Select **Yes** to publish the project.



__ 56. Complete the fields with the correct App Connect Professional Management IP for **Host name**, **User Name**, and **Password** parameters and click **OK** (see the Connection Parameters spreadsheet).

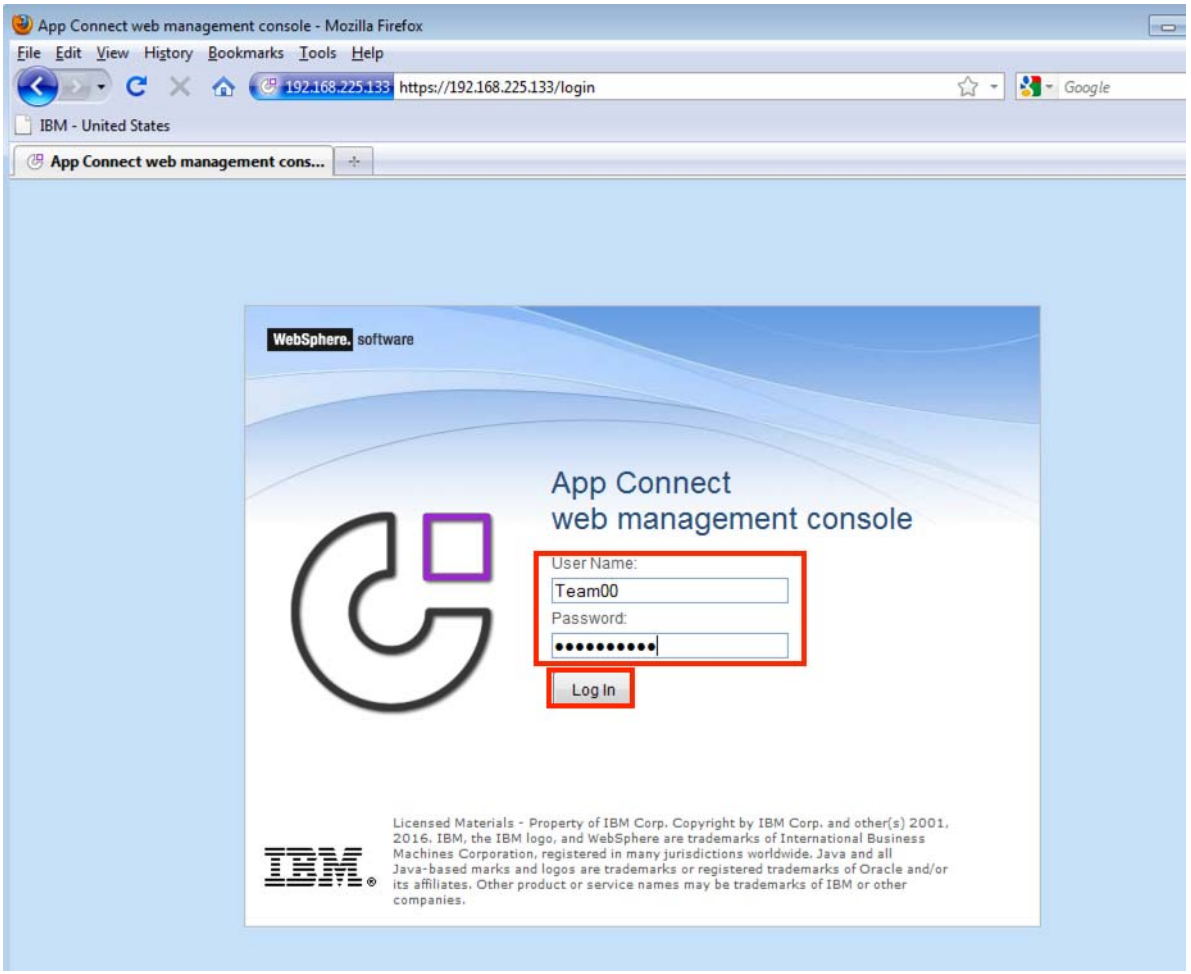


__ 57. If the project publishes correctly, you should see the Success message. Click **OK**.

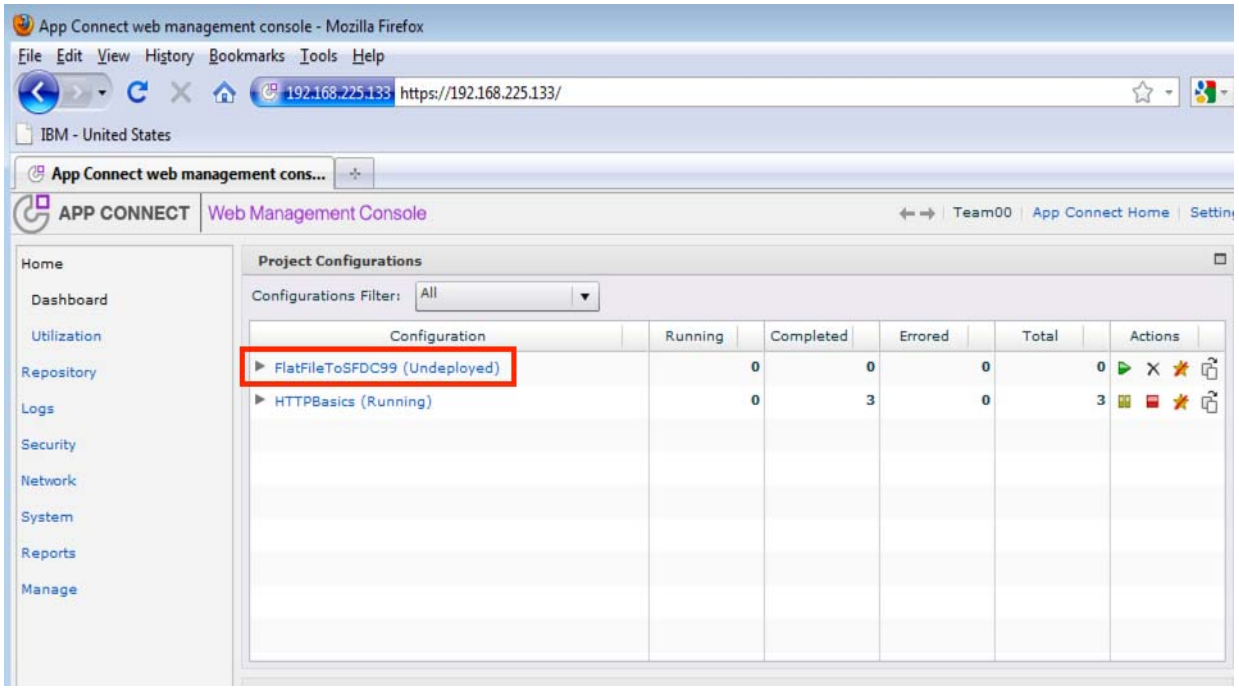


Deploy the project on the IBM App Connect Professional Web Management Console (WMC)

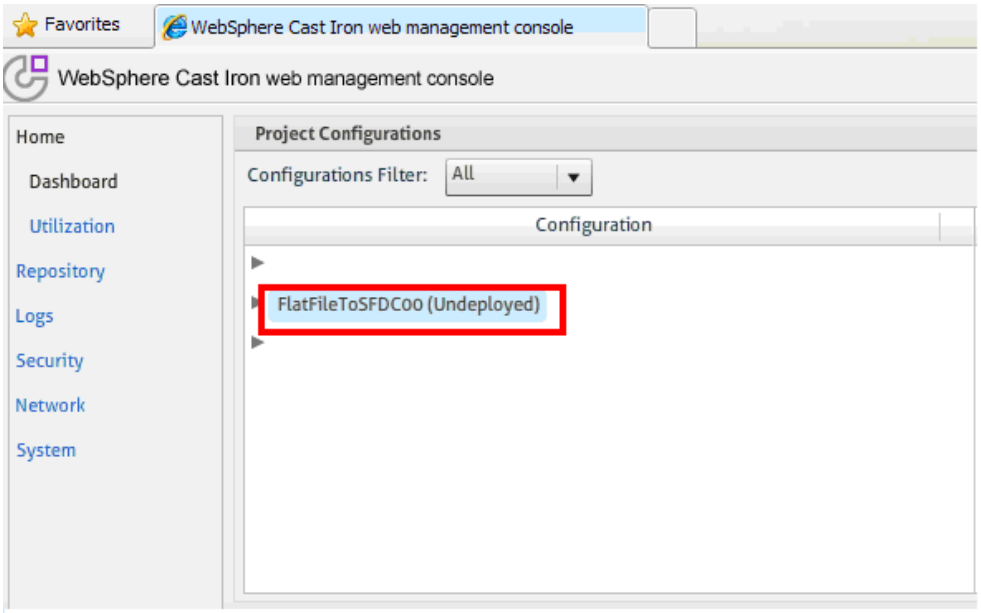
__ 58. Log in to the **WMC** by using the URL and login credentials, which the instructor provides (use WMC Admin ID and password). Click **Log In**.

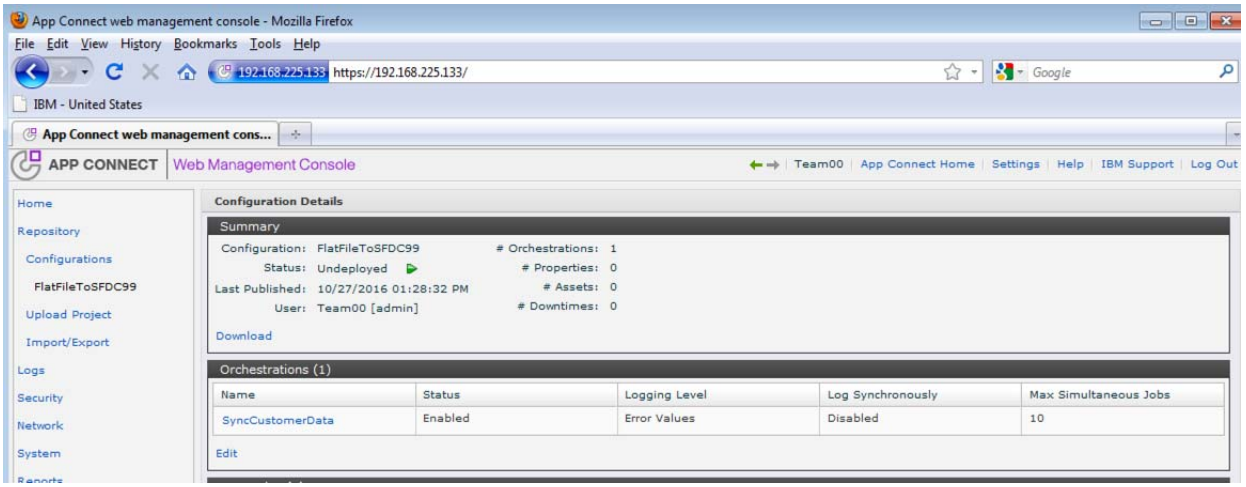


You should see that your published project is listed on the WMC dashboard.

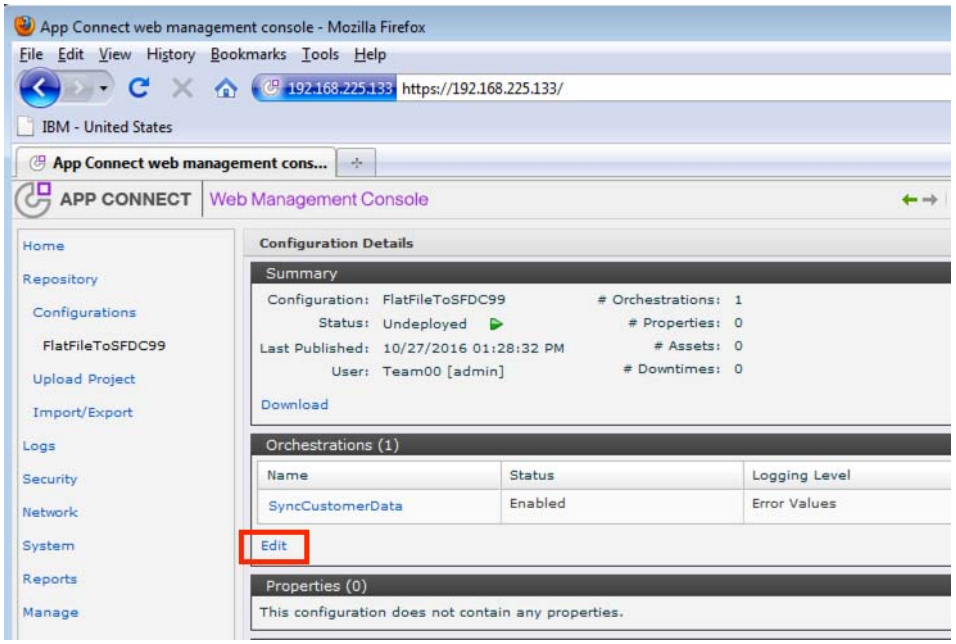


__ 59. Place the cursor on your project name and click to view the details.

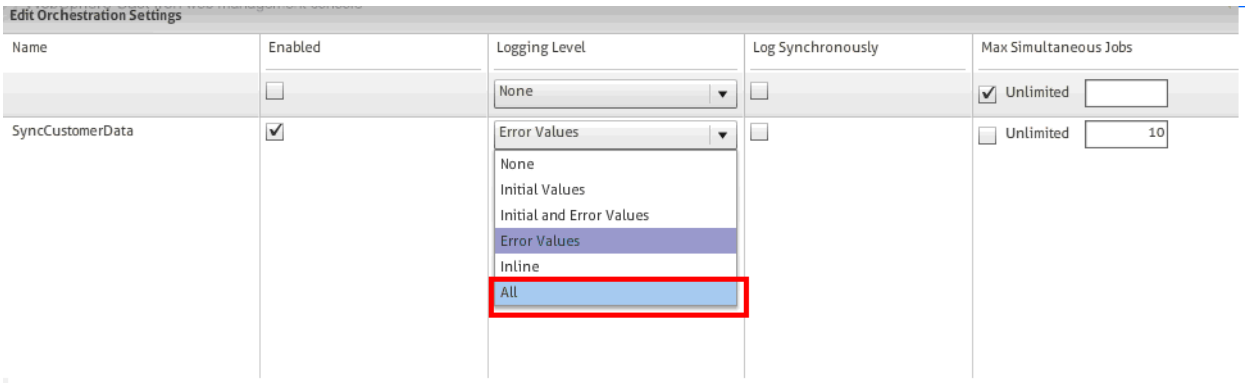




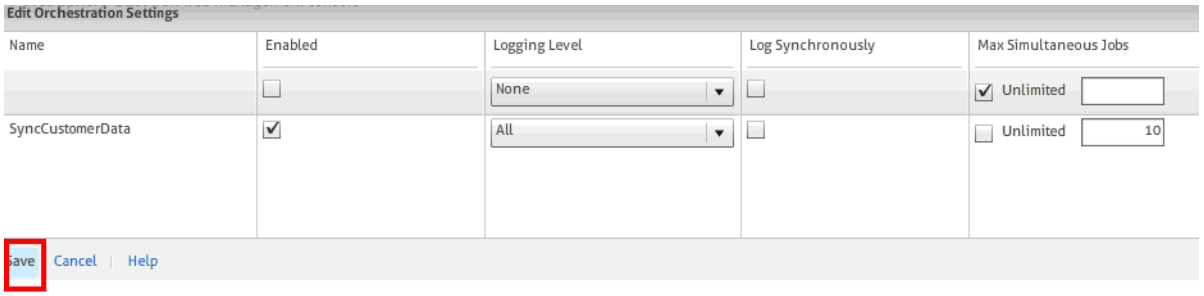
__ 60. To change the logging level, click **Edit**, which is located under your orchestration name **SyncCustomerData**.



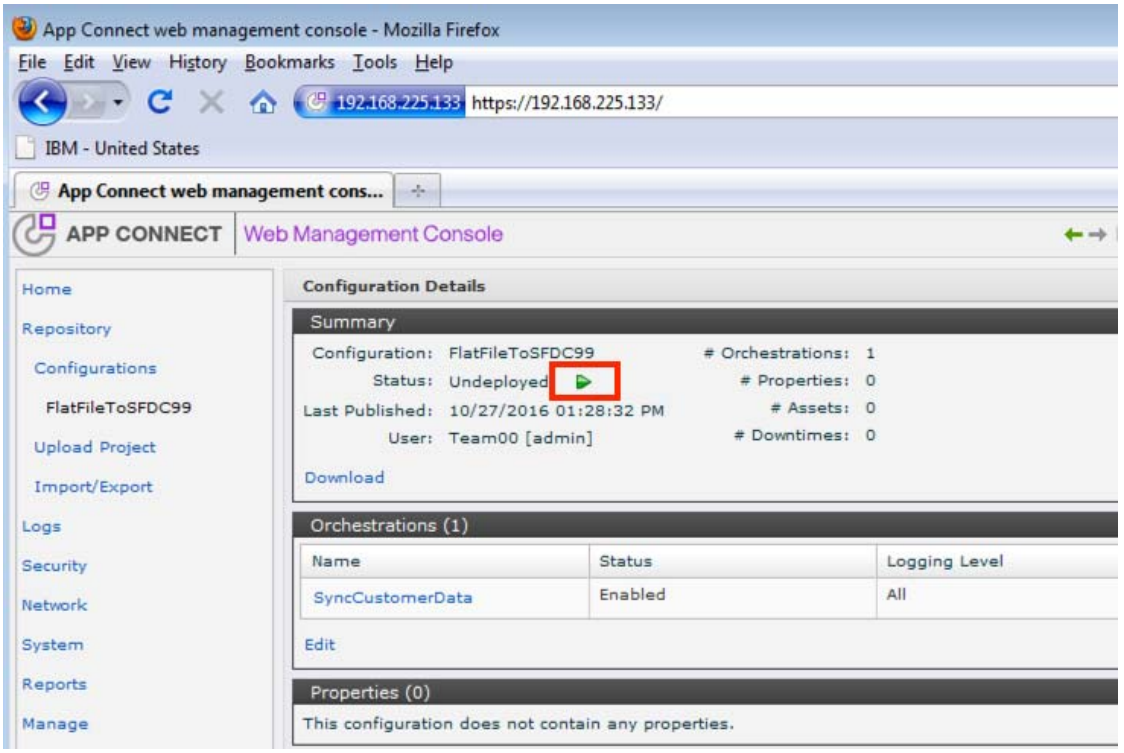
__ 61. Select **All** from the **Logging Level** menu for your enabled orchestration.



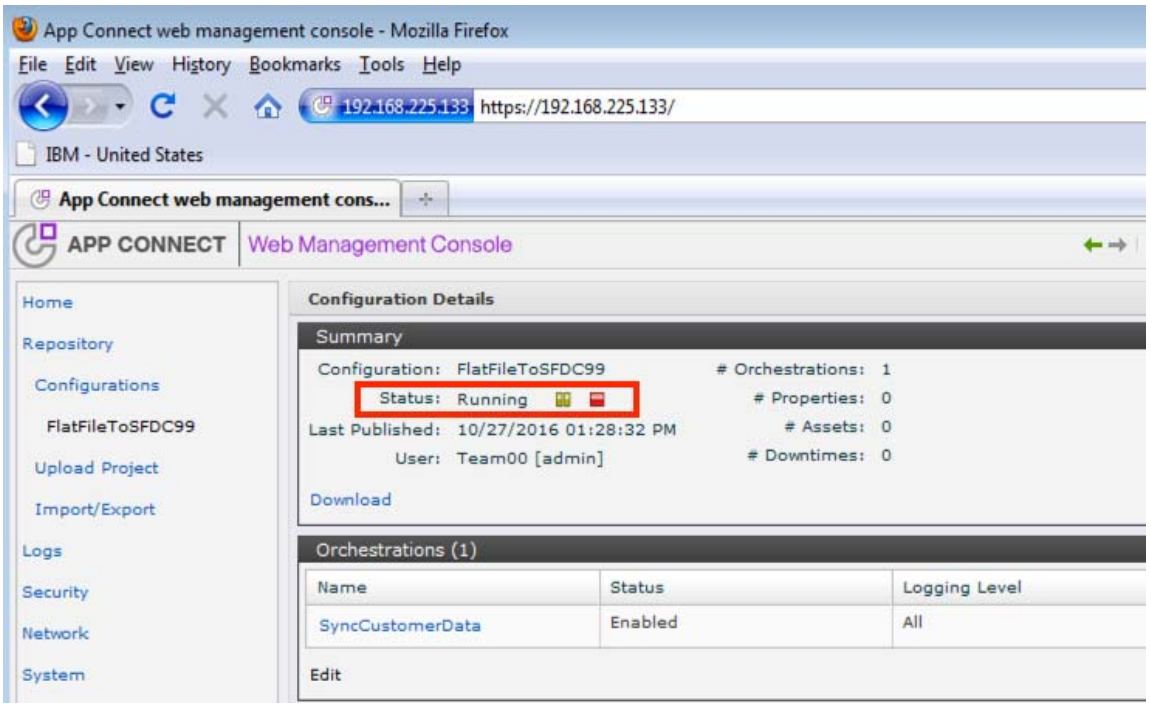
__ 62. To save the changes, click **Save** in the lower-left corner of the screen.



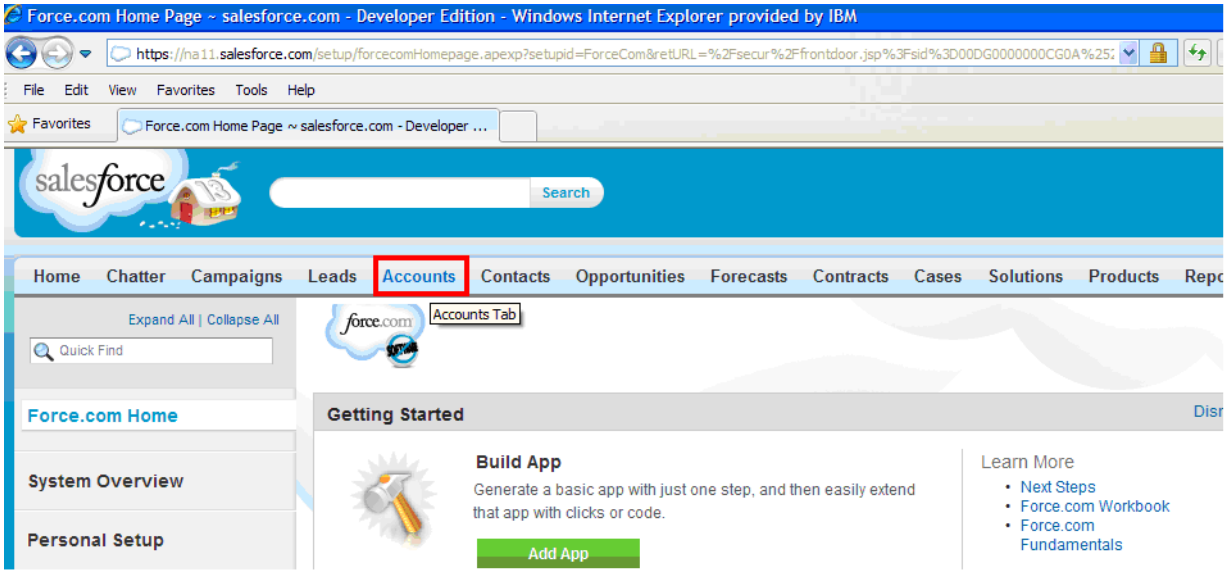
__ 63. Click the green **Run Configuration** icon to deploy the orchestration.



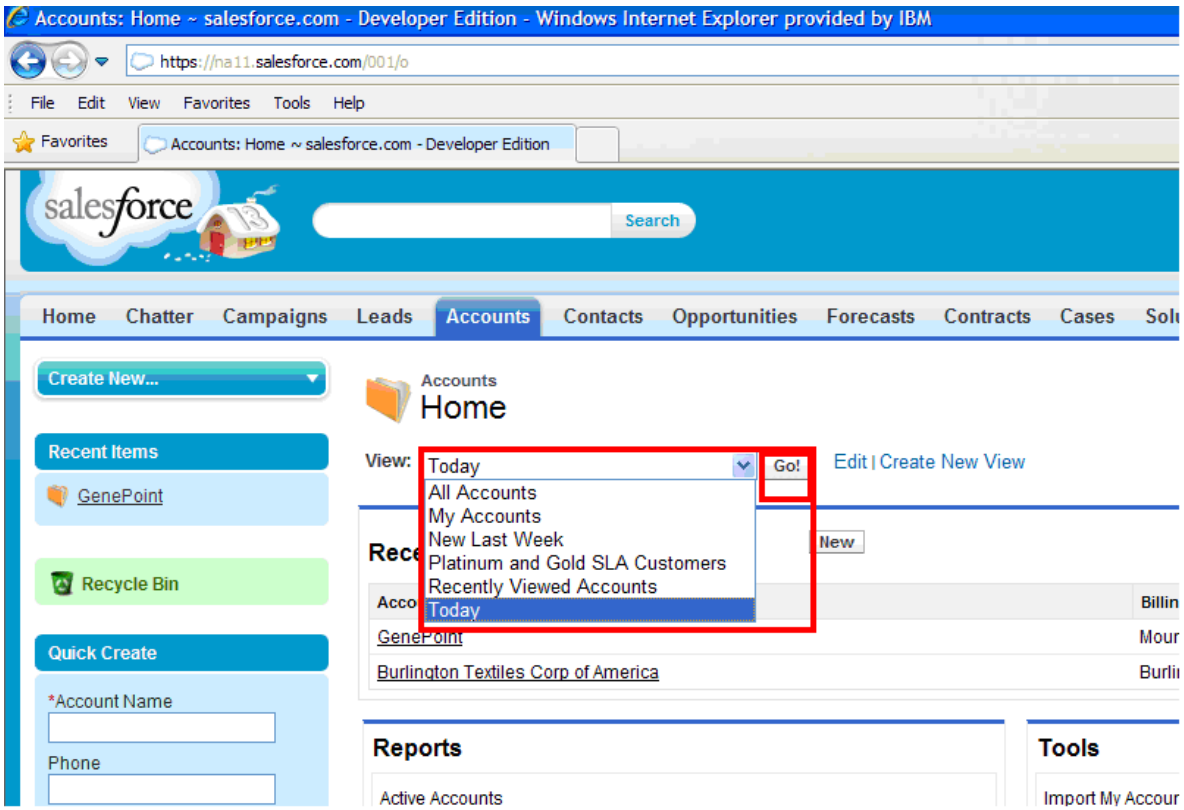
The orchestration then shows as being deployed and in a running state.



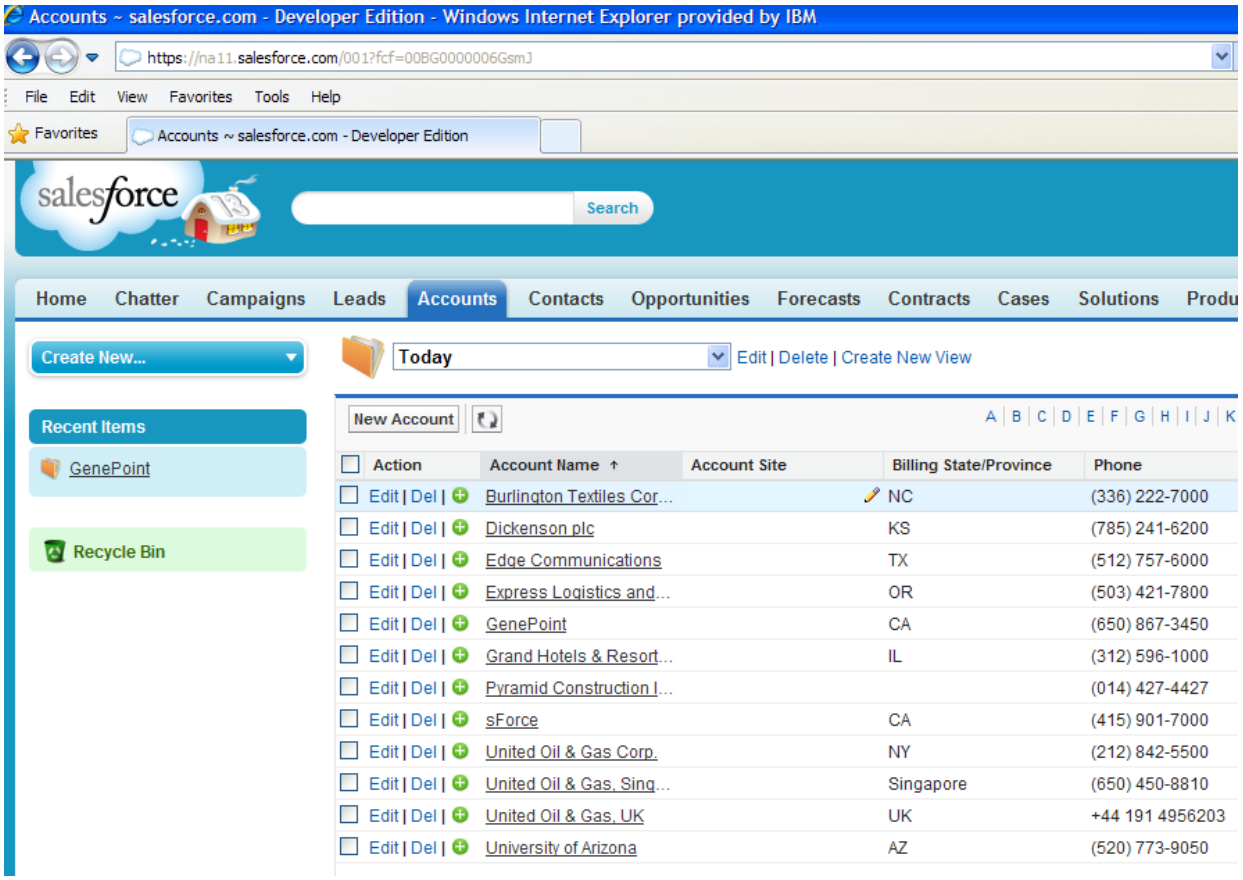
__ 64. Using a browser, log in to your Salesforce.com account and click **Accounts**.



__ 65. Select **Today** from the menu and click **Go!**



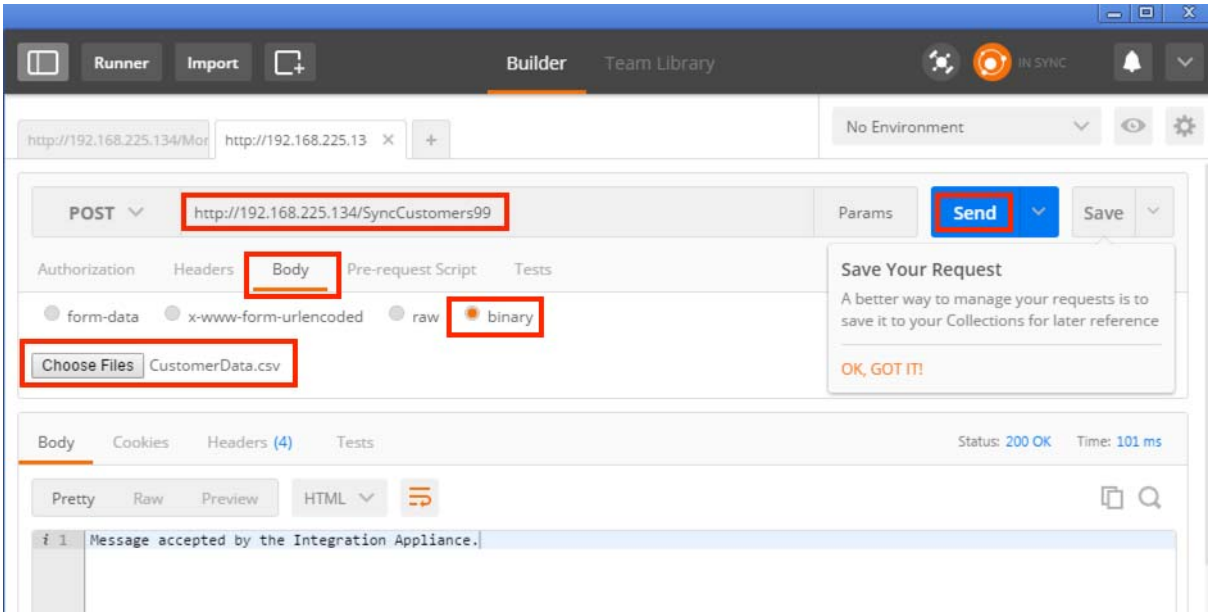
Your list of accounts should not show any test records that are to be used: **Customer One**, **Customer Two**, **Customer Three**, or **Customer Four** in the list.



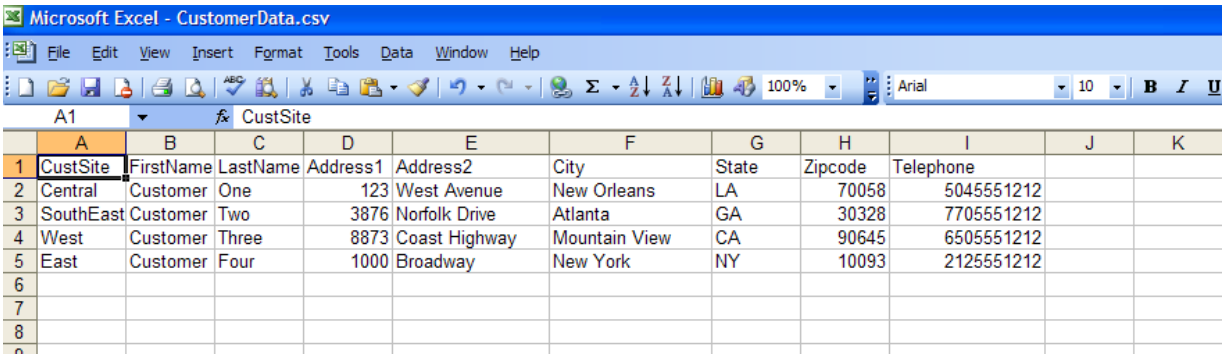
__ 66. To test your orchestration, open the **Postman** utility through the Desktop as in the previous lab. You can also download Postman utility. Using Chrome, go to this website: www.getpostman.com

__ 67. Enter the URL string that contains the App Connect Professional DATA IP address, which your instructor provides. Append the URL parameter `/SyncCustomersXX`, which is then used to invoke the orchestration on the App Connect Professional appliance.

__ 68. Click **Choose Files** and navigate to the `C:\Student\Lab Resources\Lab 2` folder to locate the `CustomerData.csv` document to post. Click **Open**. Then, click **Send** to invoke the orchestration. If it is done correctly, you get a message that says the “Message was Accepted by the Integration Appliance”.



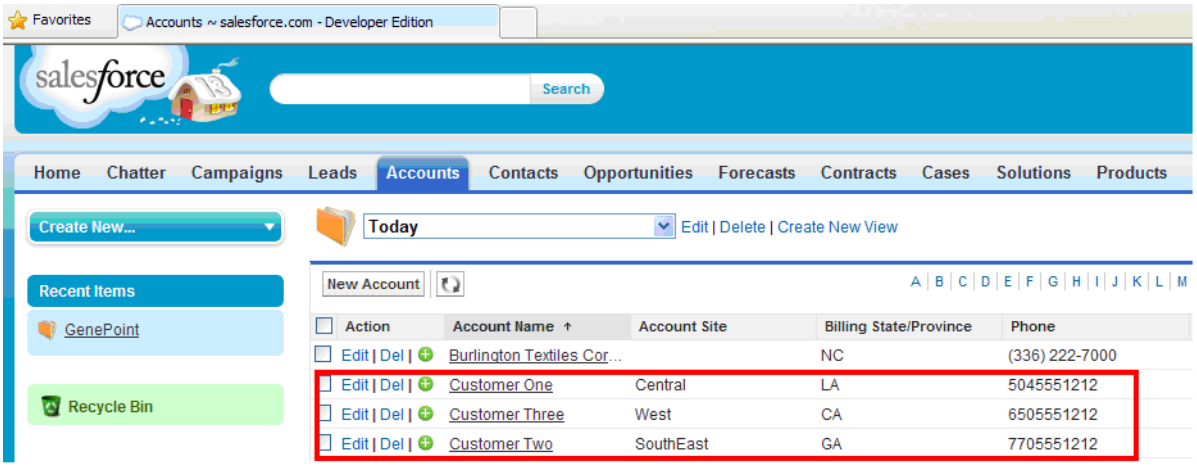
The CustomerData file contains four records to be processed. Since the orchestration is screening the records as they are processed, only those records where CustSite is not equal to `'East'` pass the validation. Only three records are created in your Salesforce.com instance.



	A	B	C	D	E	F	G	H	I	J	K
1	CustSite	FirstName	LastName	Address1	Address2	City	State	Zipcode	Telephone		
2	Central	Customer	One	123	West Avenue	New Orleans	LA	70058	5045551212		
3	SouthEast	Customer	Two	3876	Norfolk Drive	Atlanta	GA	30328	7705551212		
4	West	Customer	Three	8873	Coast Highway	Mountain View	CA	90645	6505551212		
5	East	Customer	Four	1000	Broadway	New York	NY	10093	2125551212		
6											
7											
8											

If your orchestration ran successfully, you should be able to go back to your Salesforce.com window and refresh your Accounts Today screen to see the three records that were created.

Notice that the Customer Four `'East'` record was not created.



End of exercise

Exercise review and wrap-up

In this exercise, you configured the HTTP Receive Request, Transform Read Flat File activity, For Each statement, and If Then…Else statements. You created objects within Salesforce and log messages that are used to create internal log messages.

Exercise 3. Single database table operation into Salesforce.com

Estimated time

00:75

Overview

In this lab, you read a DB2 database table record and create a new or update an existing Salesforce Account. Since an account originates in DB2, it is the “system of record”. The **Try** and **Log Message** activities are used to catch and log errors.

This exercise works as a synchronization of records from DB2 into Salesforce.com.

Objectives

After completing this exercise, you should be able to:

- Create a Database Poll table
- Write logic for the For Each statement
- Configure Salesforce.com upsert objects
- Configure the Send Email function
- Write logic for the Try statement
- Read the Utilities log message

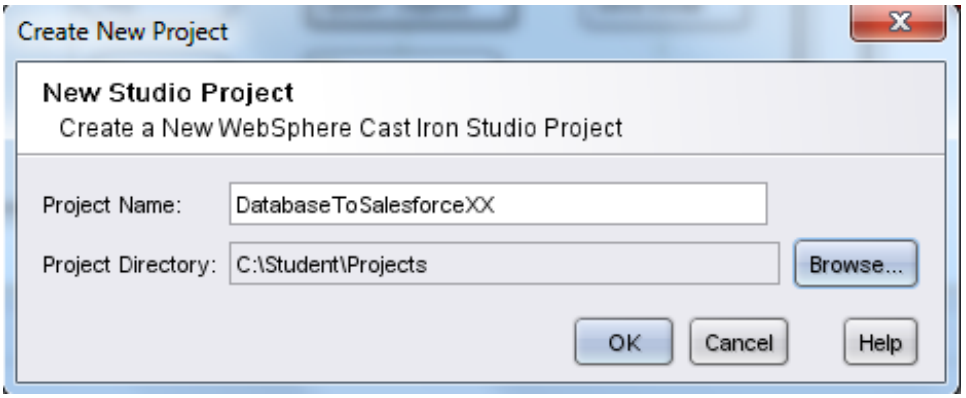
Introduction

The version 3.0 Salesforce.com connector is built in, so its usage is similar to that of the database connector. As with the database connector, Studio discovers the structure of connected objects. Logic activities can be used to interrogate input data and change processing paths, depending on field values, or to capture and handle database errors.

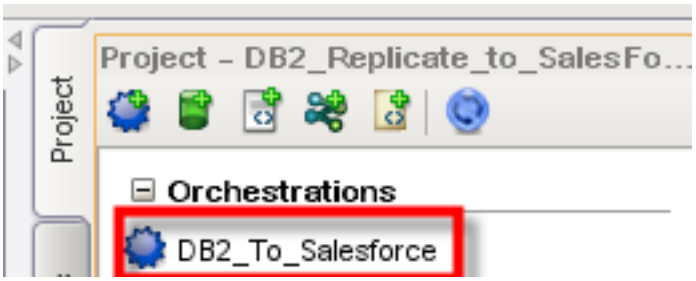
Exercise instructions

3.1. Load database records into Salesforce

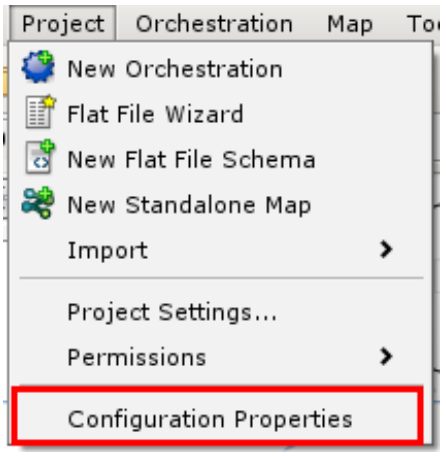
__ 1. Create a project: `DatabaseToSalesforceXX`



__ 2. Define the orchestration by renaming the default orchestration to: `DB2_To_Salesforce`
Right-click **Orchestration** and select **Rename**, or double-click the orchestration name.

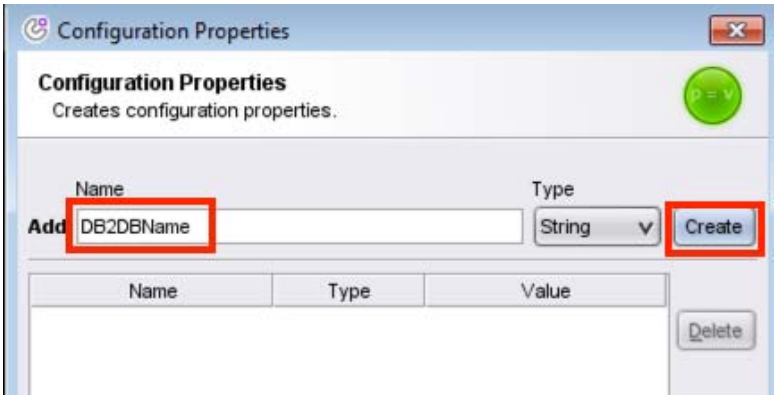


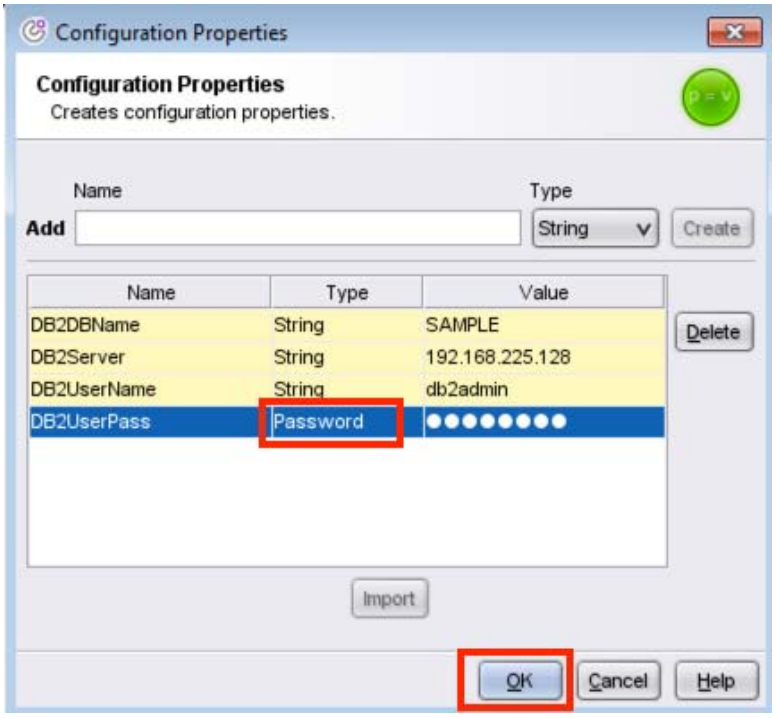
__ 3. Create configuration properties. Select **Project > Configuration Properties**.



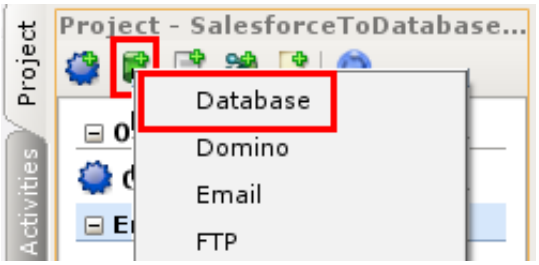
__ 4. Add the following configuration properties (see the Connection Parameters spreadsheet):

- DB2DBName: `SAMPLE`
- DB2Server: `XX.XX.XX.XX` - Endpoint Server Address (DB2)
- DB2UserName: `db2admin`
- DB2UserPass: `db2admin`
  ***(type = password) see highlighted in RED.
- When all the parameters and values are added, select **OK**. Make sure that you add `DB2UserPass` as the Password type.





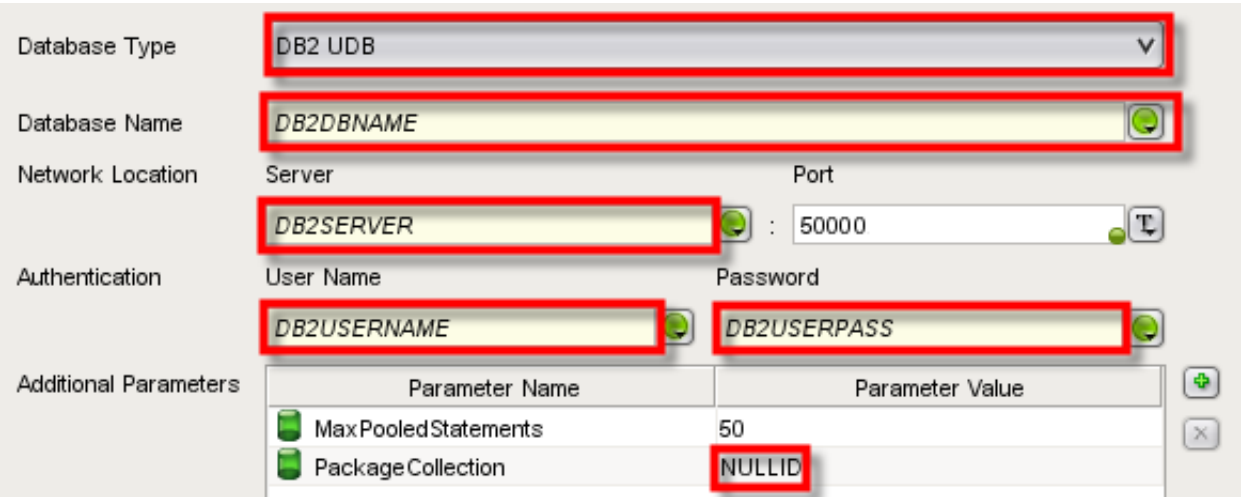
__ 5. Create a Database DB2 endpoint.



__ a. On the right side of the screen, under the endpoint listing, rename the endpoint by right-clicking the default endpoint name **Database** and renaming it: `DB2`

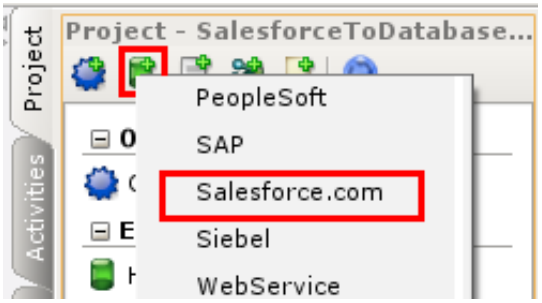
__ b. From the **Database Type** list, select **DB2 UDB**.

__ c. Select the list for the four fields corresponding to the configuration parameters entered in Step 3.
Under **Additional Parameters**, set **PackageCollection** to string `NULLID`.

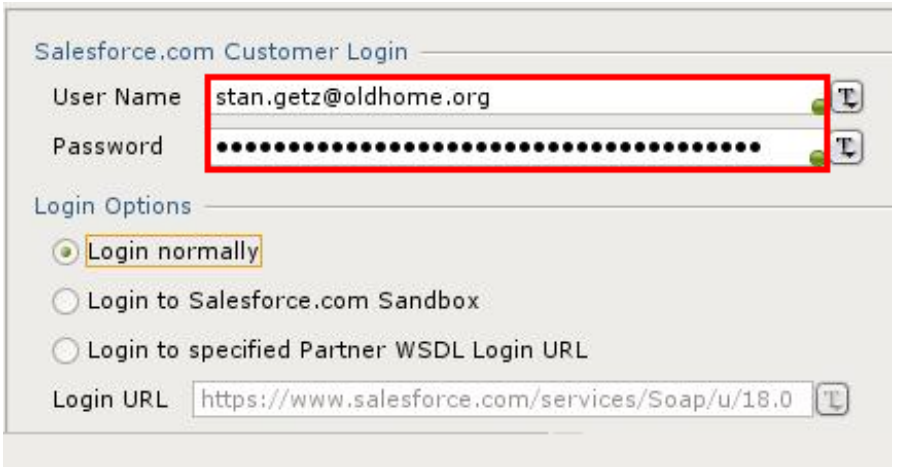


__ d. Test the connection on the bottom of the page.

__ 6. Create a Salesforce.com endpoint (use information from the Connection Parameters spreadsheet).

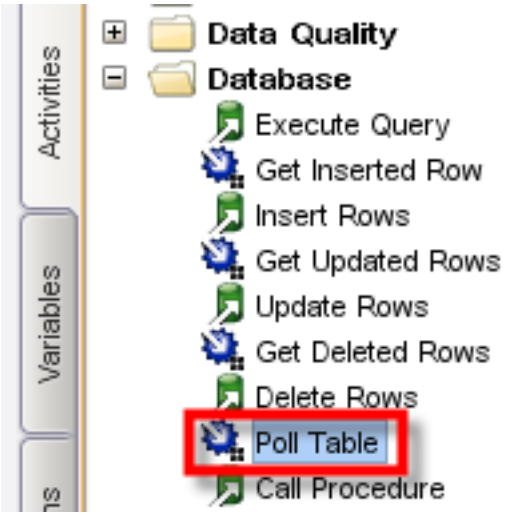


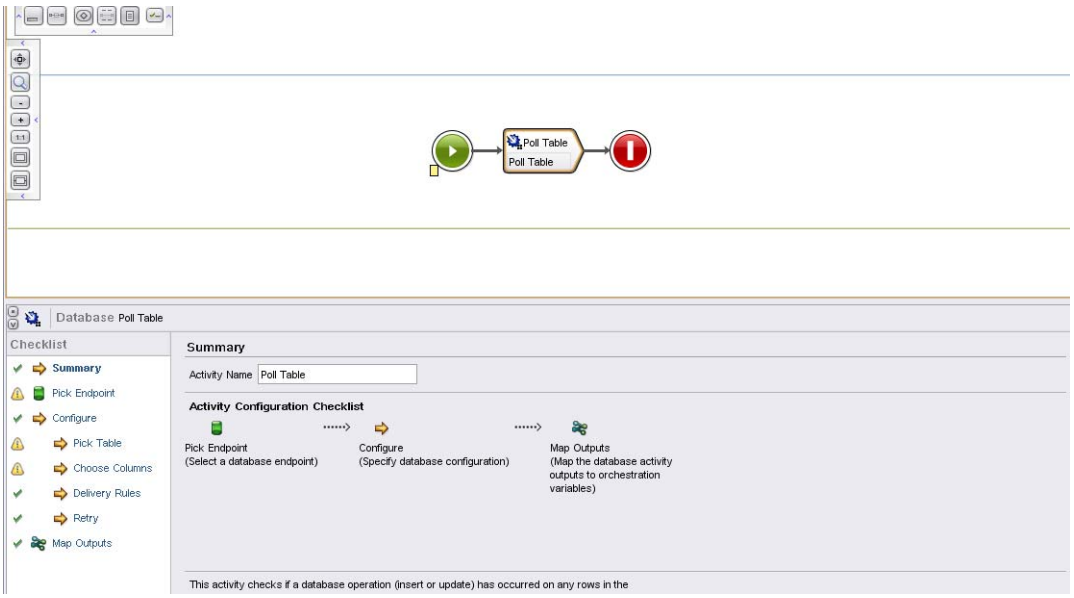
__ a. Enter your Salesforce user name. The Password field is constructed as: `<password><security token>`



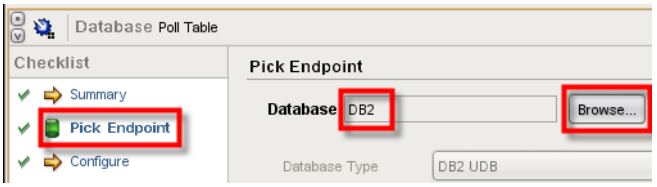
__ 7. Test your connection.

__ 8. Add a **Database Poll Table** activity to the orchestration by dragging the activity onto the canvas. Then, click the activity and the checklist becomes active on the bottom part of the screen.



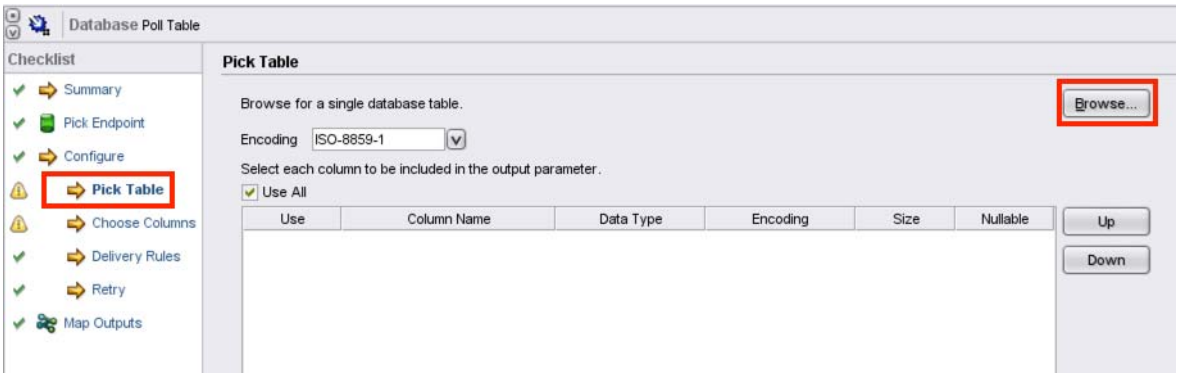


__ a. Working your way down the checklist, start with **Pick Endpoint** and select the **DB2** endpoint.

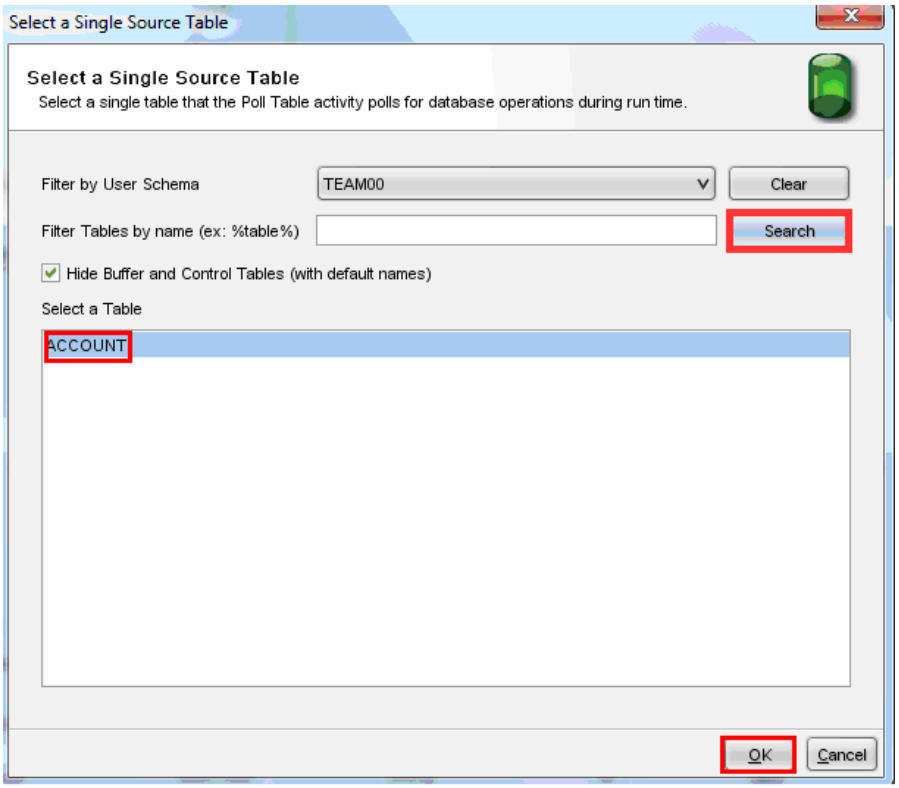


__ 9. Pick Table:

__ a. Click **Browse**.



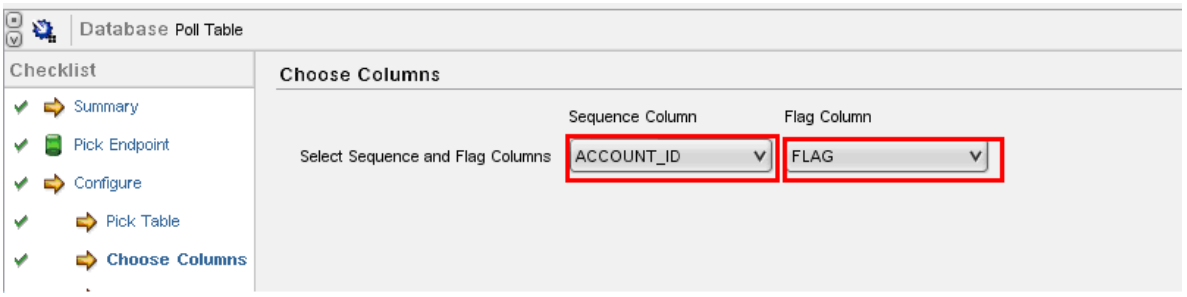
__ b. From the drop-down list, select the **Team00** schema. Click **Search**, and the connector populates the list of all the tables that are available to you. As soon as the requisite **Account** table is selected, click **OK**.



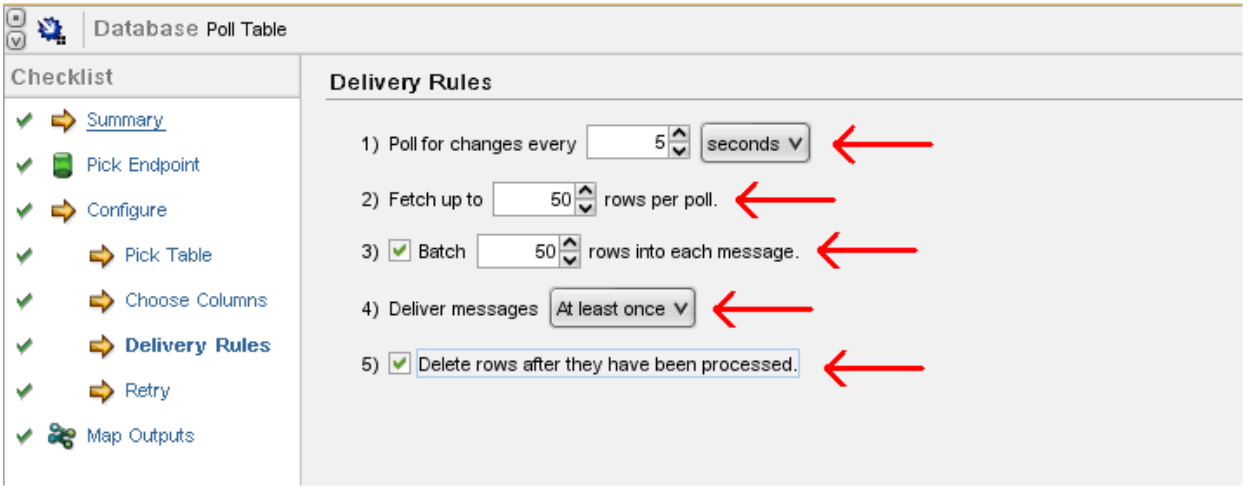
__ c. Clear **Use All** and select fields from the table that is shown.

DB2 Activity Field
ACCOUNT_ID
ACCOUNT_NAME
ACCOUNT_TYPE
ACCOUNT_NUMBER
BILLING_STREET
BILLING_CITY
BILLING_STATE
BILLING_ZIP_CODE
BILLING_COUNTRY
PHONE
INDUSTRY
OWNERSHIP
ACCOUNT_RATING
CUSTOMER_PRIORITY
SLA
ACTIVE
NUMBER_OF_LOCATIONS
UPSELL_OPPORTUNITY
SLA_EXPIRATION_DATE
FLAG

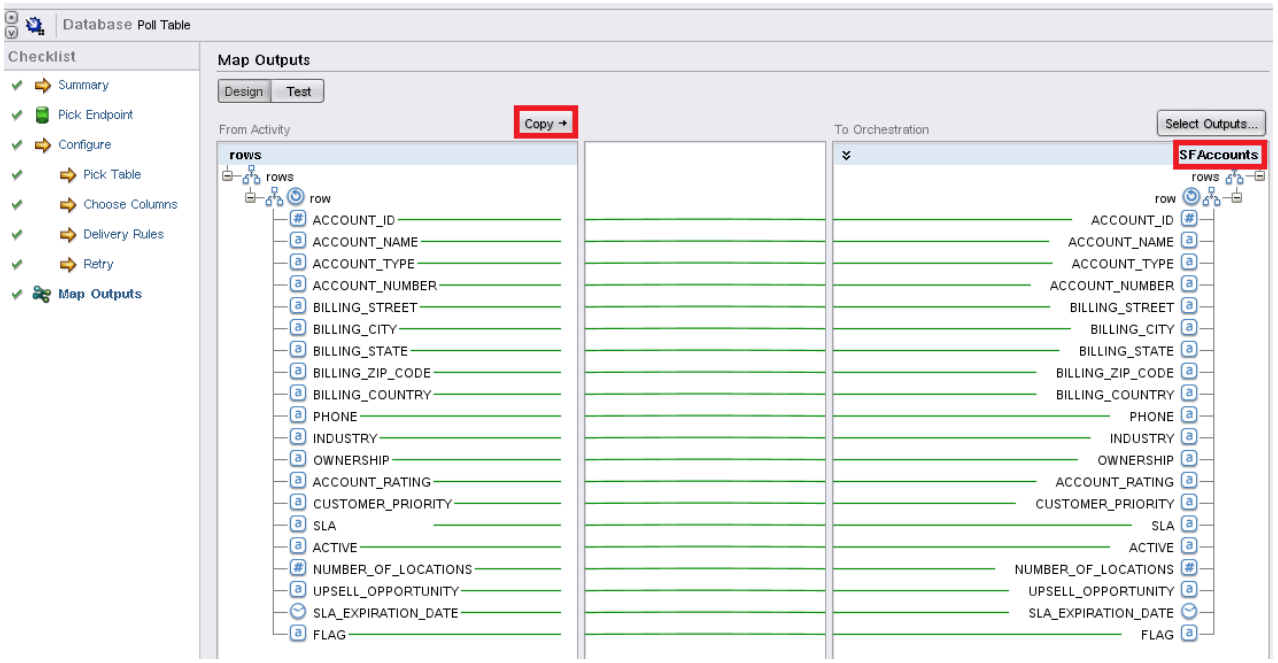
__ d. Choose Columns: Ensure that the **Sequence Column** and **Flag Column** properties are set up on the drop-down lists as shown in the screen capture.



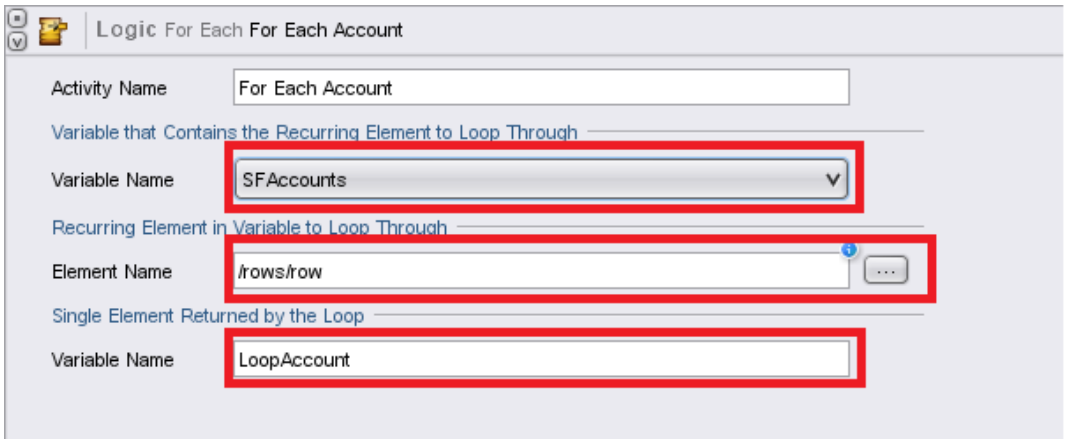
__ 10. Delivery Rules. Ensure that the polling parameters are indicated as follows:



__ 11. Click **Map Outputs**. Click **Copy** to copy the rows that are returned from the DB to a variable. Rename the output variable from `rows` to: **`SFAccounts`**
You are now finished with configuring the Poll Table activity.

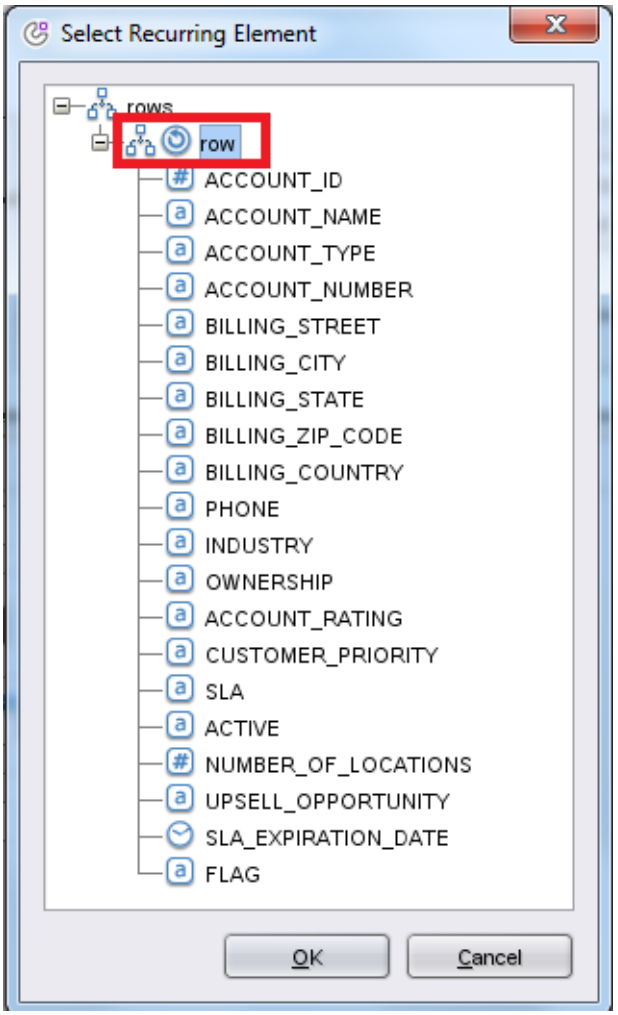


__ 12. Drag the **For Each** activity onto the canvas. This action causes the orchestration to iterate a loop that goes through each of the values that are returned from the database. Configure the activity as shown:



__ a. Variable Name (containing recurring element to loop through): From the menu, select the variable name of the data set that you want to iterate through. In this case, it is the variable `SFAccounts`.

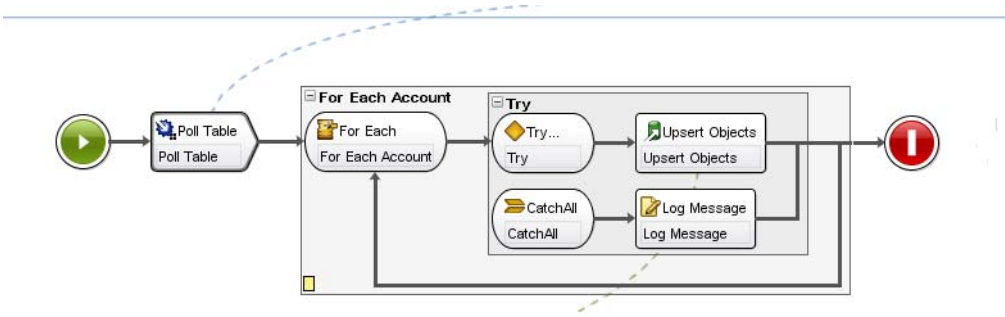
__ b. Element Name: Click the dialog button on the right side to select the appropriate element on the data set. With the **For Each** function, this element is always a repeating variable that has the repeating symbol. In your case, it is going to be the `row` variable. Select **row** and then click **OK**.



__ c. Variable Name (containing a single element that is contained by group): This variable gets created and represents each individual occurrence of `row`. Name this variable: `LoopAccount`

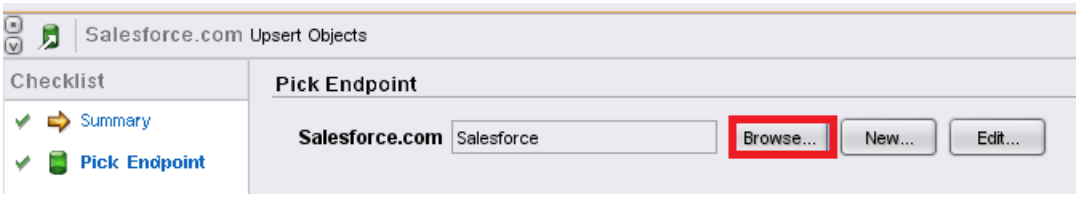
__ 13. You now create the process that executes once for each row that comes in from the input. All of the subsequent activities are dropped into the **For Each** box. Drag the following activities from the **Activities** tab as depicted in the picture that follows. You configure each subsequent activity afterward.

- **Try..Catch**
- **Salesforce.com > Upsert Objects** (next to the Try..Catch Box)
- **Log Message** (next to the **Catch All** Block)

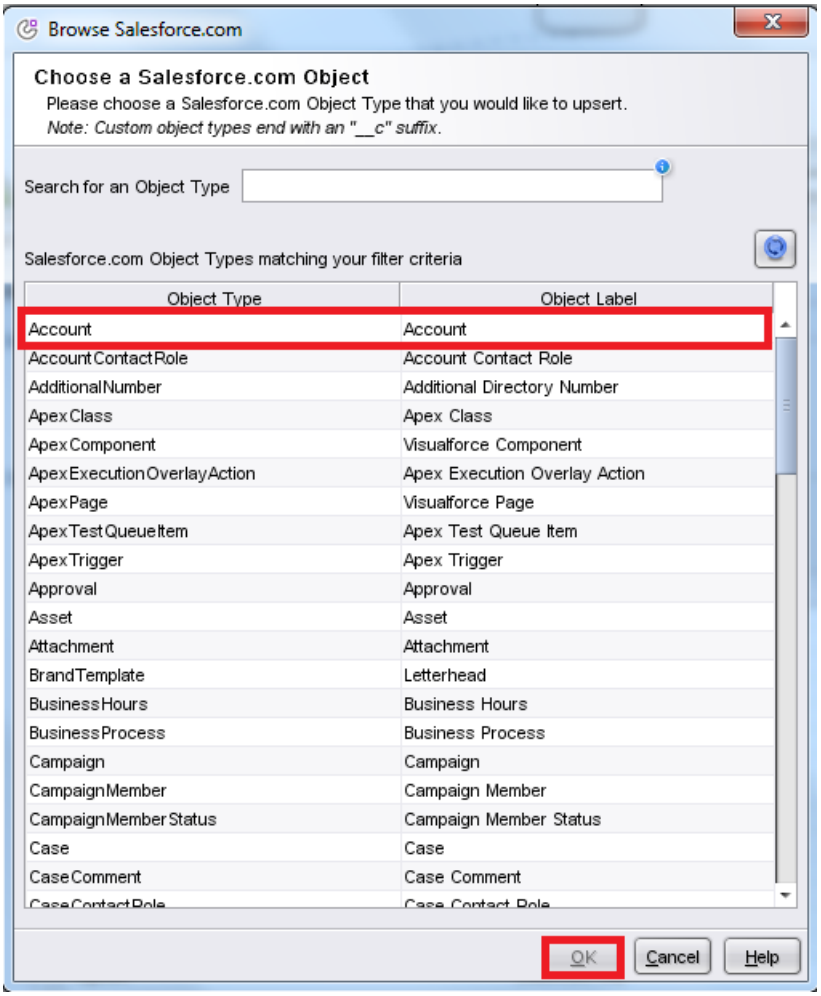


__ 14. The next activity to configure is the Salesforce.com **Upsert Objects**. Click the activity on the orchestration canvas.

__ a. For the Pick Endpoint step in the checklist, click **Browse** and select the **Salesforce** endpoint.



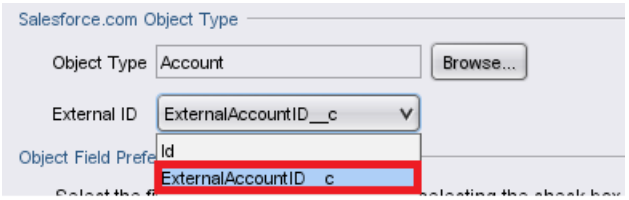
__ b. Select the appropriate object. For this lab, you use the **Account** object. Select **Account** and **OK**.



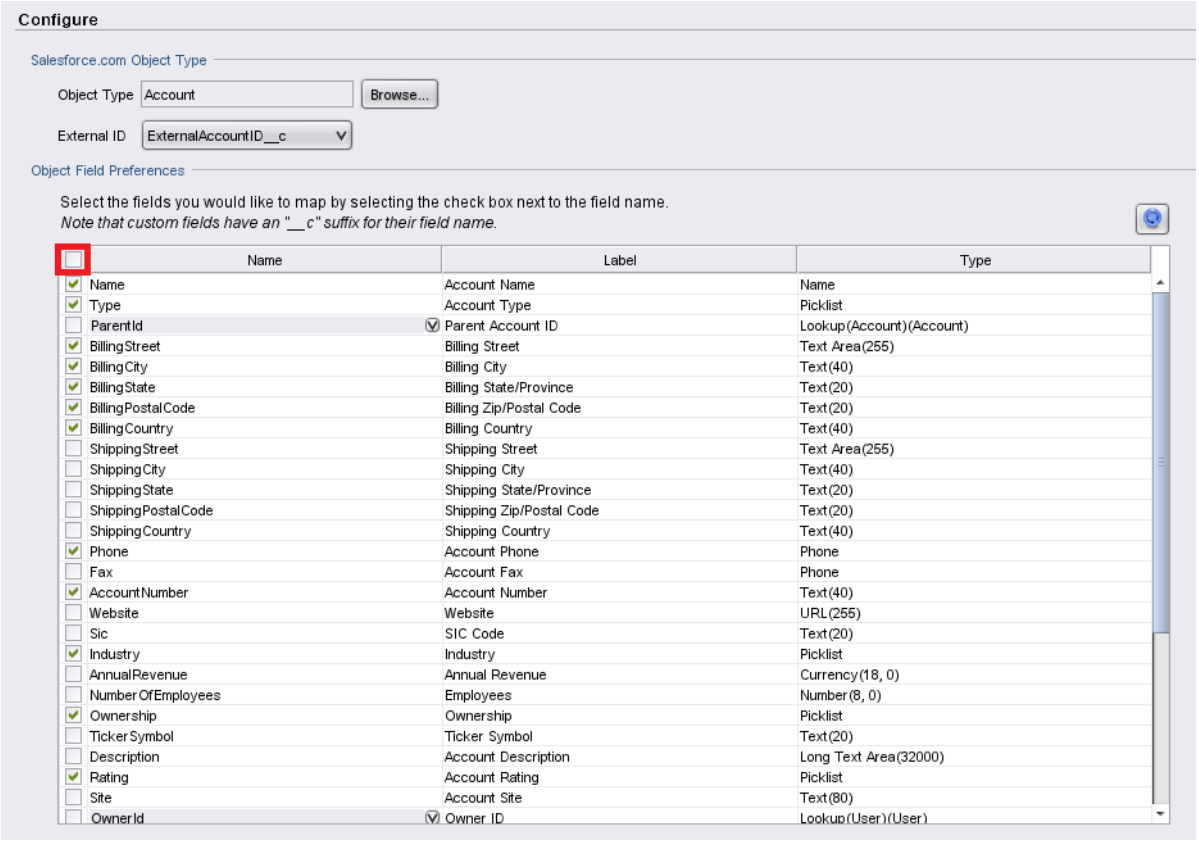
__ c. Select the proper **External ID** to be used to drive the functionality around the Upsert. This custom field represents a key value that ties the record in Salesforce with the data in your on-premises DB2 table. If the record exists, it updates the data. If it does not, it inserts a new record.

Important

If you do not have a custom field that is already set up for the External ID, see Appendix B.



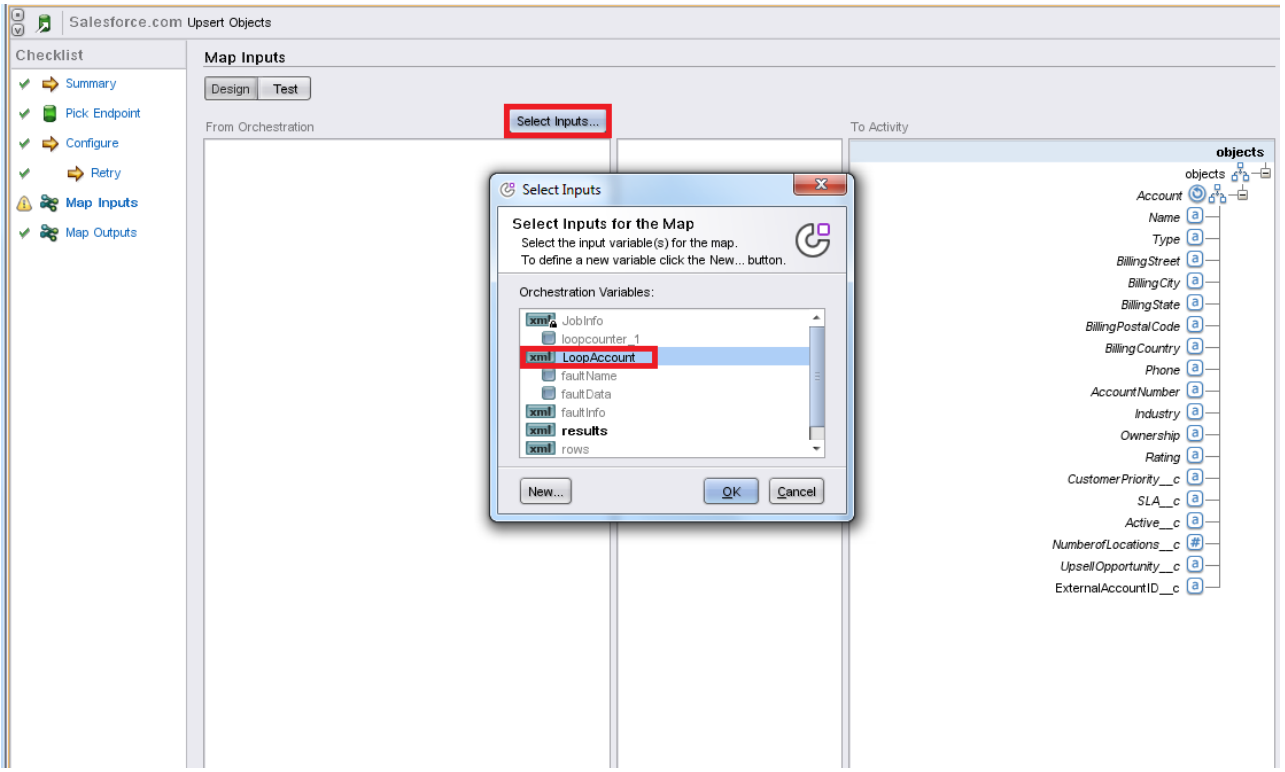
__ d. To clear all fields, ensure that the check box at the top of the field selection screen is ***not*** selected. Proceed to the next step to select the fields to be used.



__ e. Select the following fields to be used in the **Account** object. Check the check box in front of the field name to enable it.

Salesforce.com Fields
AccountNumber
Name
Type
BillingStreet
BillingCity
BillingState
BillingPostalCode
BillingCountry
Phone
Industry
Ownership
Rating
CustomerPriority__c
SLA__c
Active__c
NumberofLocations__c
UpsellOpportunity__c
SLAExpirationDate__c
ExternalAccountID__c

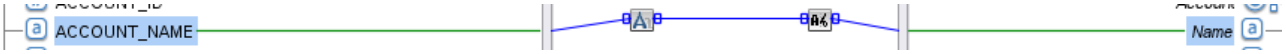
__ f. Map Inputs. Map the fields from the left side (input) to the right side (output). Use the mapping guide chart that follows to assist you.



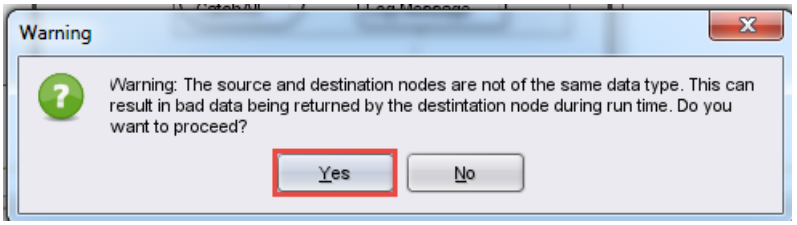
A	B	C
Input Field	Output Field	Transformation Rules
ACCOUNT_ID	n/a	Not Mapped
ACCOUNT_NAME	Name	Uppercase, Trimmed
ACCOUNT_TYPE	Type	Direct Mapping, no transformation
ACCOUNT_NUMBER	AccountNumber and ExternalAccountID__c	Direct Mapping, no transformation
BILLING_STREET	BillingStreet	Direct Mapping, no transformation
BILLING_CITY	BillingCity	Direct Mapping, no transformation
BILLING_STATE	BillingState	Direct Mapping, no transformation
BILLING_ZIP_CODE	BillingPostalCode	Direct Mapping, no transformation
BILLING_COUNTRY	BillingCountry	Direct Mapping, no transformation
PHONE	Phone	Direct Mapping, no transformation
INDUSTRY	Industry	Direct Mapping, no transformation
OWNERSHIP	Ownership	Direct Mapping, no transformation
ACCOUNT_RATING	Rating	Direct Mapping, no transformation
CUSTOMER_PRIORITY	CustomerPriority__c	Direct Mapping, no transformation
SLA	SLA__c	Direct Mapping, no transformation
ACTIVE	Active__c	Direct Mapping, no transformation
NUMBER_OF_LOCATIONS	NumberOfLocations__c	Direct Mapping, no transformation
UPSELL_OPPORTUNITY	UpsellOpportunity__c	Direct Mapping, no transformation
SLA_EXPIRATION_DATE	n/a	Not Mapped
FLAG	n/a	Not Mapped

Important

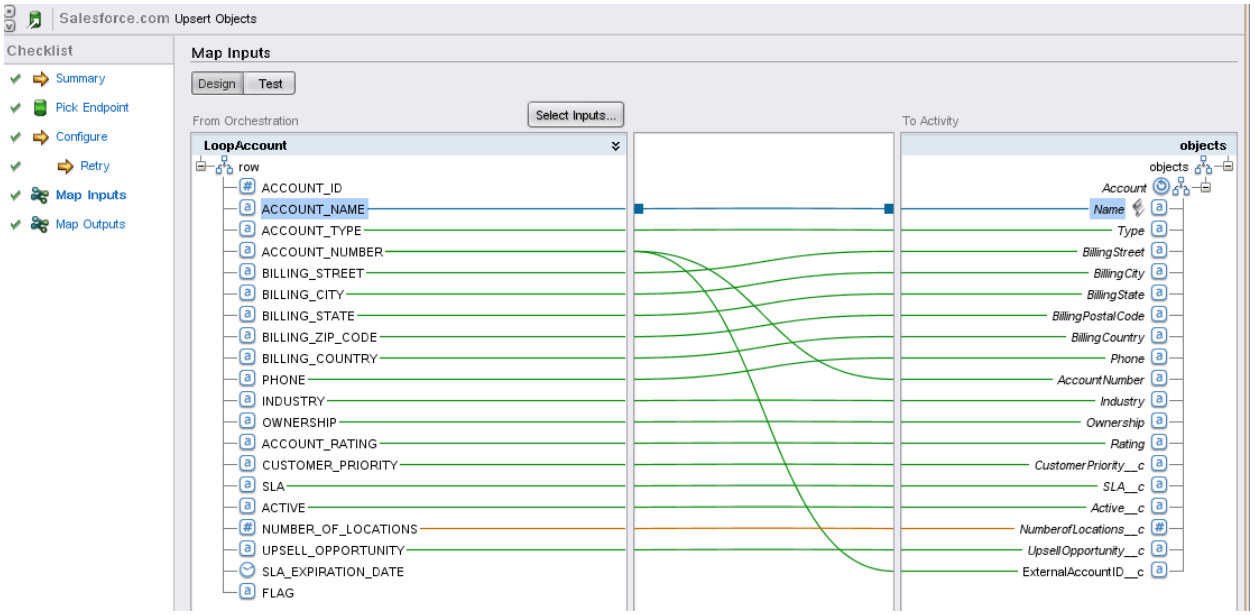
Use the functions tab to apply any transformation rules. Drag the **Upper Case** and **Trim** functions onto the center pane and drag the inputs onto the function. When properly mapped, account name looks as follows (be sure to right-click and apply the function graph when done).



You might receive a warning about mismatched data types when mapping numeric fields. For this exercise, you can disregard this warning.



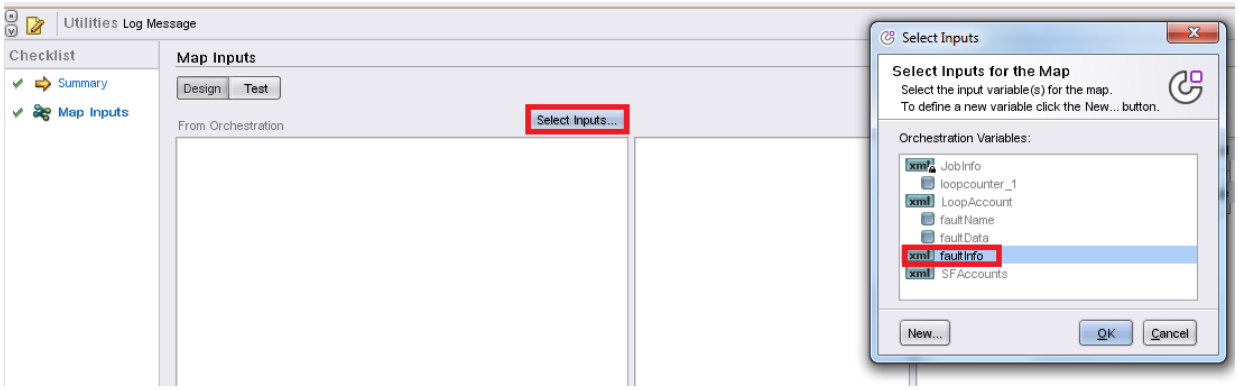
When complete, the mapping grid looks as follows:



__ 15. Configure the **Log Message** activity. This Activity is in the **Catch All** block of the **Try..Catch** function. Whenever a failure to insert the record into Salesforce.com occurs, the API provides an error message. You map the requisite information to populate the WMC with some customized messaging to assist with troubleshooting.

__ 16. Move on to the Map Inputs step

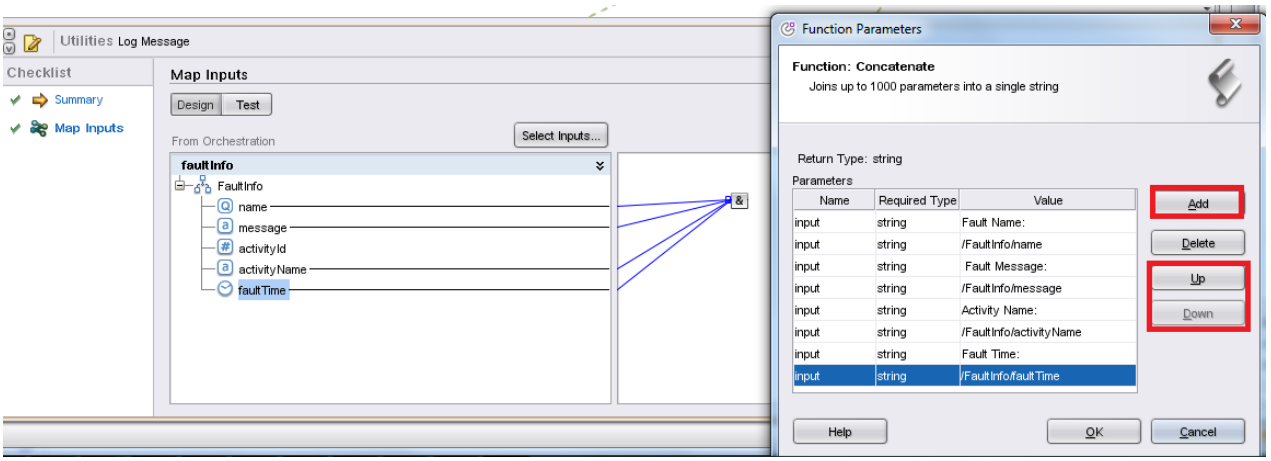
__ a. Map inputs. By default, when a **Try..Catch** is placed onto an orchestration, it automatically creates a variable that is called `faultInfo`. This variable contains all the data about a specified fault. Click **Select Inputs** to add that variable to the screen. Click **OK**.



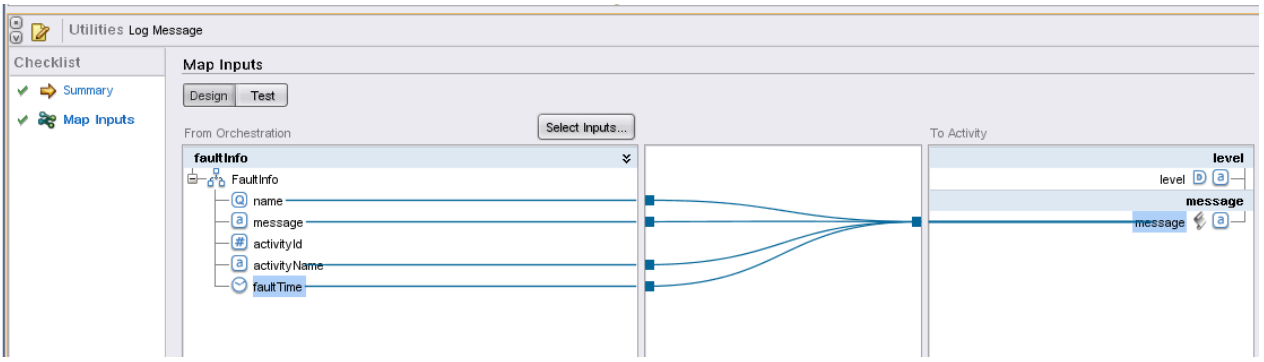
__ b. Use the following mapping guide to map the fields.

Input Field	Output Field	Transformation Rules
Fault Name	message	concatenated
Message	message	concatenated
activityId	message	not mapped
activityName	n/a	concatenated
faultTime	message	concatenated
	level	set to default "Warning"

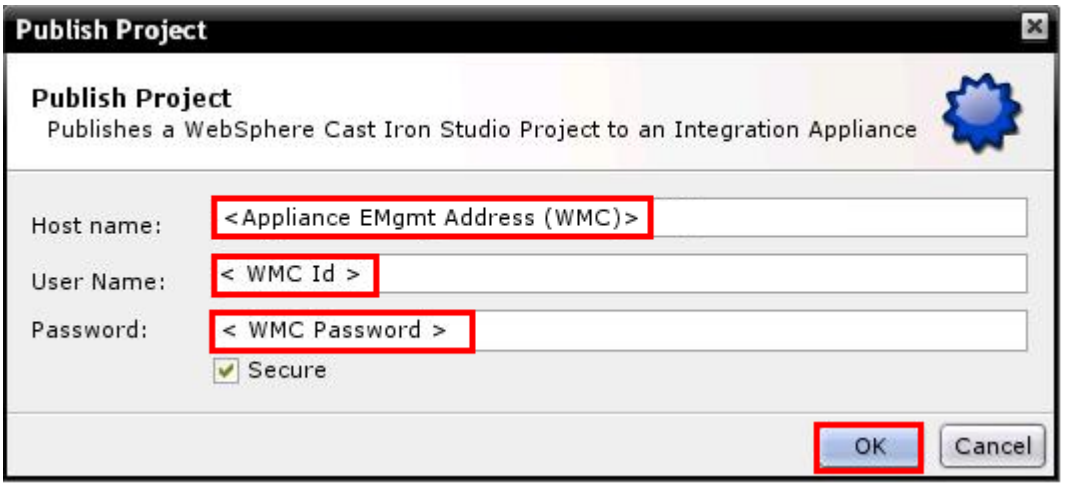
__ c. Use literal values on the concatenate function to separate the fields by clicking **Add** on the concatenate function dialog box. Use the **Up** and **Down** buttons to adjust the order of the fields that are being concatenated.



When complete, the activity should look as follows:



__ 17. Deploy the project to your appliance with **File > Publish** or by using the icon from the top toolbar. Use the credentials from the previous lab.



__ 18. Log in to the Web Management Console and then find your project on the dashboard. If you like, use the **Configurations Filter** to show only your projects. To start the orchestration, click the green play icon.

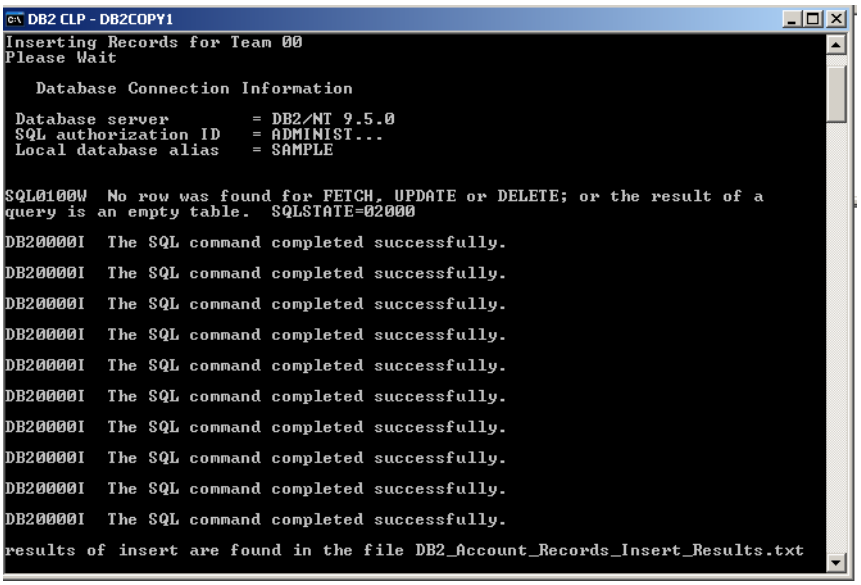
__ 19. To run the orchestration, you need to insert some rows into the account table. You can do so by executing the following script from your VM.

__ a. Go to **Start > Command Prompt**. From the command prompt, type `db2cmd` to start the DB2 command-line interface.

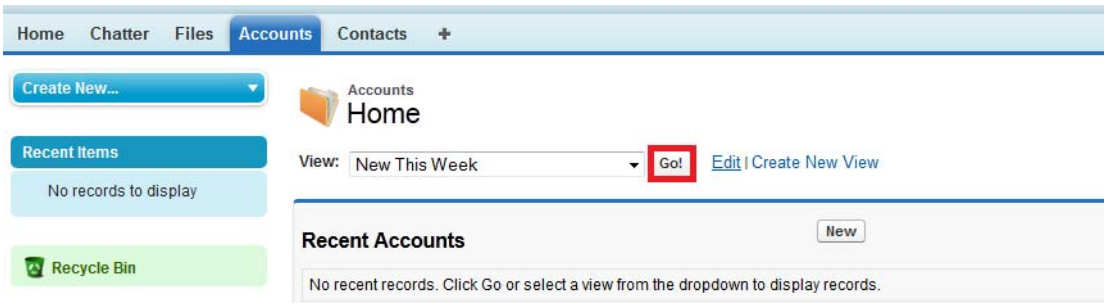
__ b. Next, change directories to `C:\Student\Lab Resources\Lab 3`. The screen capture that is shown has the commands to change the directory.

__ c. Execute the `InsertAccountRecords.bat` batch file, which starts the process to insert the records. You see a prompt that asks for your team ID. All students use the **00** team. Type `00` and then press the Enter key.

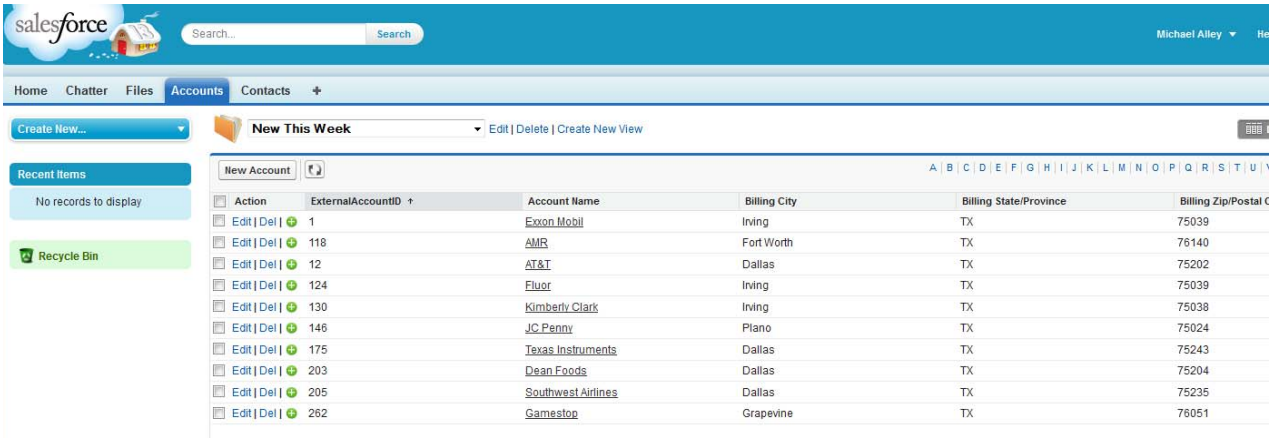
__ d. The script runs and then inserts the rows.



__ 20. To see whether everything ran properly, you need to log in to Salesforce.com and then go to the **Accounts** object. Click a new view that is called **New This Week**.



__ 21. If everything ran properly, you see 10 new accounts.



__ 22. You can also log in to the Web Management Console to analyze the run results. If everything ran correctly, the logs contain only minimal details. You can enable more detailed logging by stopping the project, undeploying, and setting the logging to all (see exercise #1 for a sample).

End of exercise

Exercise review and wrap-up

This exercise read a DB2 database table record and updated an existing Salesforce account.

Exercise 4. Loading a Salesforce.com account to ESB by using IBM MQ

Estimated time

00:75

Overview

This lab demonstrates how App Connect Professional can be used to extract information from a SaaS (software as a service) based application and format messages that can be passed along to an ESB backbone. ESBs display their interfaces typically by two means: web services (either SOAP or REST) and messaging middleware, such as IBM MQ, which is the focus of this lab.

Objectives

After completing this exercise, you should be able to:

- Configure HTTP Receive
- Configure Salesforce.com query objects
- Configure Map variables
- Write XML
- Create IBM MQ Put messages

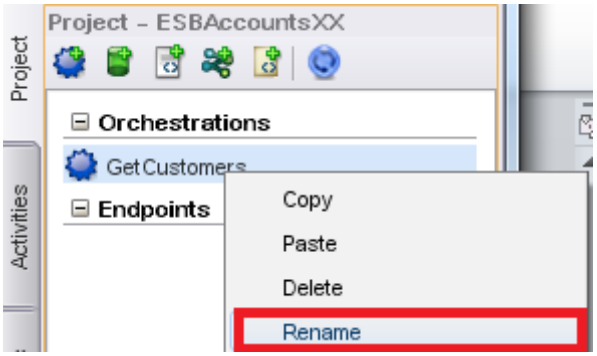
Introduction

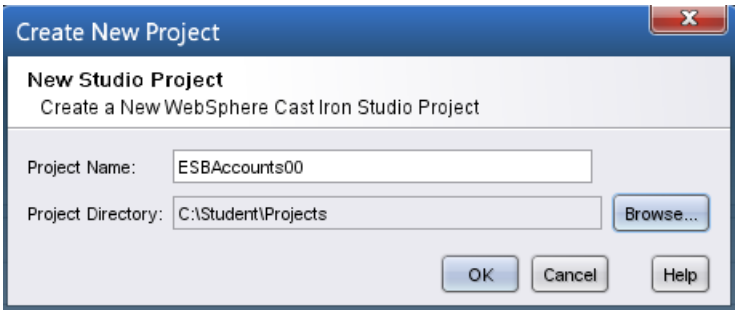
As seen in previous labs, App Connect Professional has a preconfigured Salesforce.com connector that provides a fast and simple means by which to interact with the Force.com API with no code. You use the same concepts to extract account data from Salesforce by using the native query language called SOQL (Salesforce Object Query Language). You take this data and serialize it into an XML message that gets dropped onto a message queue. It is assumed then that the ESB is going to consume the message and route it to some other back-end system.

Exercise instructions

4.1. Load Salesforce.com Accounts to ESB by using IBM MQ

__ 1. Create a project: `ESBAccounts00`
Rename the orchestration to: `GetCustomers`





__ 2. Set up the configuration properties by selecting **Project > Configuration Properties**. Create each of the configuration properties that are listed as follows:

__ a. **Salesforce.id**: Your salesforce.com user ID. Use the same ID as in previous exercises.

__ b. **Salesforce.pw**: Create as type password. Copy and paste in your password+tokenid.

__ c. **http.listen.port**: The listening port for the HTTP endpoint of the orchestration. Set it to port **80**.

__ d. **QNAME**: The queue name to use. The queue that you use is: `STUDENT1.OUT`

__ e. **MQHOST**: The host name of the IBM MQ server. See the Connection Parameters spreadsheet for the appropriate IP address or host name.

__ f. MQPORT: Hardcoded as `1414`

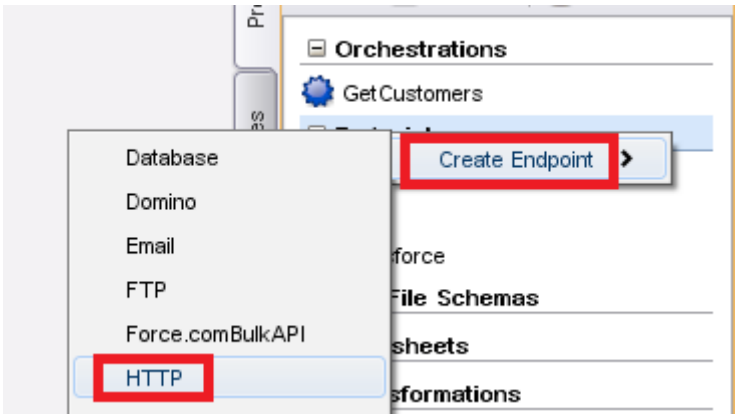
__ g. QMGR: Hardcoded as `CASTIRON`

__ h. ServerChannel: Hardcoded as `SVR.CHL`

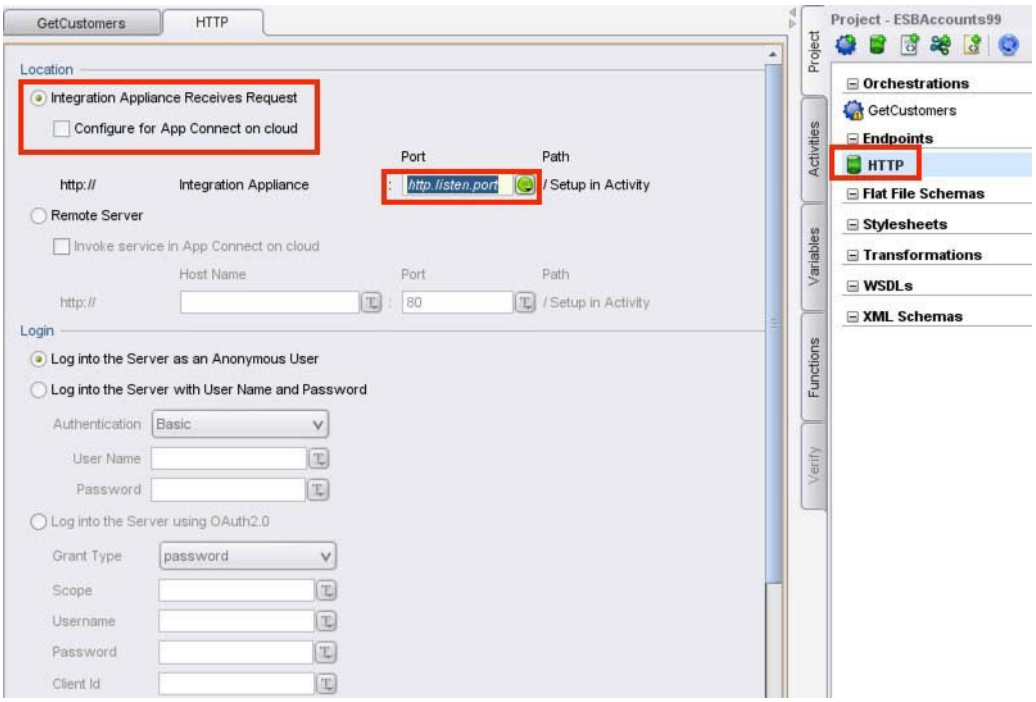
__ i. MQ User: Hardcode to `db2admin`

__ 3. Create the endpoints.

__ a. Create an **HTTP** endpoint. Right-click **Endpoints** and select **Create Endpoint**, which is the HTTP Listener that is going to enable the orchestration to execute in real time.



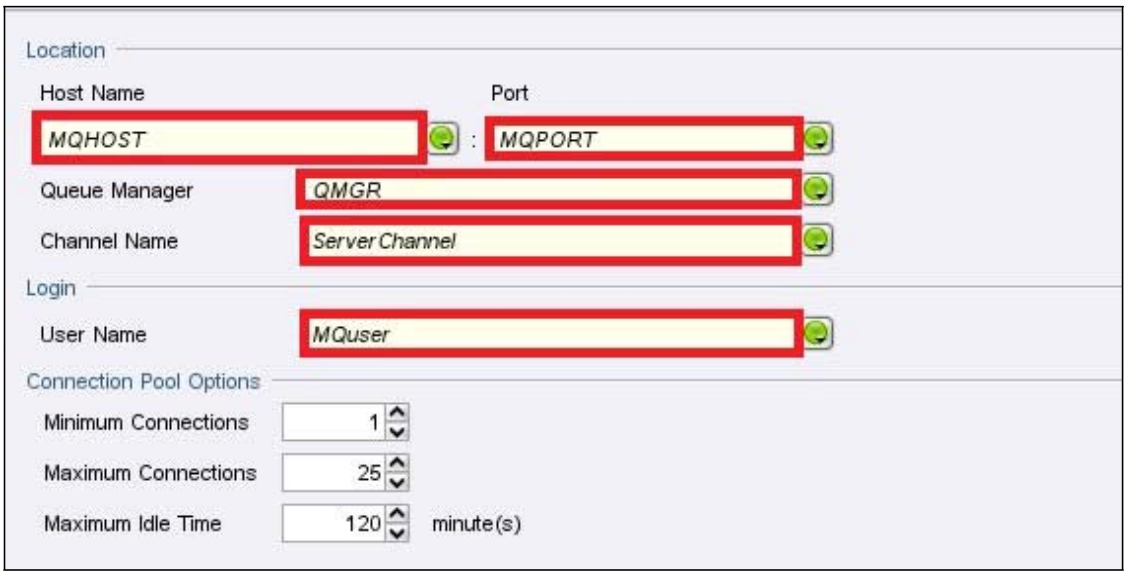
__ b. The Configuration window should appear. If it does not, then double-click new HTTP endpoint created. Ensure that **Integration Appliance Receives Request** is selected, and the **Configure for IBM App Connect on cloud** check box is ***not*** selected. Set the HTTP Receive Port to the http.listen.port property name. Use the green icon to the right to select the property.



__ 4. Create an MQ endpoint.

__ a. Right-click endpoints and select **Create Endpoint > MQ**.

__ b. The Configuration window should appear. If it does not, double-click the **MQ** endpoint to configure.



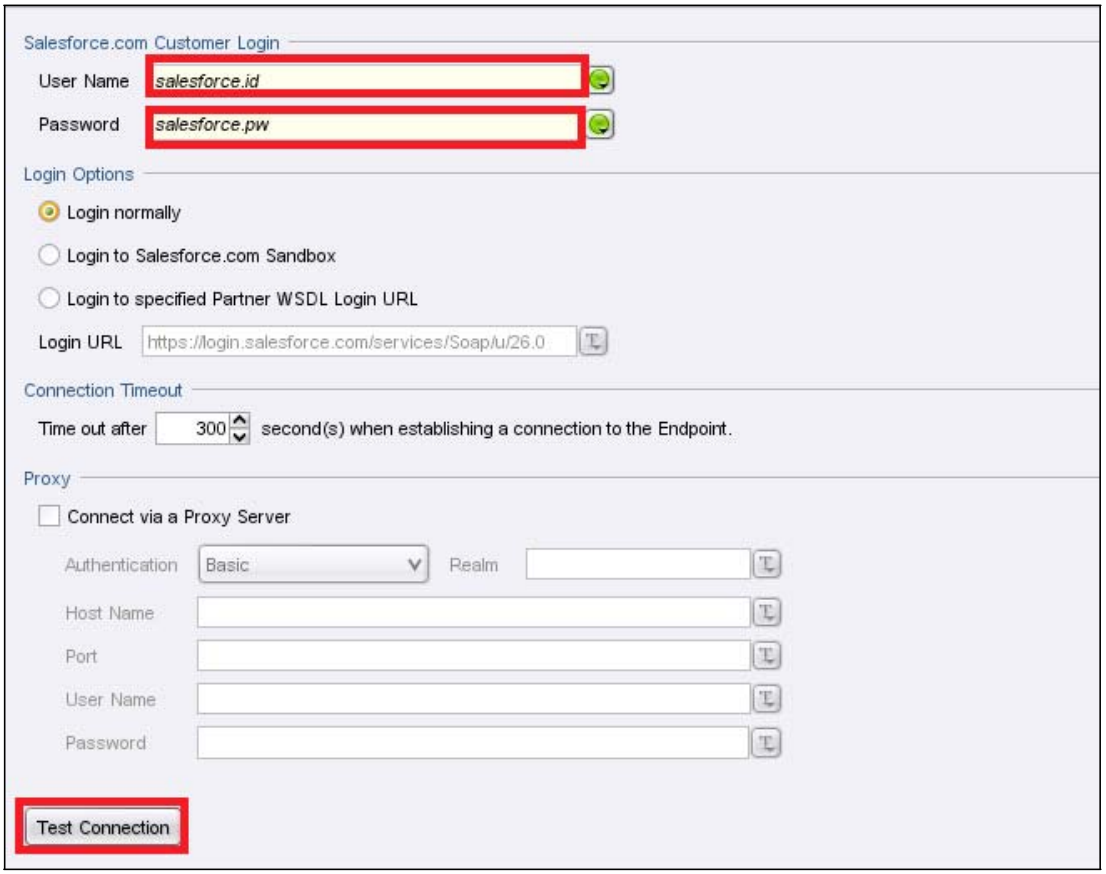
__ c. Assign the following configuration properties as highlighted in the previous image:

- Host Name = `MQHOST`
- Port = `MQPORT`
- Queue Manager = `QMGR`
- Channel Name = `ServerChannel`
- User Name = `MQuser`

__ 5. Create a **Salesforce.com** endpoint.

__ a. Right-click endpoints and select **Create Endpoint > Salesforce**.

__ b. The Configuration window should appear. If it does not, double-click the **Salesforce** endpoint to configure.

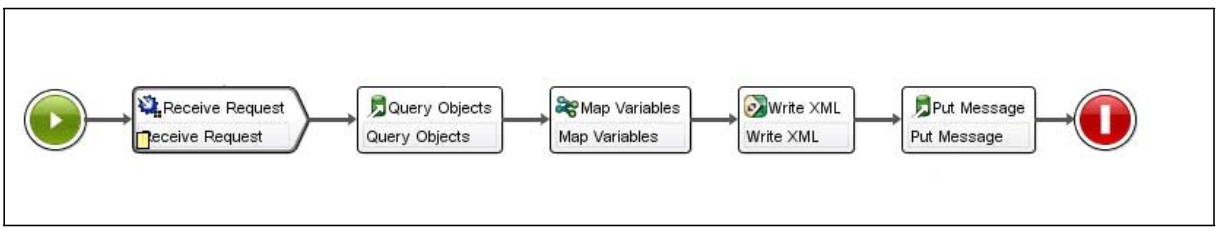


__ c. Assign the following configuration properties as highlighted in the previous image:

- User Name = `salesforce.id`
- Password = `salesforce.pw`

__ d. To test the connectivity to Salesforce, click **Test Connection**.

__ 6. Configure the orchestration by dragging and dropping activities as shown in the screen capture:



__ 7. You need the following activities, which are listed in the order that you need them:

__ a. **HTTP > ReceiveRequest**

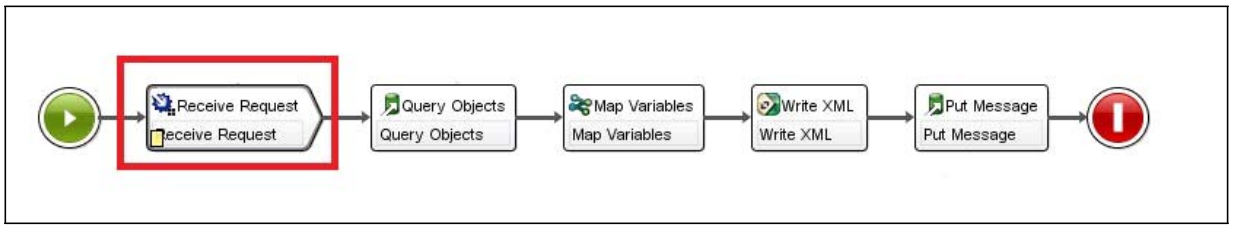
__ b. **Salesforce > QueryObjects**

__ c. **Transform > MapVariables**

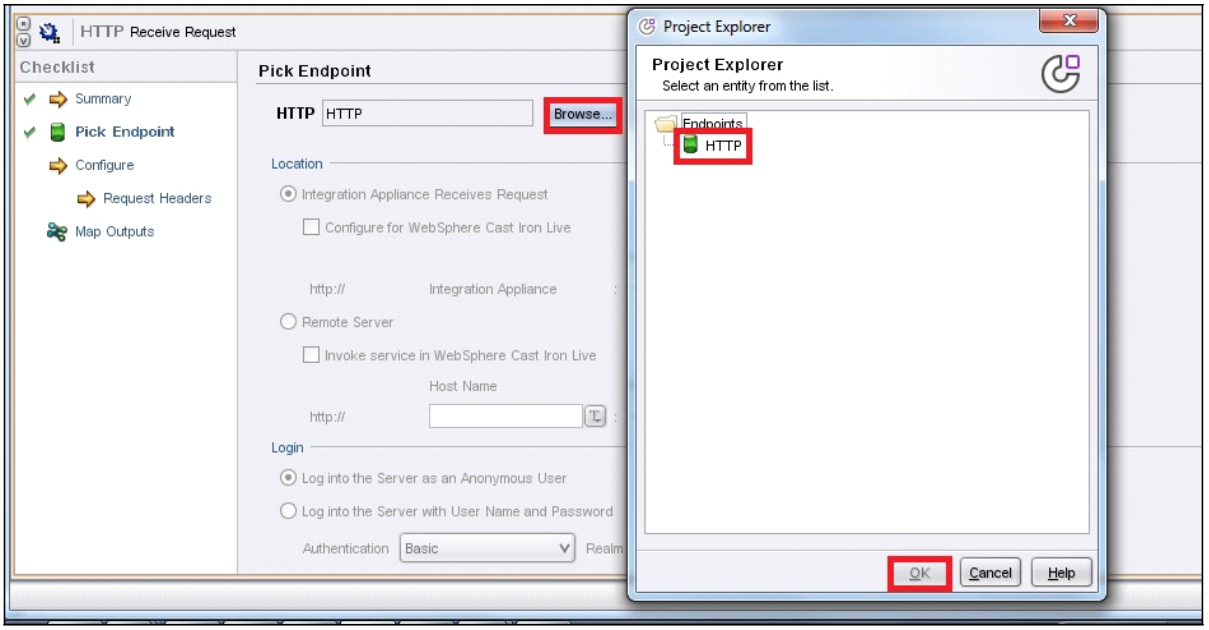
__ d. **Transform > WriteXML**

__ e. **MQ > PutMessage**

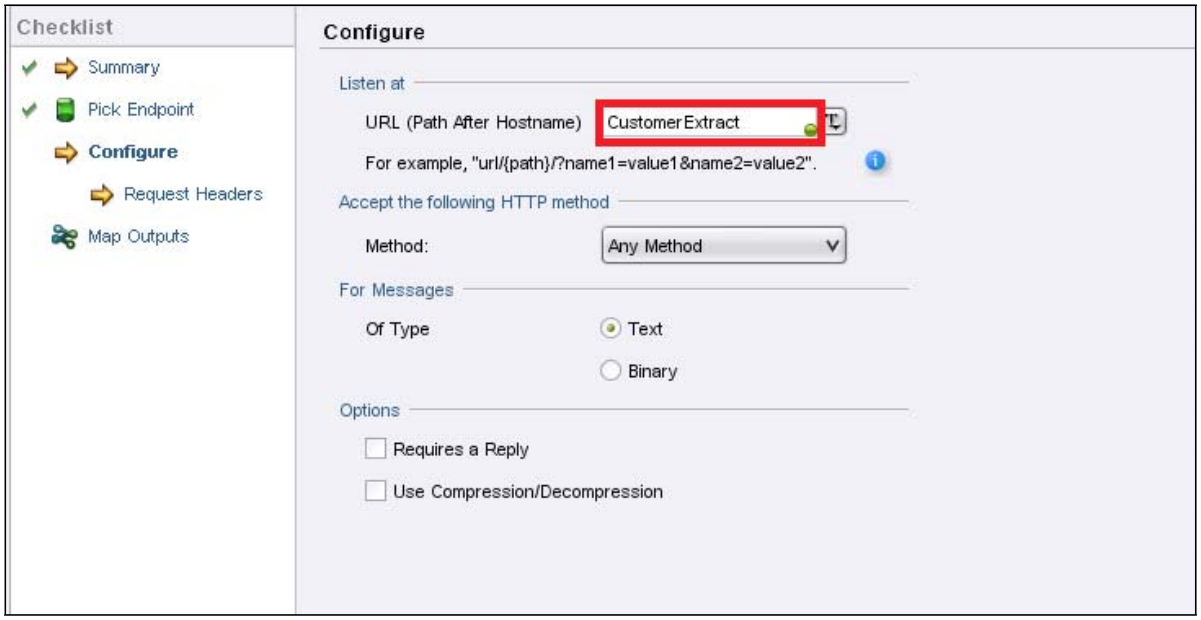
__ 8. Configure the **HTTP Receive Request** activity by clicking the activity in the orchestration.



__ a. Pick Endpoint. Select the **HTTP** endpoint by clicking **Browse**. Click **OK** to select.

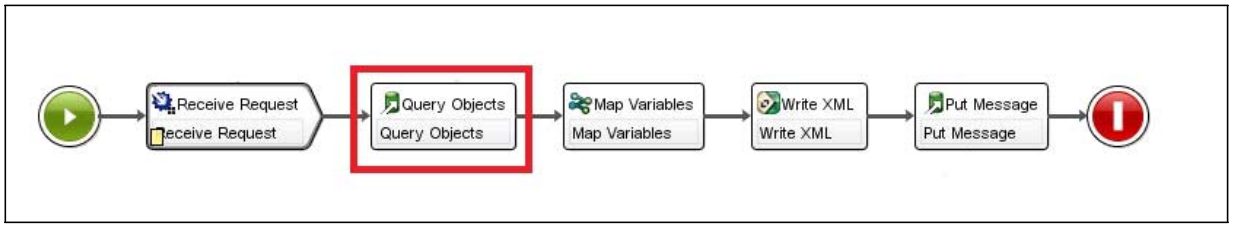


__ b. Configure. Change the URL to the hardcoded value: `Customer Extract`

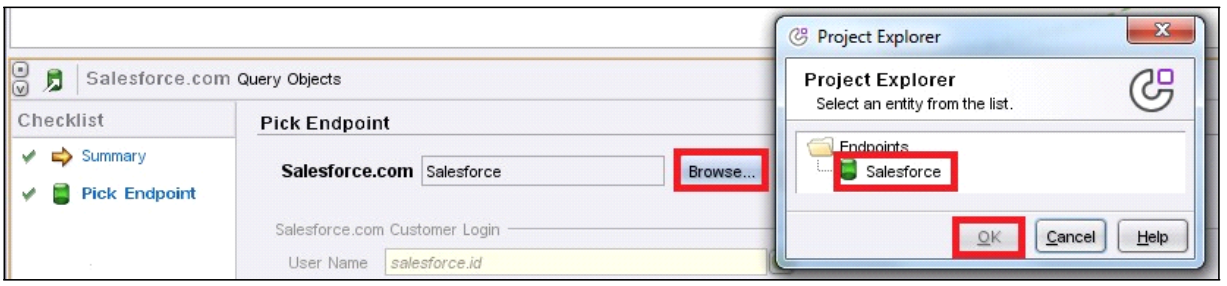


__ c. Map Outputs. No mapping is required here, as the **HTTP Receive** is set up only as a mechanism to start the orchestration.

__ 9. Configure the **Query Objects** activity by clicking the activity in the orchestration.

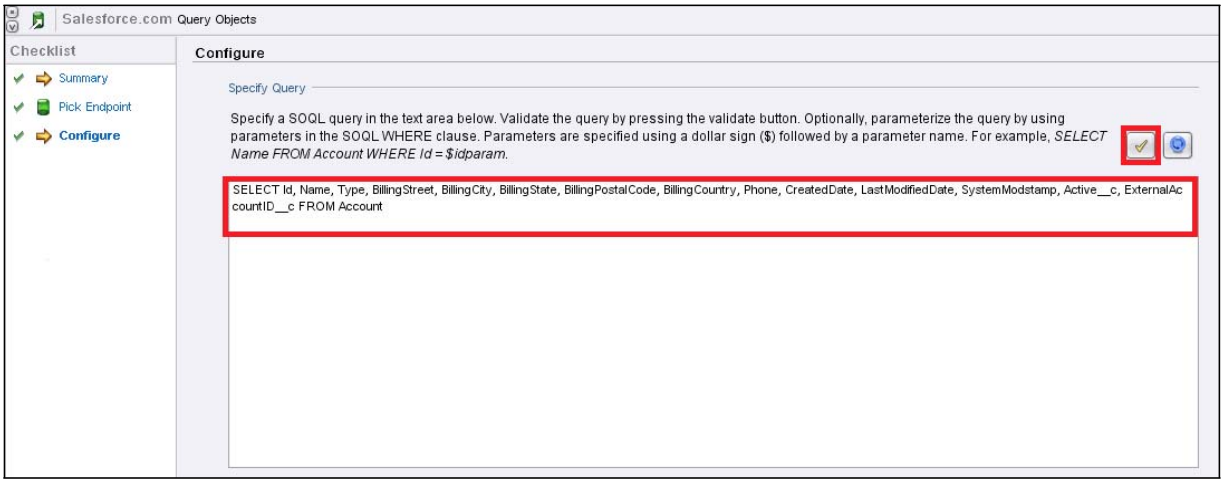


__ a. Pick Endpoint. Select the **Salesforce** endpoint by clicking **Browse**. Click **OK** to select.



__ b. Configure. Enter in the SOQL query in the white space. The SOQL Query to use for this lab is provided here and also in the directory `c:\Student\LabResources\Lab4\soql.txt`. Click the check mark to confirm syntax and objects that were entered in:

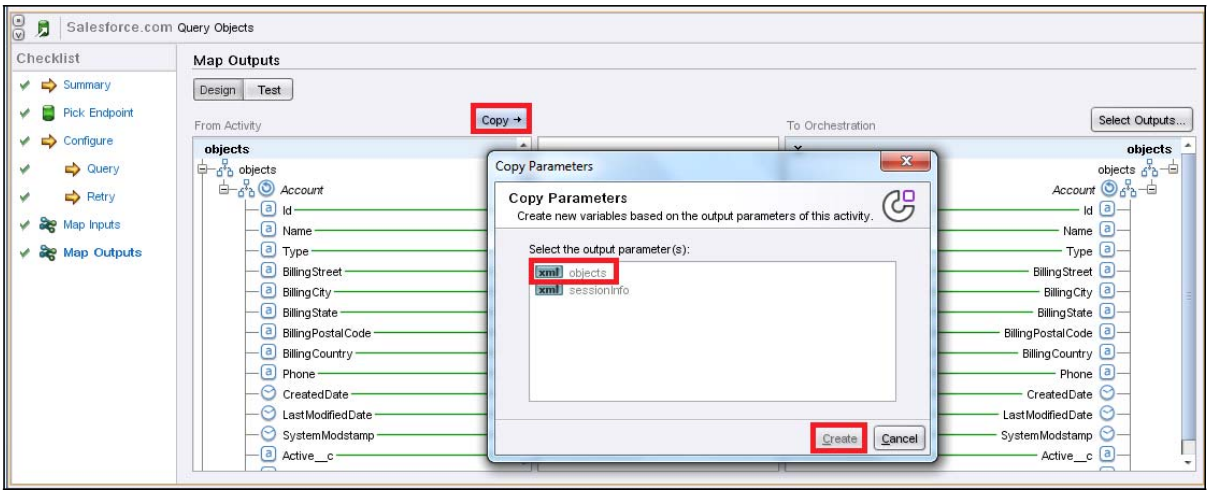
```
SELECT Id, Name, Type, BillingStreet, BillingCity, BillingState,
BillingPostalCode, BillingCountry, Phone, CreatedDate, LastModifiedDate,
SystemModstamp, Active__c, ExternalAccountID__c FROM Account
```



__ c. Query and Retry. Skip these steps.

__ d. MapInputs. If your SOQL query had any input parameters to map into the query, you would handle that here. In this Lab, this step can be skipped.

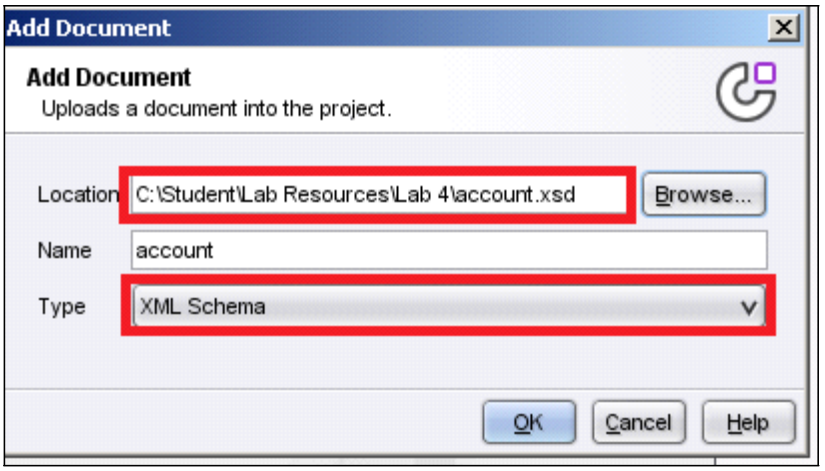
__ e. MapOutputs. Click **Copy** to initialize a new variable that stores all the data that is retrieved from the query. Be sure to select the `objects` variable and then click **Create**.



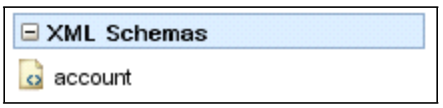
__ 10. Before you can continue with the next activity, you need to add the XML schema that the data set is going to generate. To do so, follow these steps:

__ a. Click the **Project** tab.

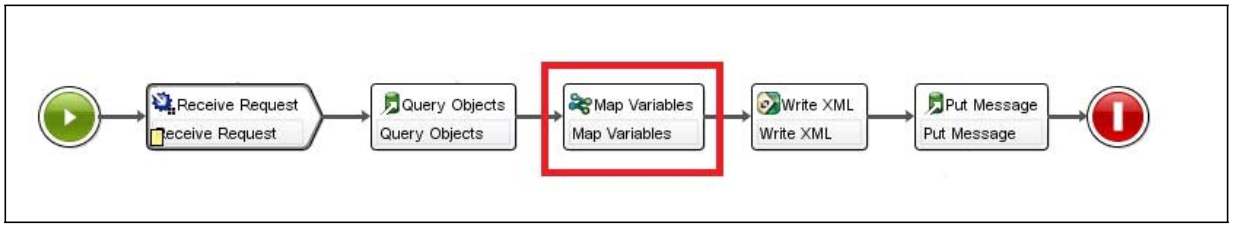
__ b. Right-click the **XML Schemas** heading and select **Add Document.** Browse to the `c:\Student\LabResources\Lab4` directory and select the `account.xsd` file. Click **OK** to add the document to the project.



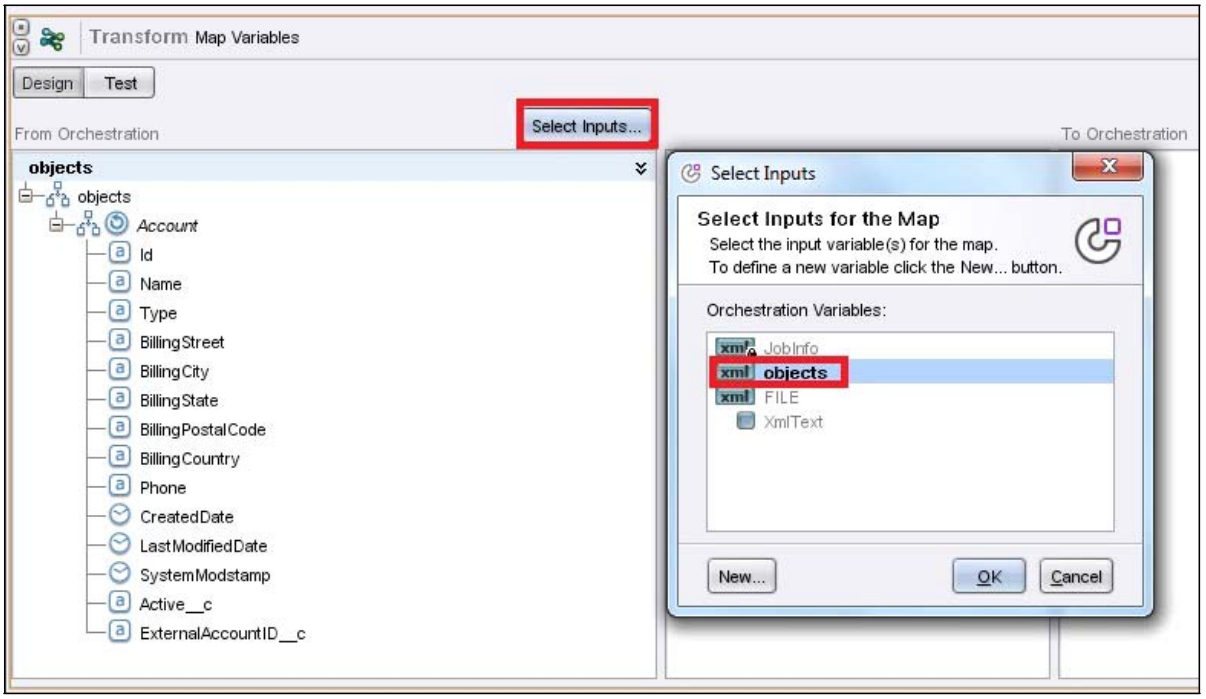
__ c. As soon as the schema is added, you see it under the schema tab.



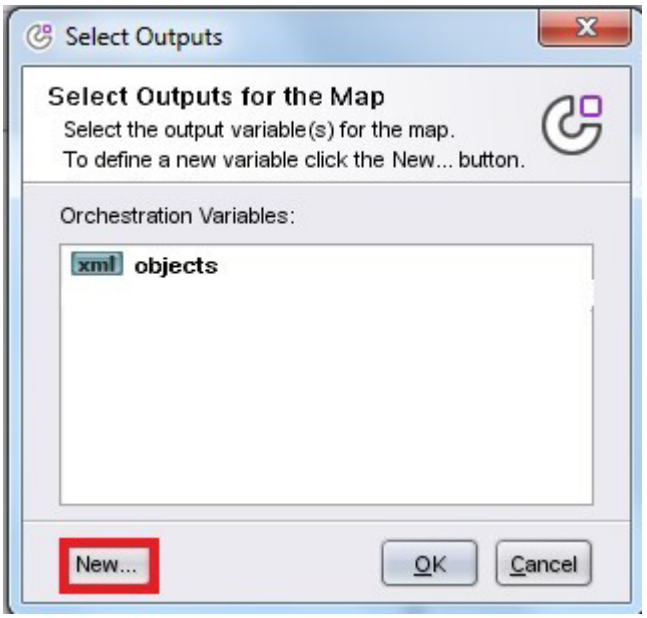
__ 11. To configure the **Map Variables** activity, click the activity in the orchestration.



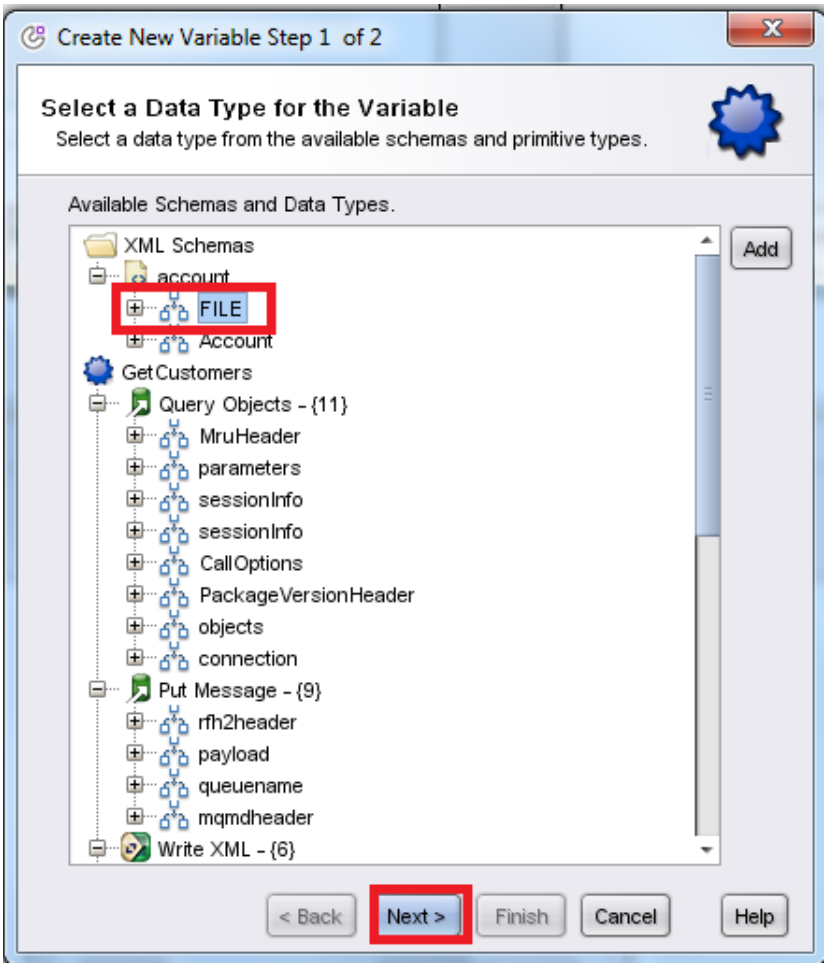
__ a. This activity has no checklist. Map Variables is an optional activity that you can use to segregate transformation from other activities for ease of maintenance. All of the mapping that is needed for this orchestration is sequestered here, which makes the subsequent activities much quicker to configure. Click **Select Inputs**. Add the `objects` variable and click **OK**, which provides the result of the Salesforce Query that acts as the input.



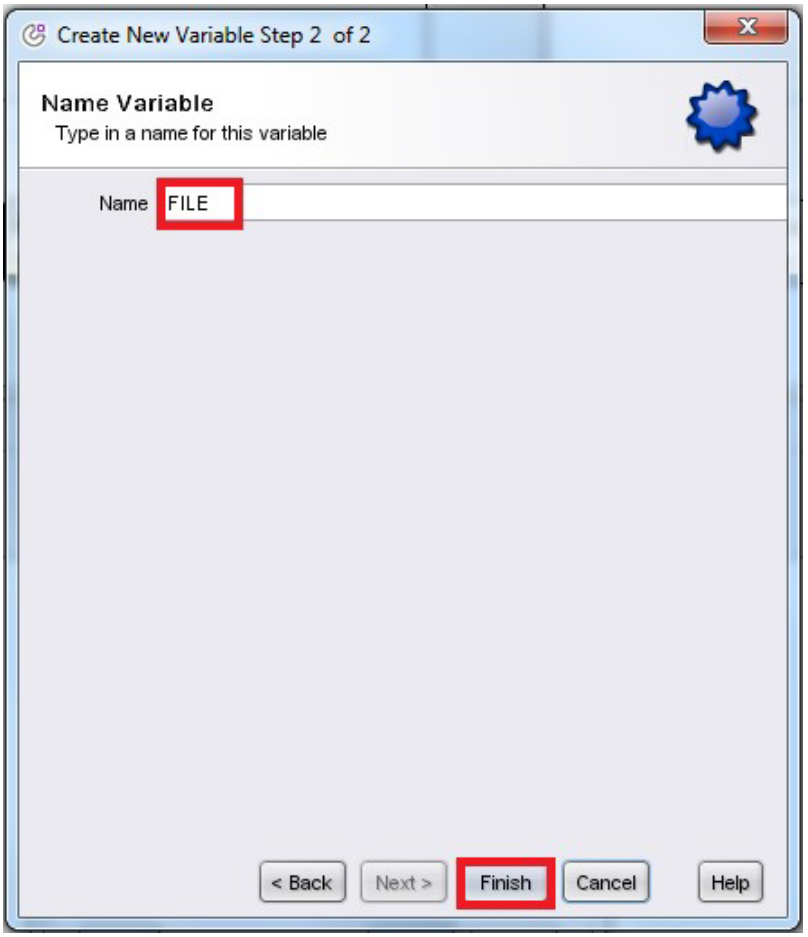
__ b. Click **Select Outputs**. Click **New** and then click **OK**. Now you can initialize a new variable to represent the XML schema for the output data that you are going to send to the ESB.

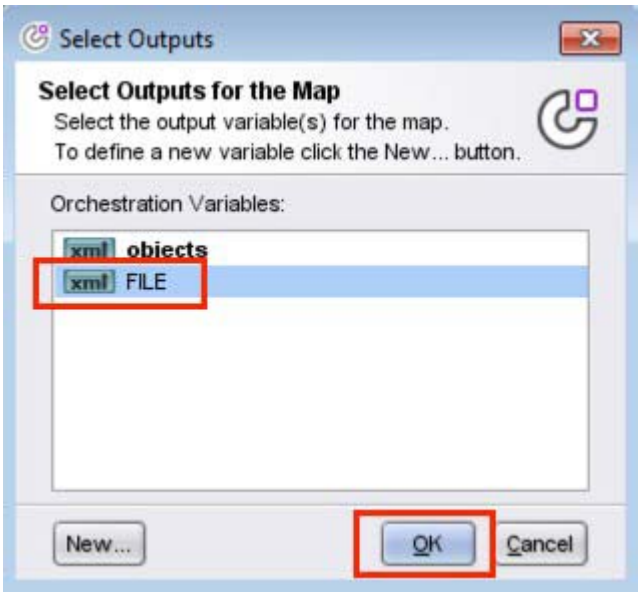


__ c. Select the **FILE** node and then click **Next**.



__ d. Name the node `File` and then click **Finish**.

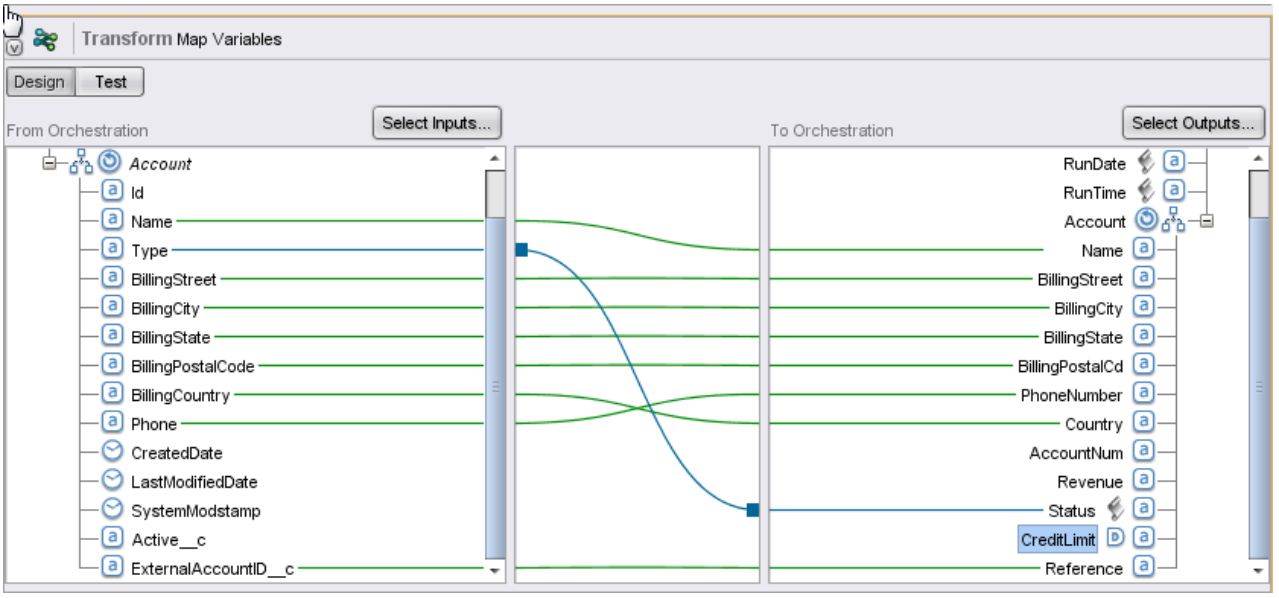




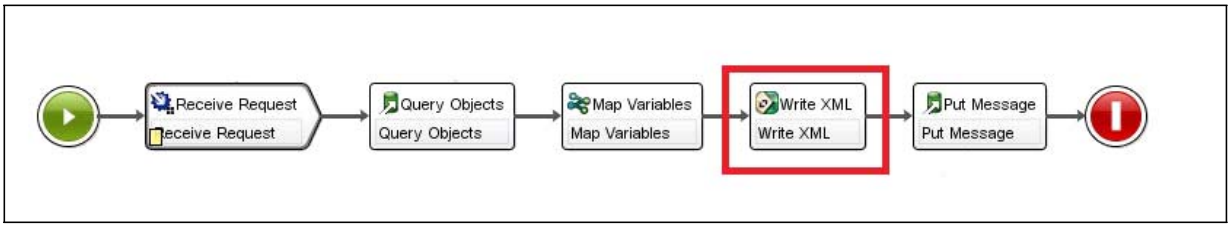
__ e. Fields now can be mapped. Use the following mapping matrix as your guide. Use the Substring After function for the transformation of the “Type” field.

Input Field	Output Field	Transformation Rules
Id	n/a	Not Mapped
Name	Name	Direct mapping, no transformation
Type	Status	Parse (Substring) Input field to extract all data after "-"
BillingStreet	BillingStreet	Direct mapping, no transformation
BillingCity	BillingCity	Direct mapping, no transformation
BillingState	BillingState	Direct mapping, no transformation
BillingPostalCode	BillingPostalCd	Direct mapping, no transformation
BillingCountry	Country	Direct mapping, no transformation
Phone	PhoneNumber	Direct mapping, no transformation
CreatedDate	n/a	Not Mapped
LastModifiedDate	n/a	Not Mapped
SystemTimeStamp	n/a	Not Mapped
Active__c	n/a	Not Mapped
ExternalAcountID__c	Reference	Direct mapping, no transformation
	RunDate	Current Date
	RunTime	Current Time
	CreditLimit	Default Value of 10000
	Revenue	Not Mapped

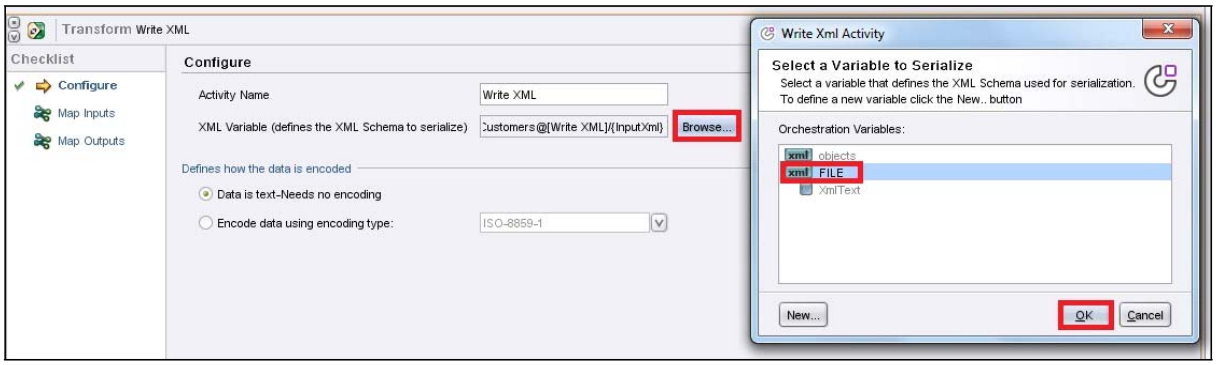
__ f. When complete, the mapping should look as follows:



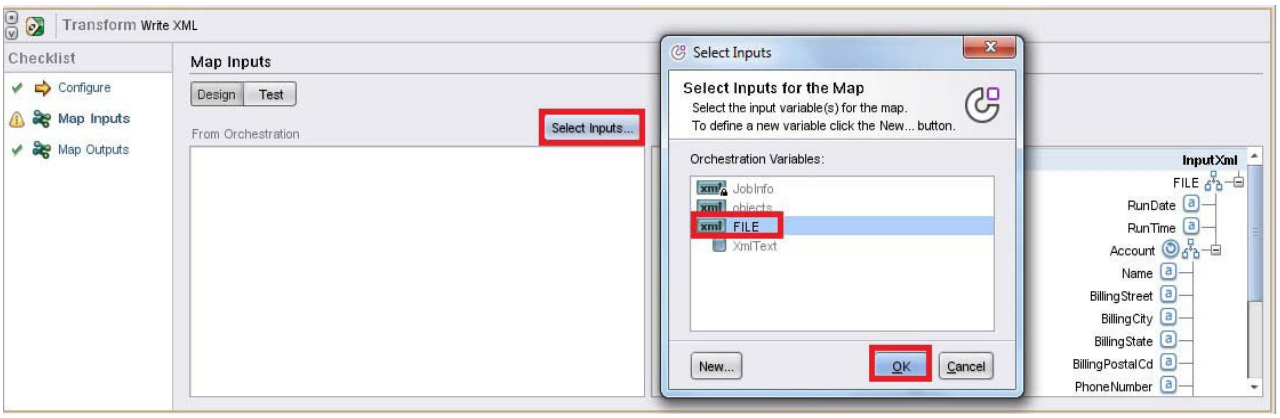
__ 12. Configure the **Write XML** activity by click activity in the orchestration.



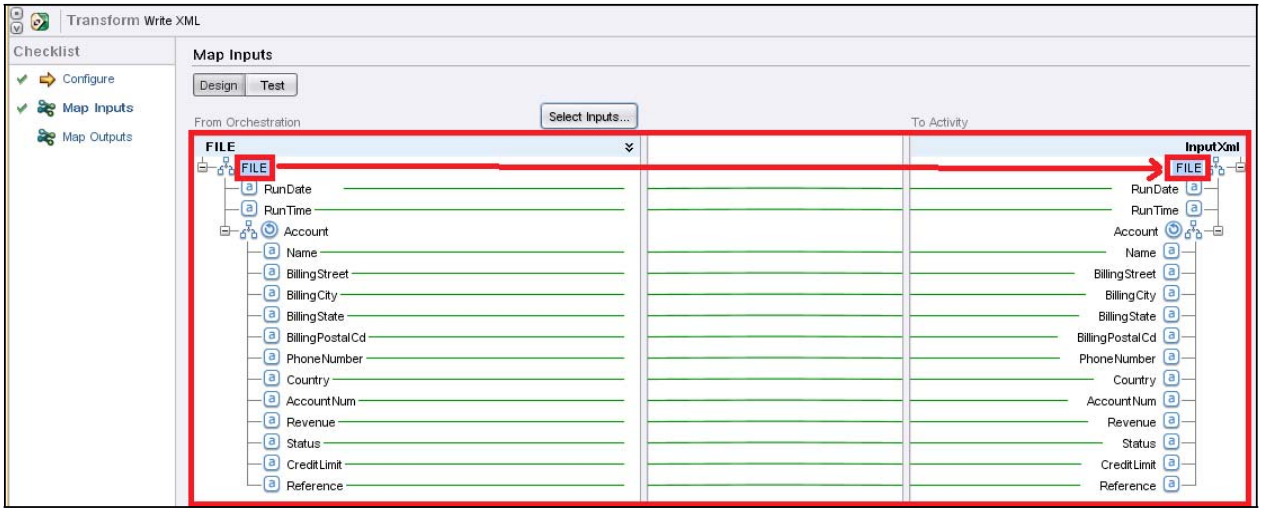
__ a. Configure. Click **Browse** next to the **XML Variable** field to select the schema to use. Select the **FILE** variable. Click **OK** to complete the task.



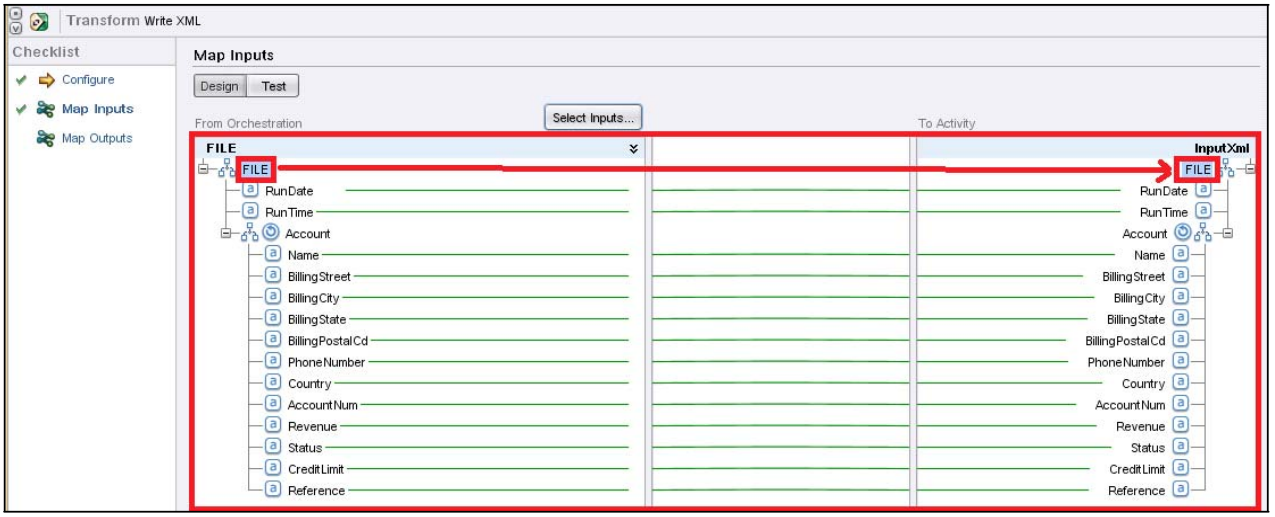
__ b. Map Inputs. Click **Select Inputs** and then browse for the **FILE** variable. Click **OK**.



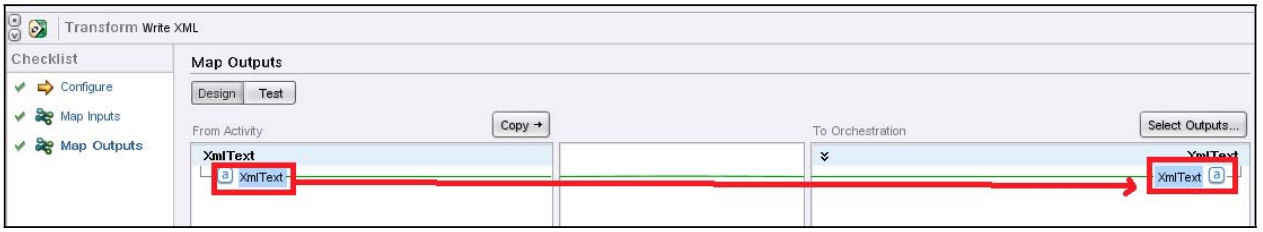
__ c. Drag the **FILE** node from the input side over to the **FILE** node on the output side. The fields are going to be automapped.



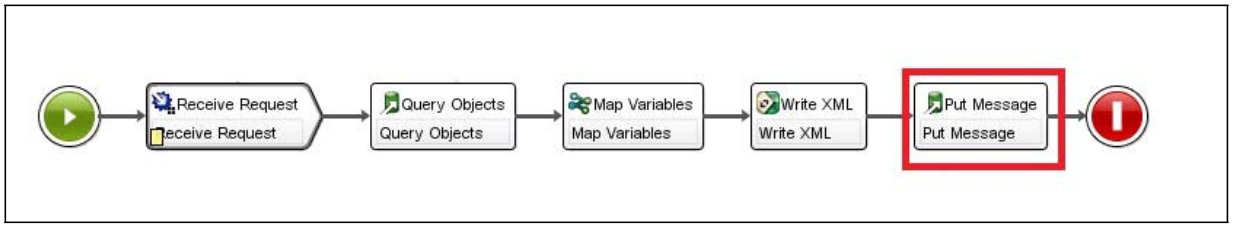
__ d. Map Outputs. Copy to the variable `XmlText` by clicking **Copy**, selecting the `XmlText` variable, and then clicking **OK**. The XML that is mapped in the previous step is serialized into a single “blob”, which creates a payload in a single field that you are going to use in the next activity.



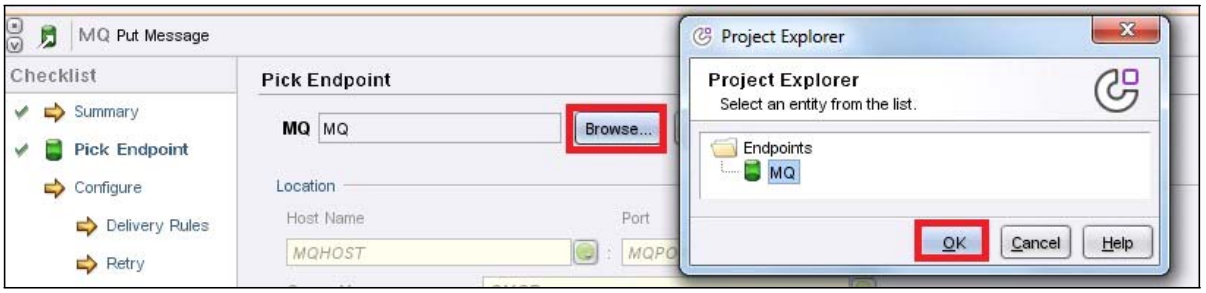
__ e. Drag the **XmlText** field from the input over to the output.



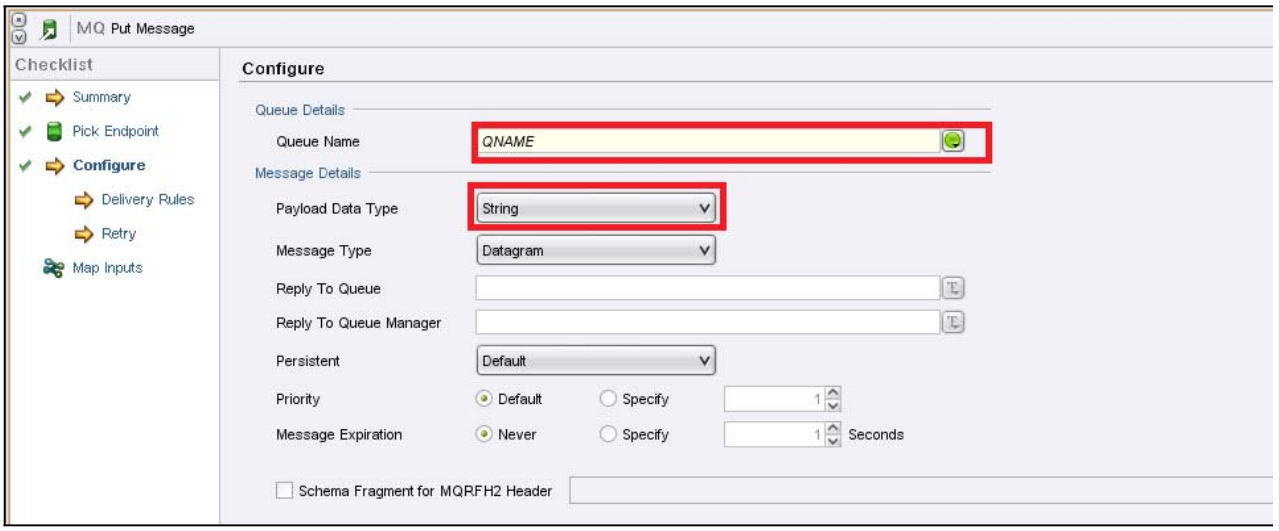
__ 13. Configure the **Put Messages** activity by clicking the activity in the orchestration.



__ a. Pick Endpoint: Select the **MQ** endpoint by clicking **Browse**. Click **OK** to complete.



__ b. Configure: Assign the QNAME configuration property variable to **Queue Name**.

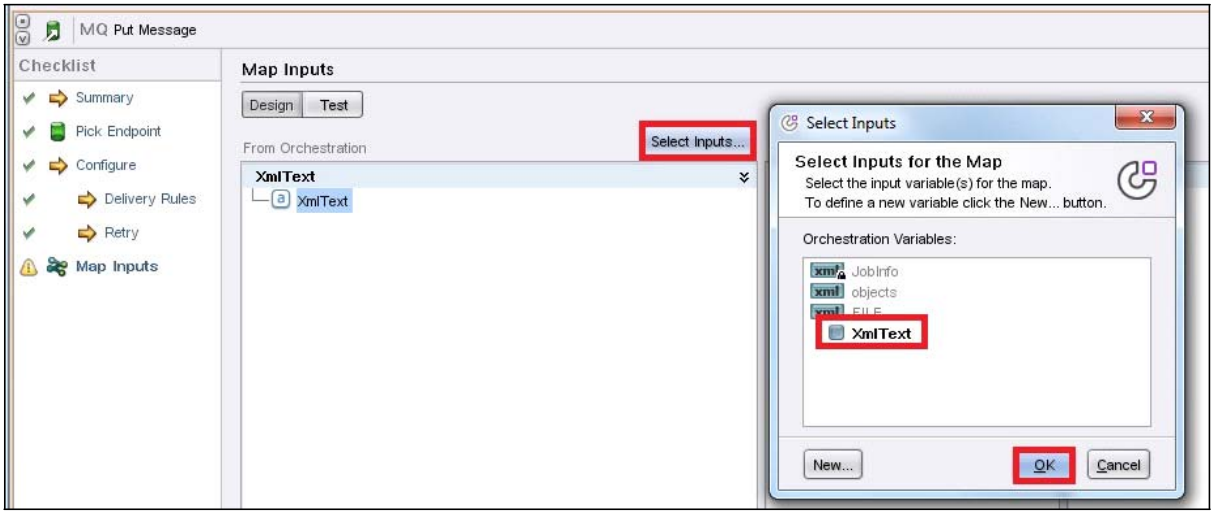


__ c. Ensure that the **PayloadDataType** is set to **String**.

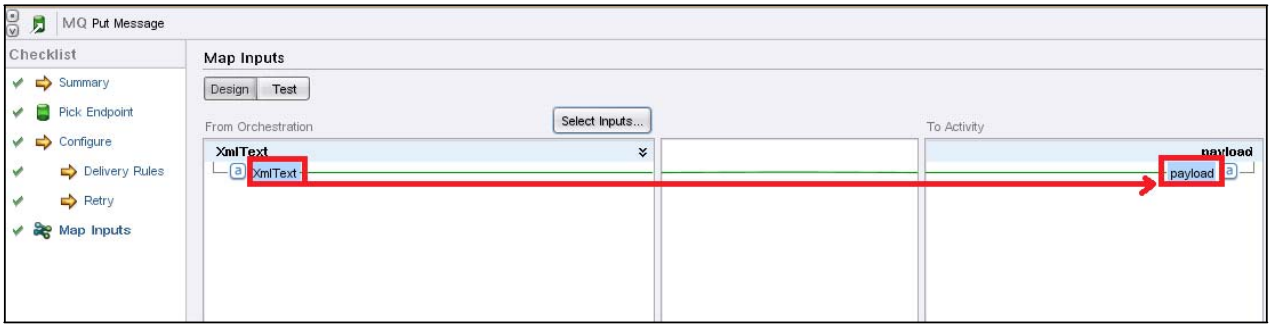
__ d. Delivery Rules: No changes.

__ e. Retry: No changes.

__ f. Map Inputs. Add the **XmlText** variable to the left side by clicking **Select Inputs**. Click **OK**.

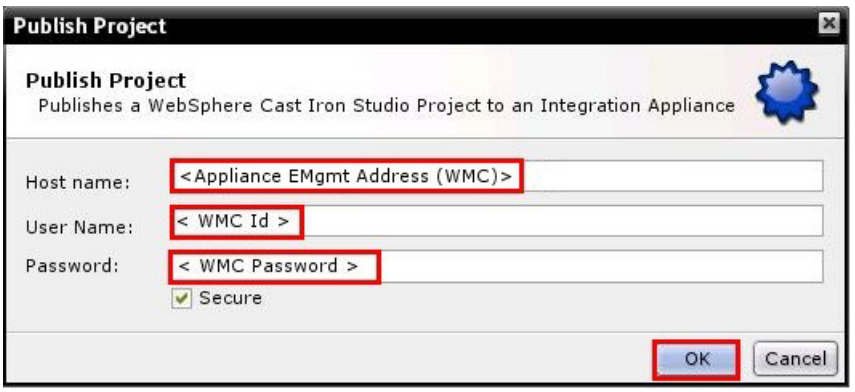


__ g. Map over the **XmlText** node from the left side to the **payload** node on the right side.

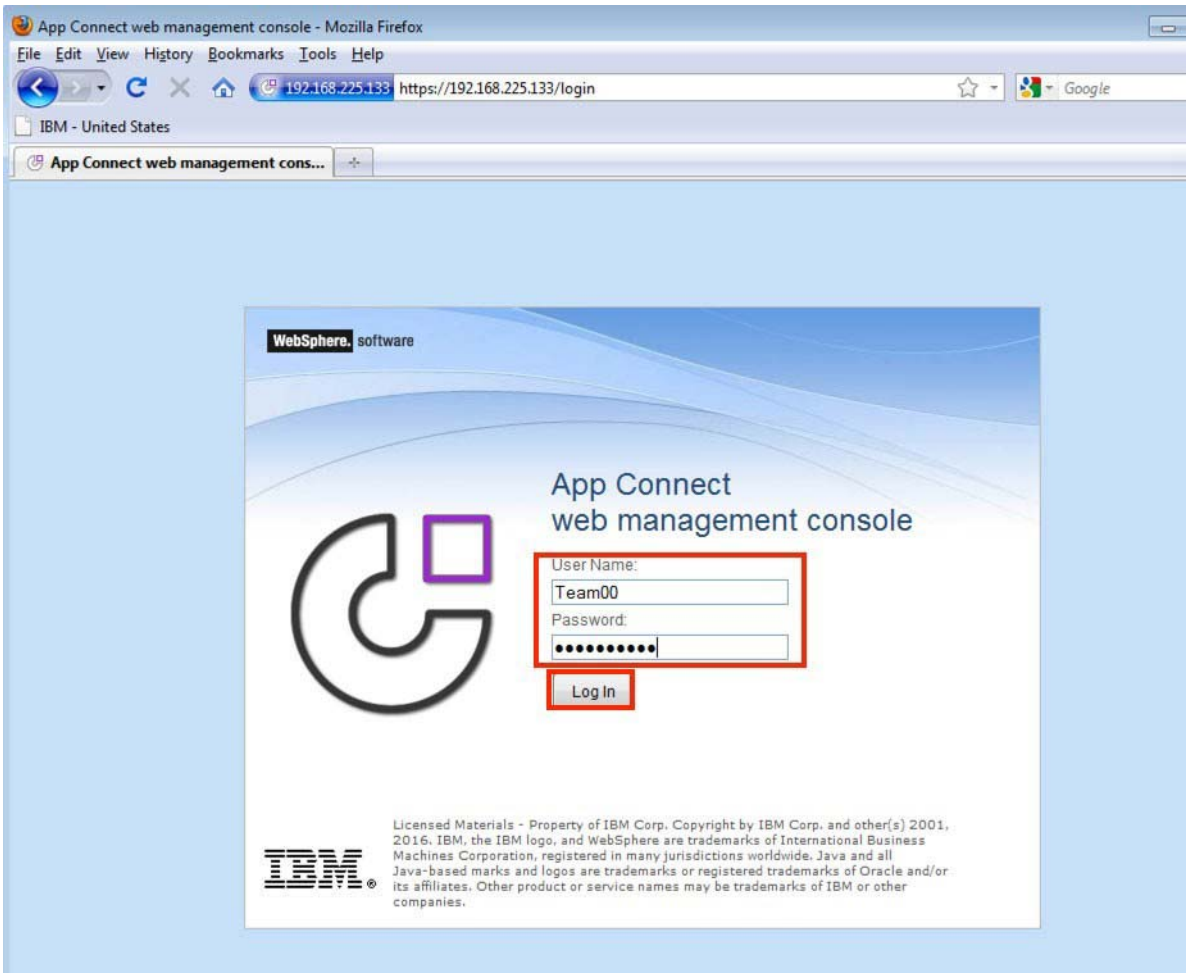


__ 14. Deploy the orchestration to your runtime environment.

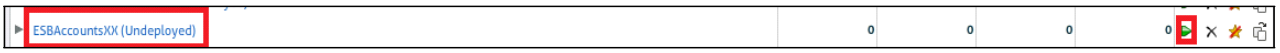
__ 15. Deploy the project to your appliance with **File > Publish** or by using the icon from the top toolbar. Use the credentials from the previous lab.



__ 16. Log in into the WMC with the URL and login credentials that the instructor provides (use the WMC admin ID and password). Click **Login**.

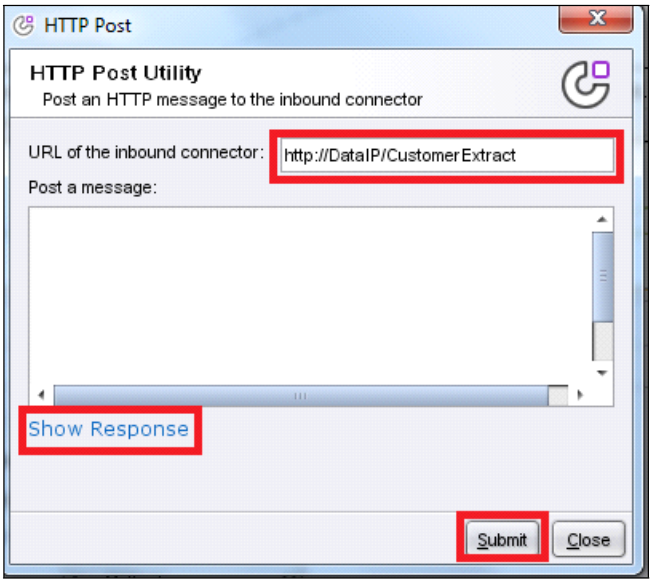


__ 17. Find your project on the list and double-click the green arrow on the right side to enable it.



4.2. Run the Load Salesforce.com Accounts to ESB by using IBM MQ Lab

__ 1. In the App Connect Studio, use the HTTP Post utility to start the orchestration. Click **Tools > HTTPPostUtility**. Complete the values on the form according to the value of the Data IP Port on your appliance. See the Connection Parameters spreadsheet for the appropriate information. Before clicking **Submit**, click the **Show Response** link to see the response from the Appliance after the request is submitted.

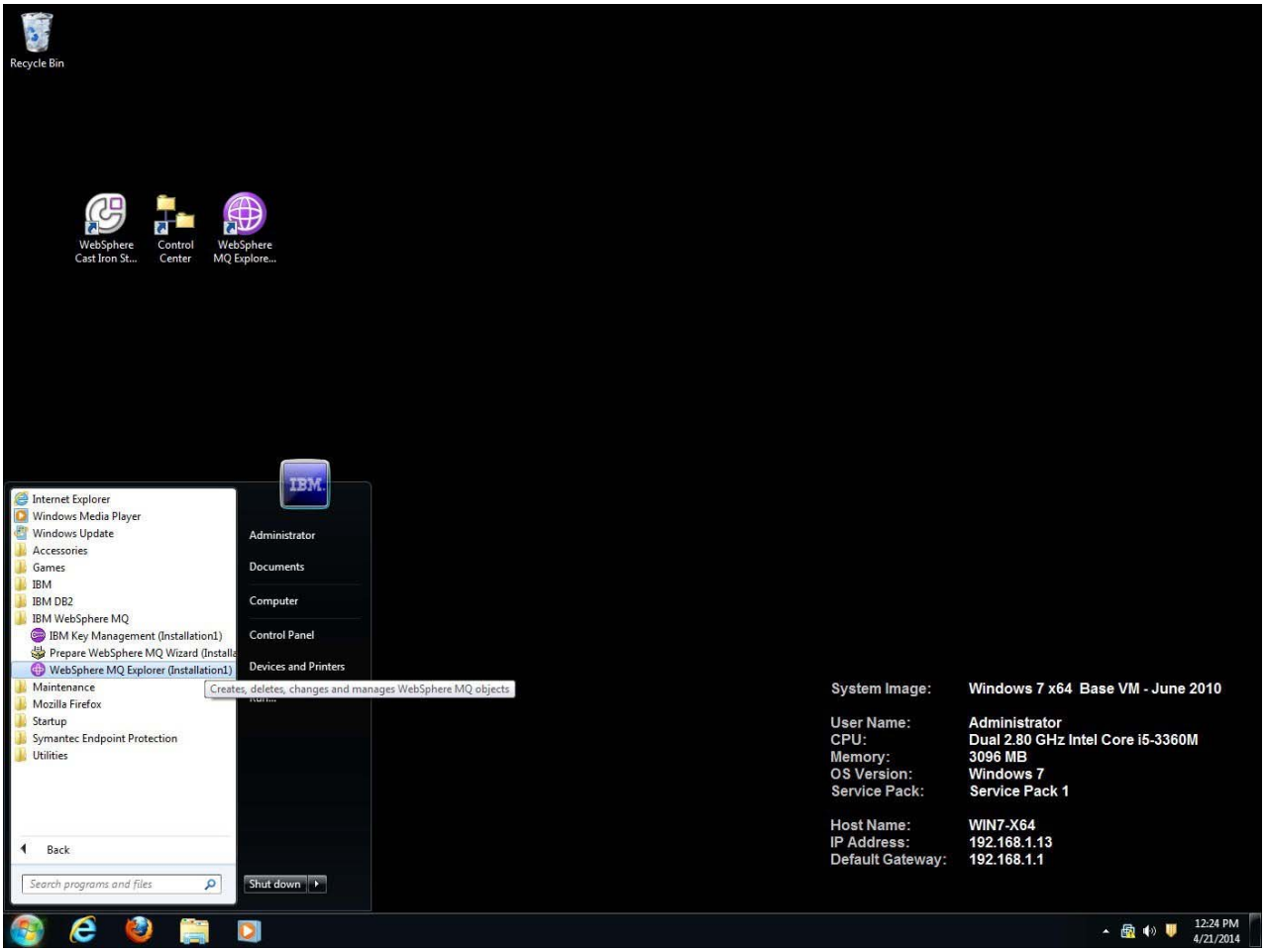


__ 2. If it runs properly, you see the following message in the response section: `Message accepted by the Integration Appliance.`

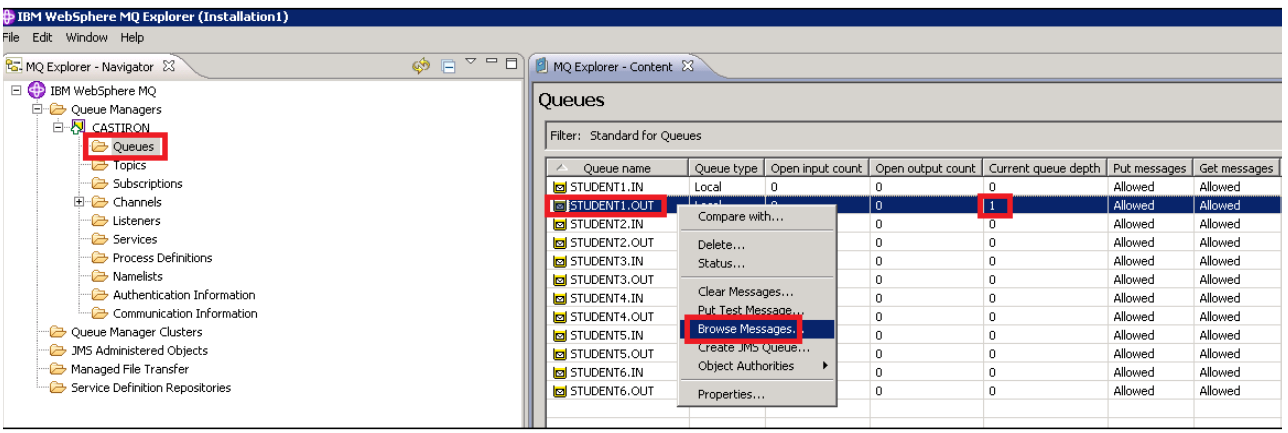
__ 3. To confirm that the output is generated properly, first check in the Web Management Console.

__ 4. Optionally, you can check inside of IBM MQ to ensure that the data was enqueued properly.

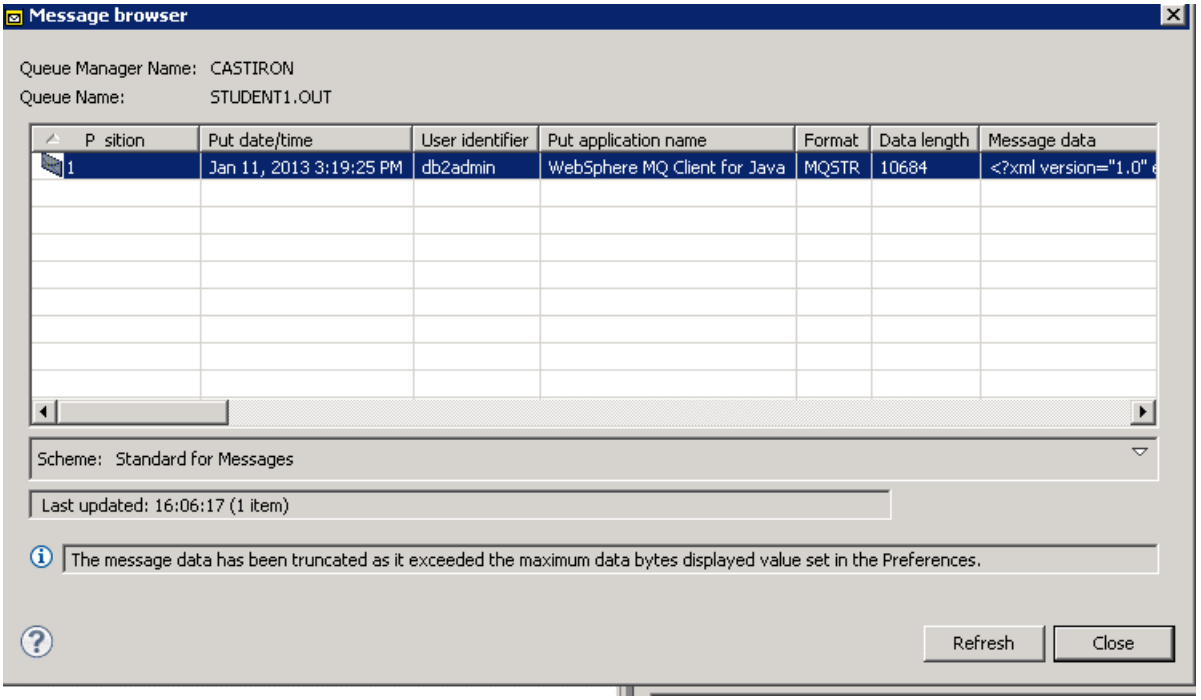
__ a. You can either click IBM MQ Explorer on the desktop or go to **Start > All Programs > IBM IBM MQ > IBM MQ Explorer**.



__ 5. Inside MQ Explorer, click **Queues**, and then select the **STUDENT1.OUT** queue. The queue depth should be 1. Right-click the **STUDENT1.OUT** queue and select **Browse Messages**.



__ 6. The approximate message size and data are visible here.



End of exercise

Exercise review and wrap-up

This lab demonstrated how App Connect Professional can be used to extract information from a SaaS-based application and format messages that can be passed along to an ESB backbone.

Exercise 5. Invoking web services

Estimated time

00:45

Overview

In this exercise, you create a project, webServicesBasics, which has a web service.

Objectives

After completing this exercise, you should be able to:

- Configure an HTTP Receive request
- Configure an HTTP Send request
- Configure a Web Service Invoke Service
- Configure a Transform Read XML

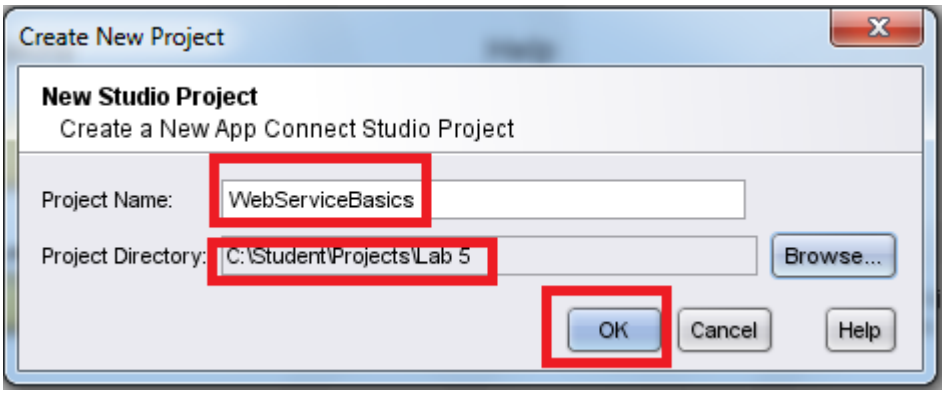
Introduction

Web services are described in WSDL files that are used to configure a consuming web service connector. Then, the **Web Services Invoke Service** activity can be used to invoke the web service and interpret its response. A web service operation is defined in terms of the XML documents that represent input and output of the operation. Often, these documents are simple one-element documents where the element contains a serialized XML document.

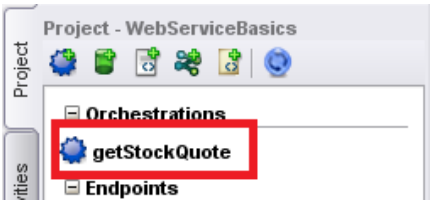
Exercise instructions

5.1. Invoke a web service with a fixed symbol

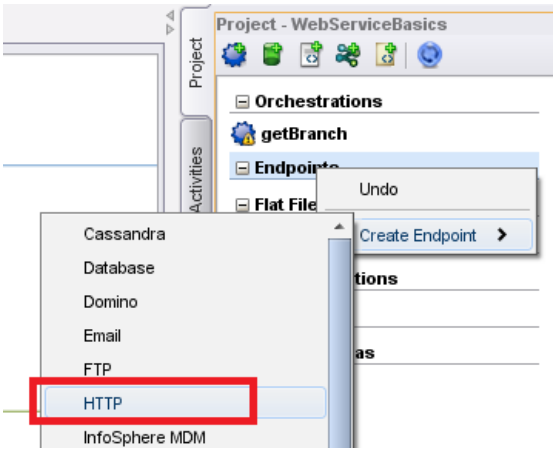
__ 1. Create a project `webServiceBasics` and save it in `C\:Student\Projects\Lab5`.



__ 2. Rename the orchestration to: `getStockQuote`

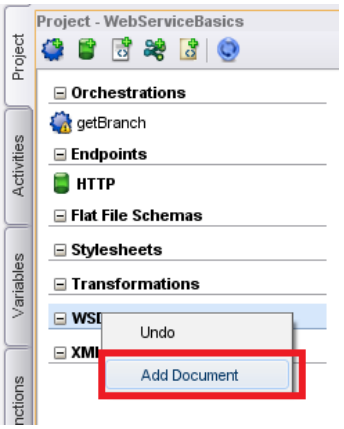


__ 3. Create an inbound HTTP endpoint and take the defaults.

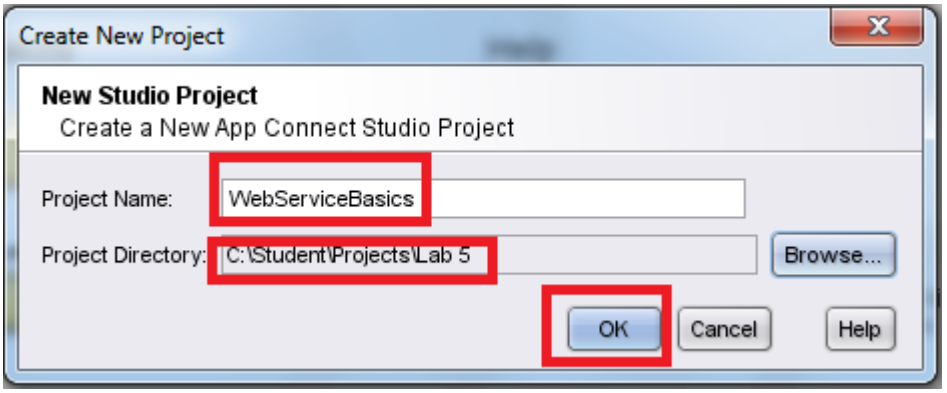


__ 4. The WSDL that describes the service needs to be loaded into the project.

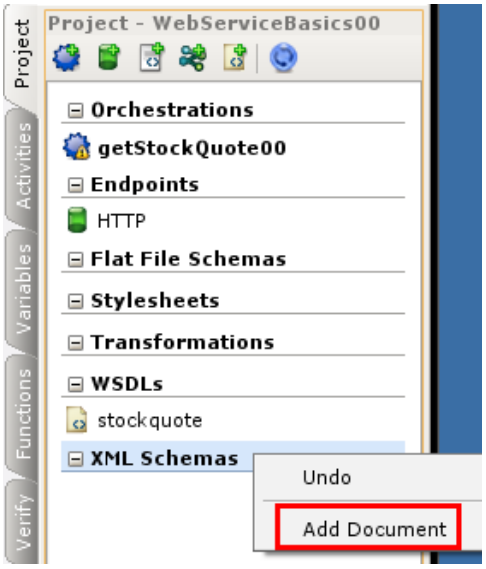
__ a. Right-click **WSDLs** and select **Add Document**.



__ b. Browse for the **`stockquote.wsdl`** file in `C:\Student\LabResources\Lab5`.

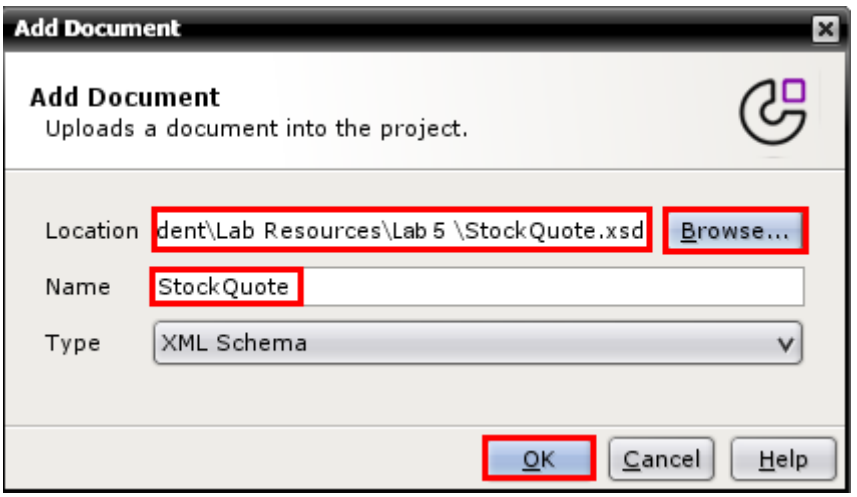


__ 5. The schema that describes the output also needs to be loaded.

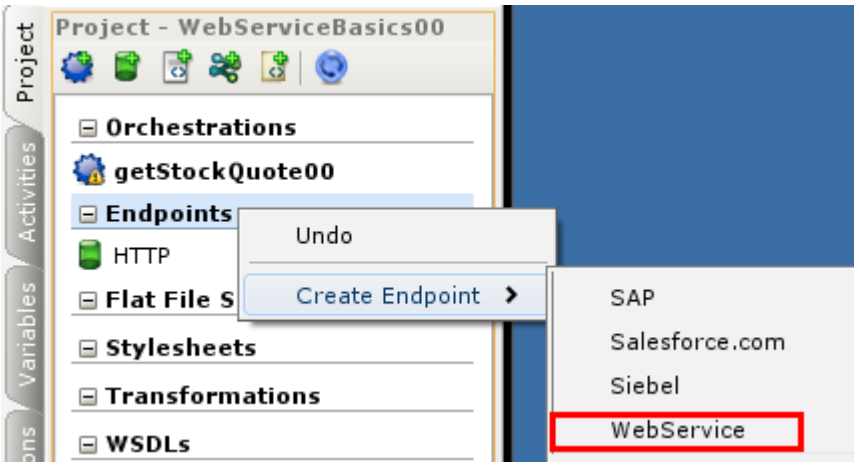


__ a. Open the menu for the XML Schemas section of the Project tab and select **Add Document**.

__ b. Browse for the `StockQuote.xsd` file in `C:\Student\LabResources\Lab5`.

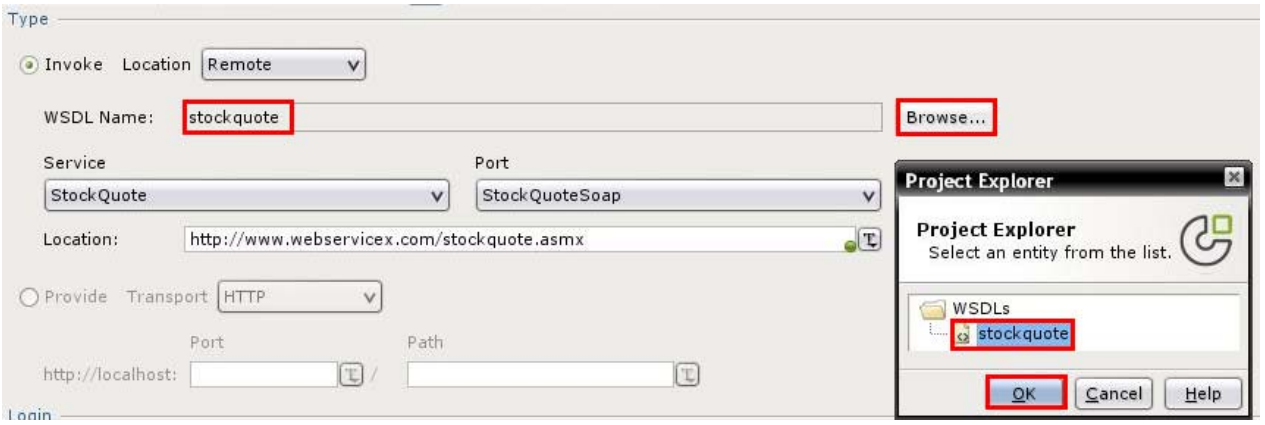


__ 6. Create a web service endpoint: `WSCStockQuote`
The service takes a securities symbol as input and returns data as an XML document.

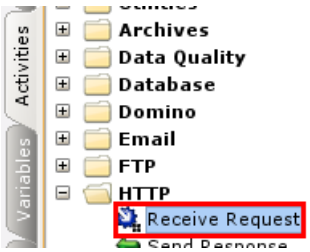


__ 7. Browse for the WSDL file that was just loaded.

__ a. The Location, Port, and Service fields are generated from the WSDL.

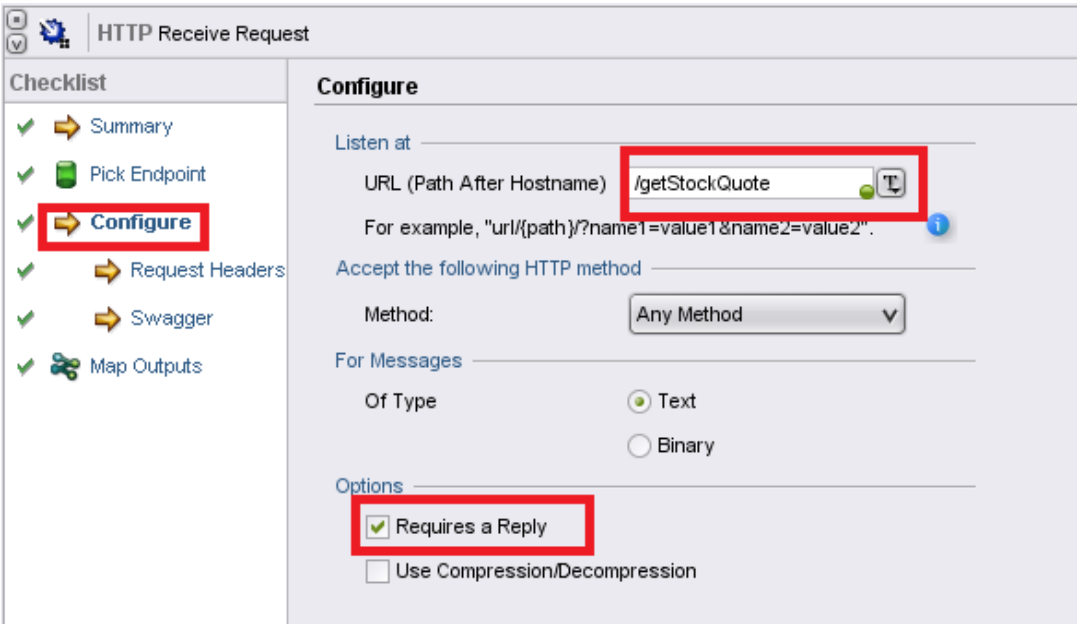


__ 8. Add an HTTP **Receive Request** activity to the orchestration.



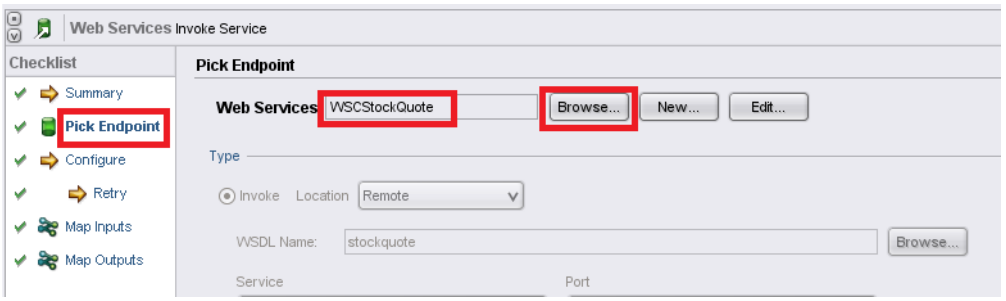
__ 9. Pick Endpoint: Browse for the HTTP endpoint that was created in step 2.

__ 10. Configure: URL = `getStockQuote` and ensure that the **Requires a Reply** check box is selected.



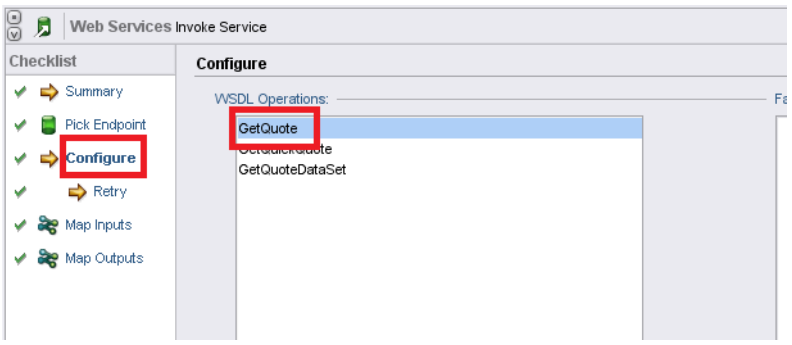
__ 11. Add a web services **Invoke Service** activity to the orchestration.

__ 12. Pick Endpoint: Select the web services endpoint **WSCStockQuote**.



__ 13. Configure: Select the operation GetQuote in the Operations pane.

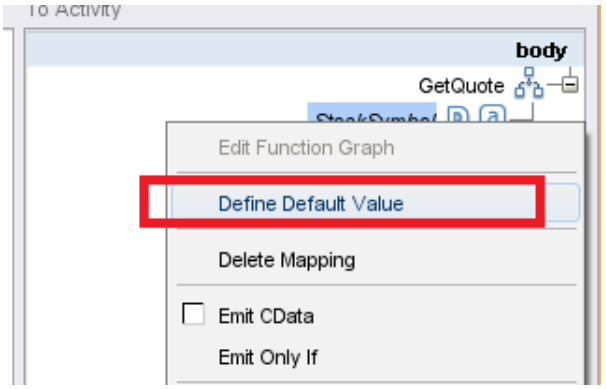
__ a. A WSDL might have only one operation, but for more complex services, all of the operations are listed.

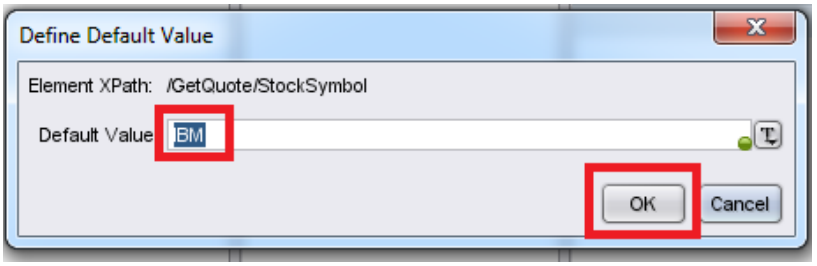


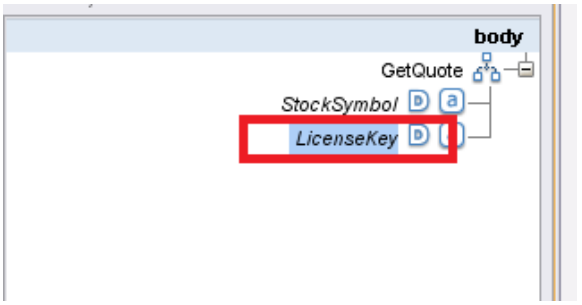
__ b. When you select the operation, the next Map steps reflect the operation's structure.

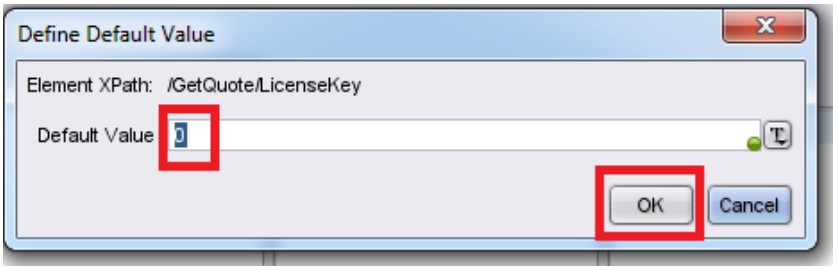
__ 14. Map Inputs: For now, you have no dynamic input.

__ a. Assign a default value to the **StockSymbol** field, such as `IBM`, by right-clicking **Stock Symbol** and selecting **Define Default Value**. Then, click **OK**. Repeat the process and assign a Default Value for **LicenseKey** of `0`. Click **OK**.

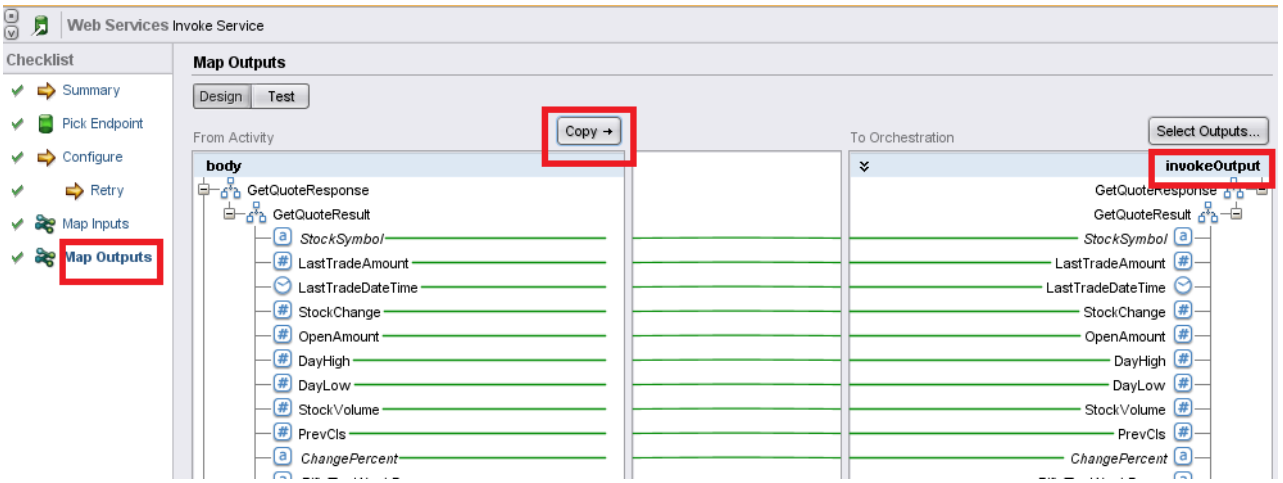




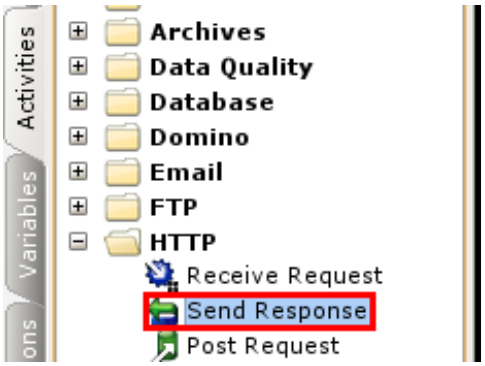




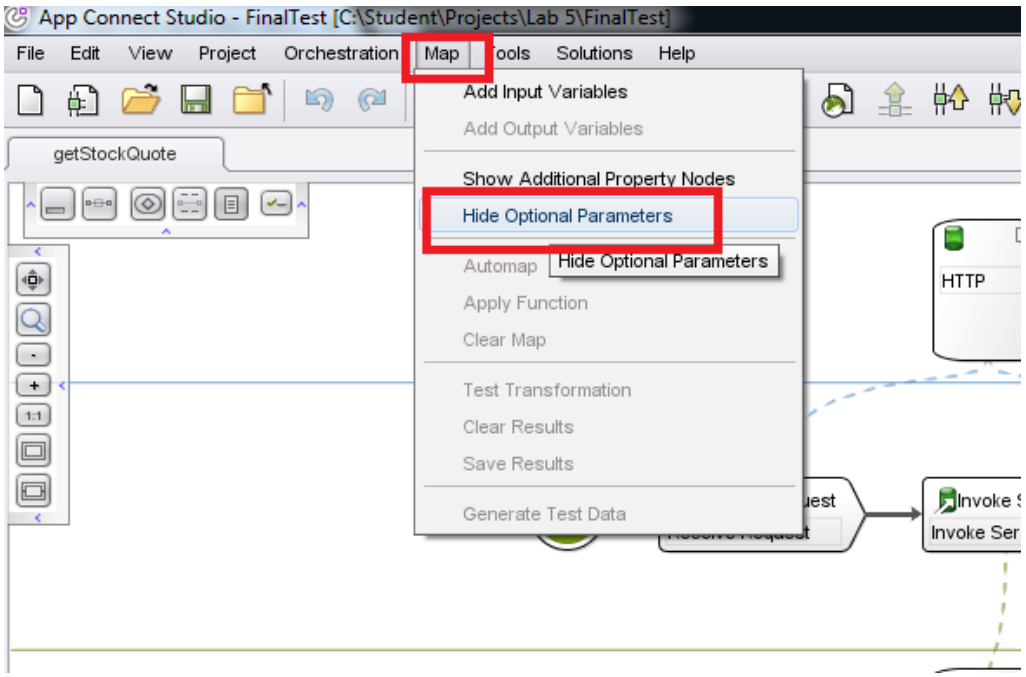
__ 15. Map Outputs: Copy only the activity's body to the orchestration and rename the variable: `invokeOutput`

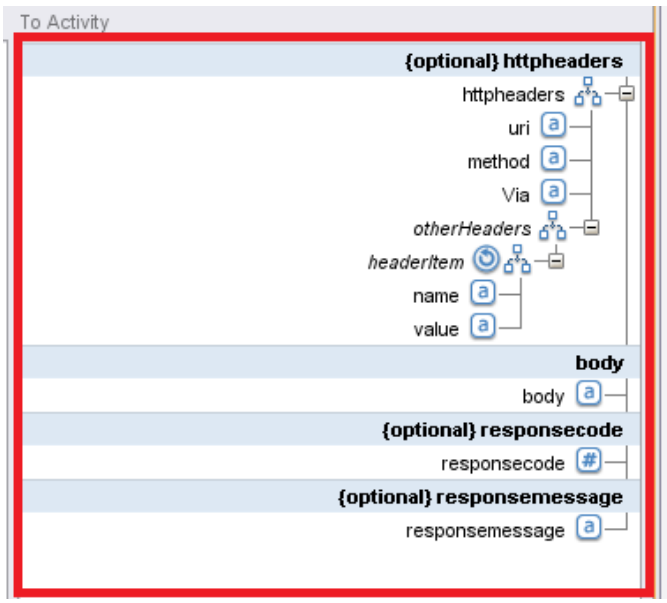


__ 16. Add an **HTTP Send Response** activity to the orchestration.

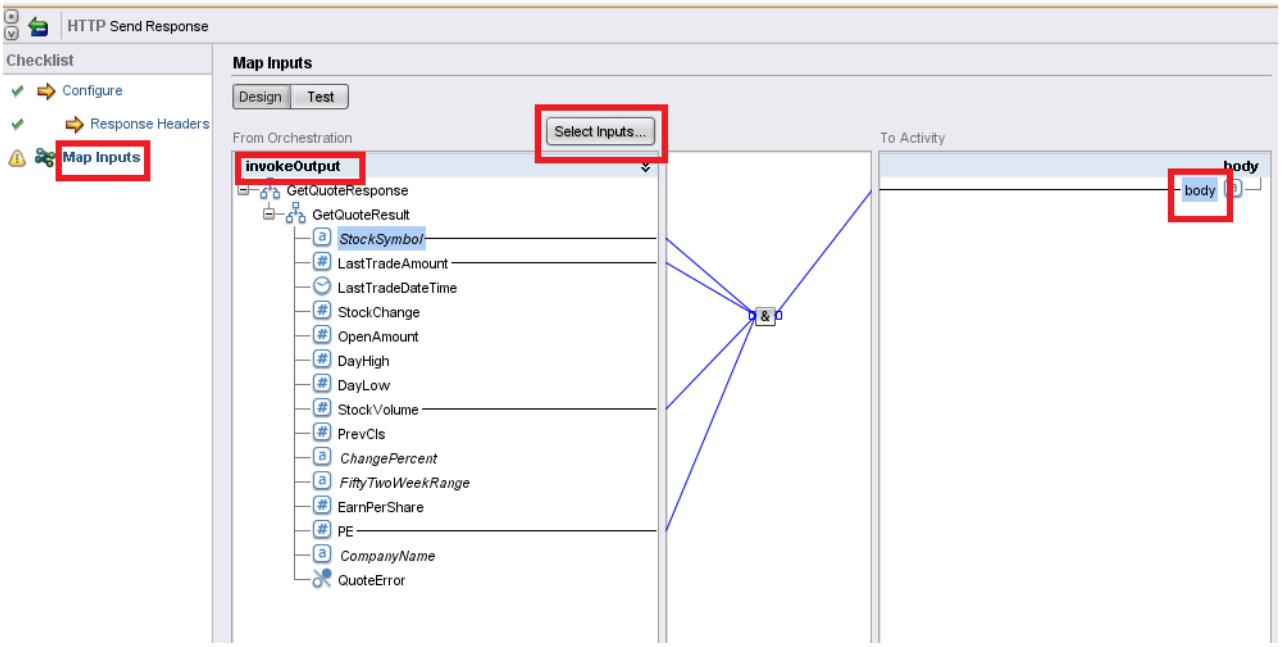


__ 17. Select **Map > Hide Additional Parameters** to display only the **body** parameter under **To Activity**.





__ 18. Map Inputs: Map the following input fields of the invoke Output structure to build an output message to the body. Click **Select Inputs** and then select **invokeOutput**. Use the Concatenate function.



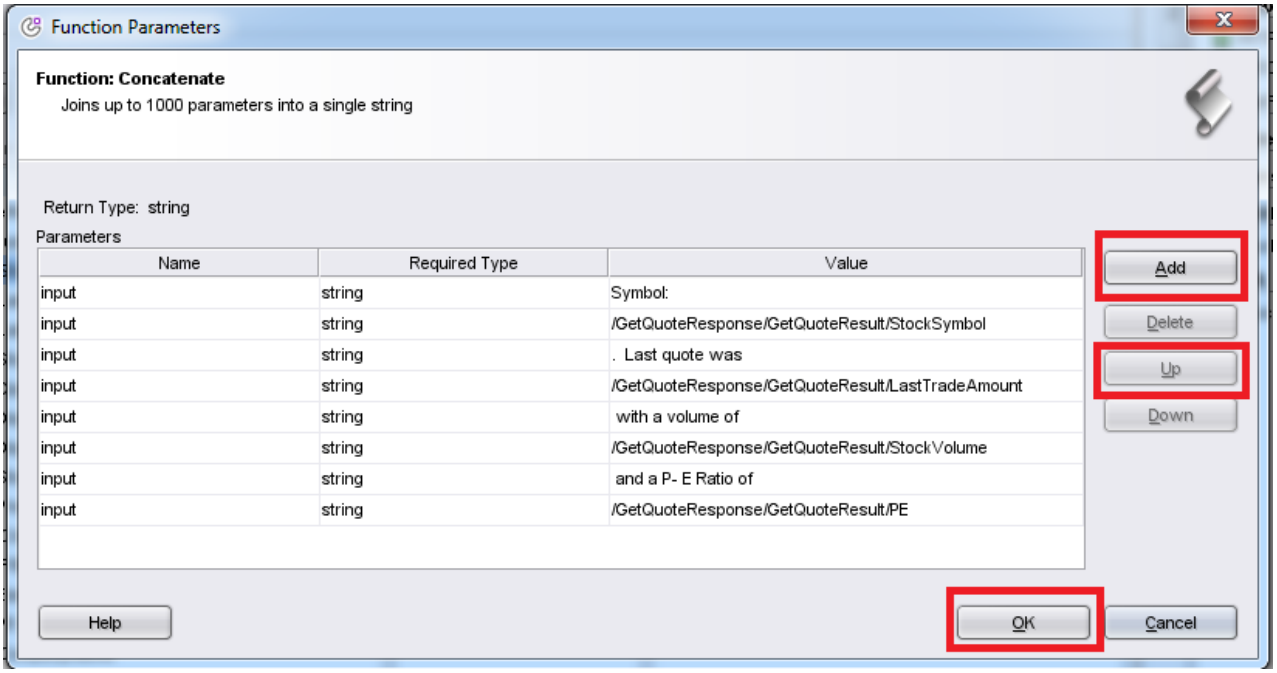
__ 19. Double-click the **Concatenate** function. Add and move UP the following string entries to build a message as follows:

```
Symbol: IBM. Last quote was 206.65 with a volume of 1673640 and a P-E Ratio
of 14.18
```

- "Symbol:"
- ".Last quote was"
- "with a volume of"
- "and a P-E Ratio of"

Note

Do not include the double quotation mark characters in the **Value** entries.



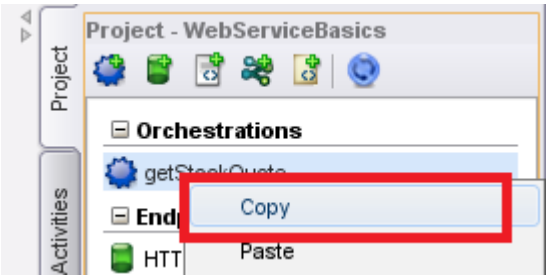
__ 20. Right-click the Concatenate function and Apply Function Graph.

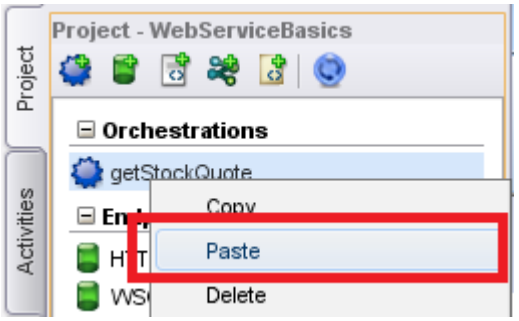
__ 21. Use the Studio Post Utility or Postman to publish, deploy, and test. Remember to POST by using the data port of your runtime environment and by using the URI as defined in the HTTP Post activity.

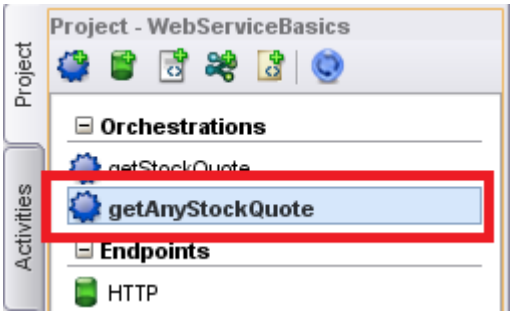
5.2. Invoke web service with Input

__ 1. Use the previous web service lab as an example for extracting the ticker symbol from user input.

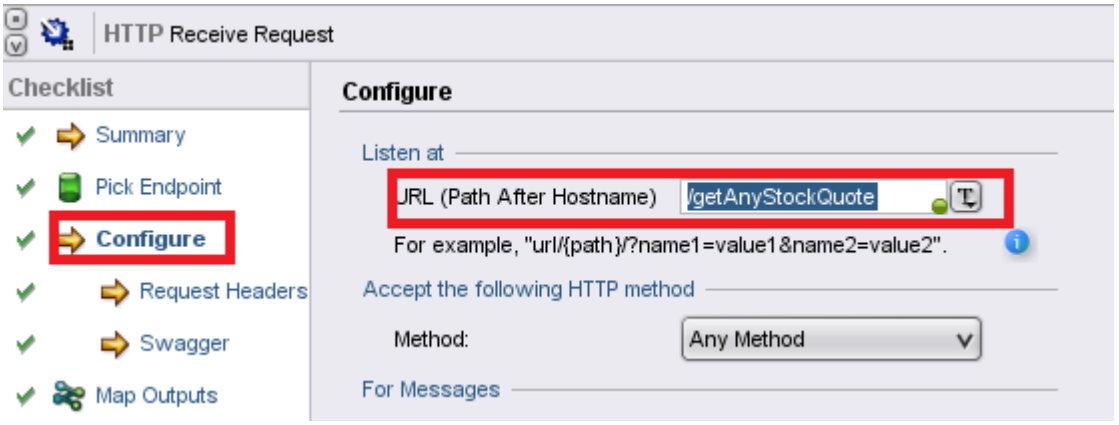
__ 2. Copy and Paste the existing getStockQuote orchestration and create a new one: `getAnyStockQuote`





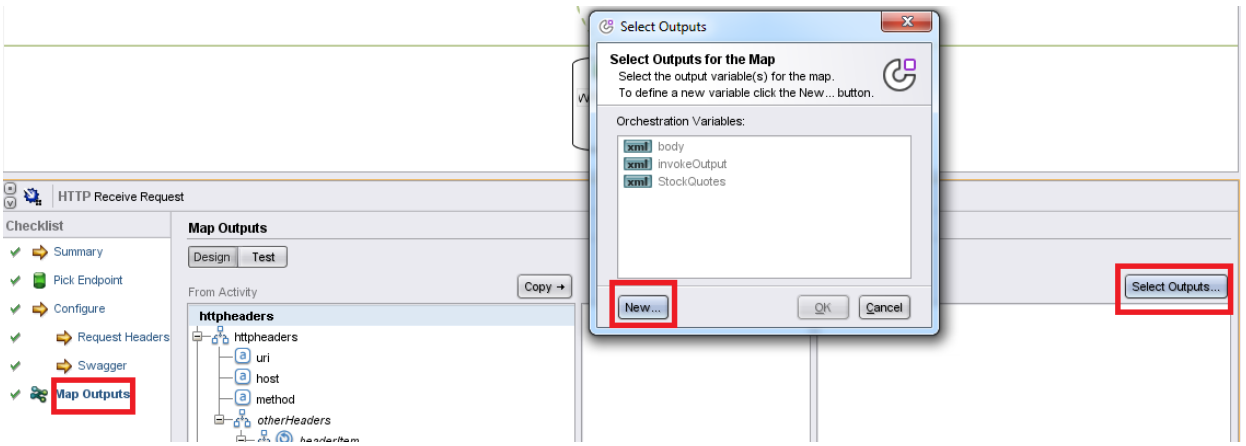


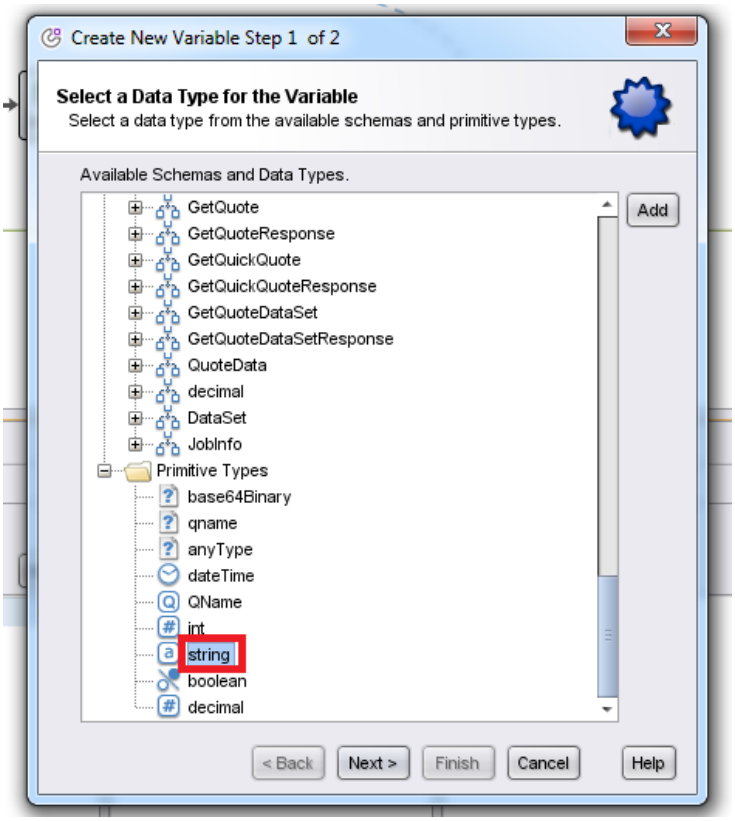
__ 3. Change the URL for the **HTTP Receive** activity to: `getAnyStockQuote`

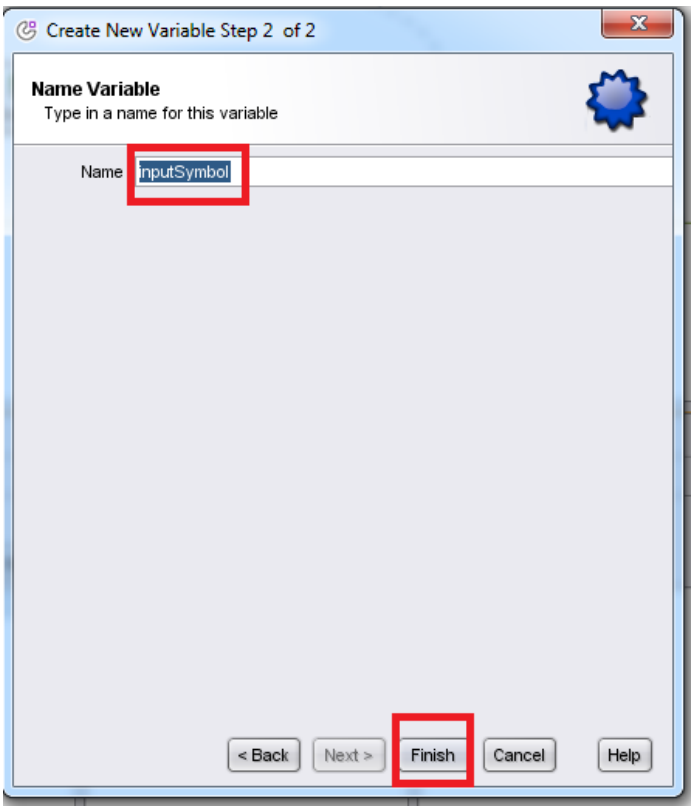


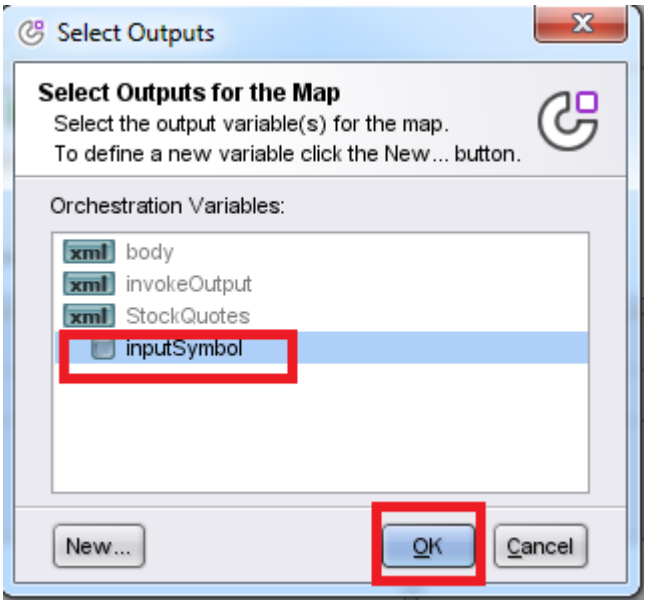
__ 4. In the HTTP Receive activity, extract the symbol value from user input and store it in an orchestration variable `inputSymbol` defined as string.

__ a. Map Outputs: **Select Outputs > New > string > Input Symbol > Finish**. Then, select **Input Symbol** and click **OK**.

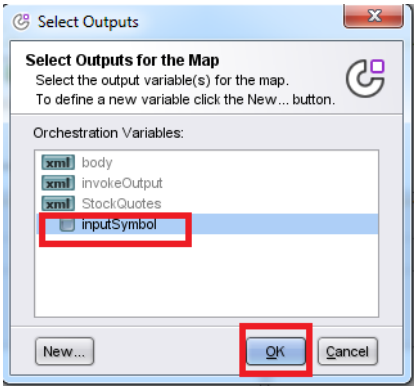




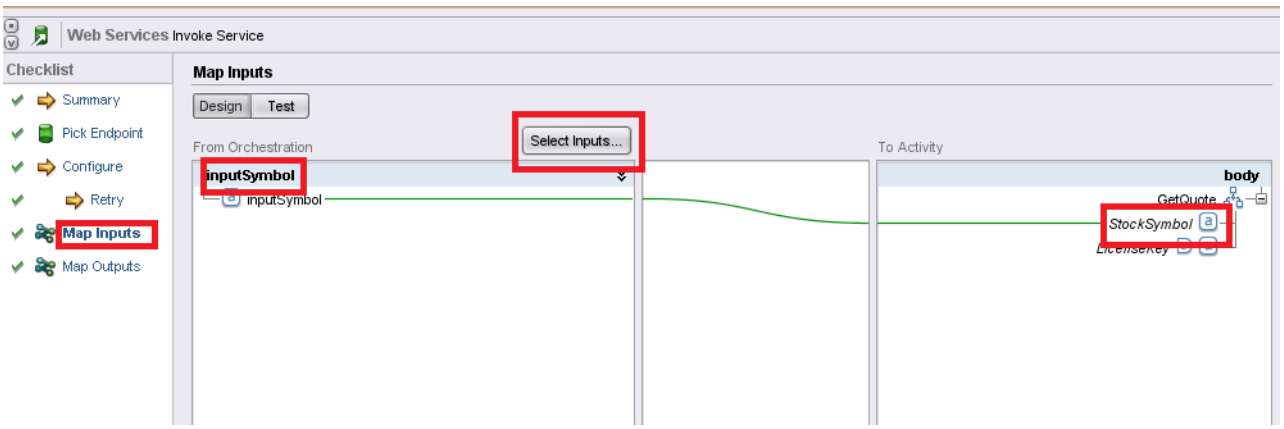




__ b. Map **body** to **inputSymbol**.

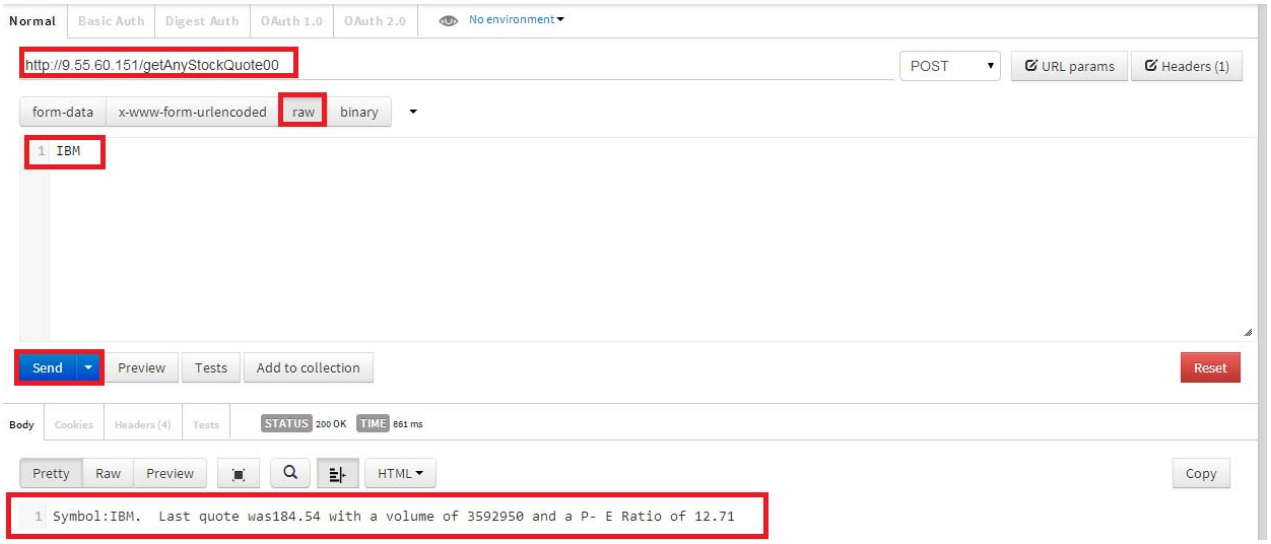


__ 5. Go to the **Invoke Service Activity > Map Inputs**. Right-click **Stock Symbol** and remove the default value of **IBM**. Click **Select Inputs** and choose **inputSymbol**. Map **inputSymbol** to **StockSymbol**.



__ 6. Save and deploy the project to your App Connect runtime environment.

__ 7. Test by opening the Postman utility in Chrome as in previous exercises. Use the Data port of your appliance as the host name. The source URI is the `/ getAnyStockQuote` that you previously set up in your HTTP Receive starter activity. You are sending XML (Text/XML). Use the raw data setting and type in the Ticker ID of the stock that you want to look up. Click **Send** when ready.



End of exercise

Exercise review and wrap-up

In this exercise, you created a project and configured the orchestration to have web services.

Exercise 6. Interacting with RESTful Services and Salesforce

Estimated time

00:60

Overview

With the massive proliferation of mobile applications, JSON over REST is finding heavy use in the industry. This lab is built around illustrating IBM App Connect Professional's ability to expose and work with REST Based services as part of an infrastructure that integrates seamlessly with software as a service, in this case it is with the Force.com API exposed by Salesforce.com. A new project is built, and it is exposed as a RESTful service, able to be consumed as an incoming request with a parameter as part of the URI, call a backend REST Service and then take the output of that request, format it such that it can be then upserted into an object residing in Salesforce.com, and provide a response to the original requester.

Objectives

After completing this exercise, you will be able to:

- Configure an HTTP Receive request
- Configure an HTTP Get request
- Configure an HTTP Send response
- Configure an If/Then command
- Read JSON
- Upsert objects with Salesforce

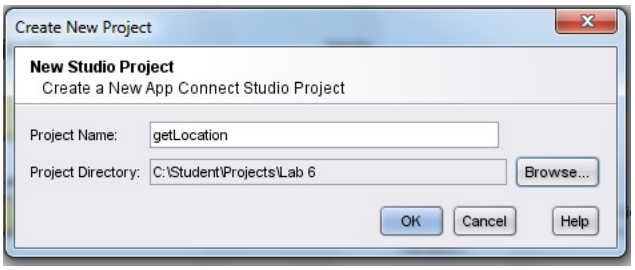
Introduction

RESTful services are different from SOAP services in the regard that there is no underlying WSDL metadata that describes the actual data payload. REST is a preferred protocol to work with by developers because of its lightweight footprint and its ability to easily work with parameterized data as part of the URL. App Connect Professional makes it easy to work with RESTful services that are now becoming part of many companies' cloud and mobile strategies.

Exercise instructions

6.1. Expose an orchestration as a RESTful service that invokes a second REST service and synchronizes data into Salesforce.com

__ 1. Create a project. Name this project: **`getLocation`**
Save this project to the following location: **`c:\Student\Projects\Lab6`**



__ 2. Rename the orchestration to `getAndSyncBranch`

__ 3. Create three configuration properties.

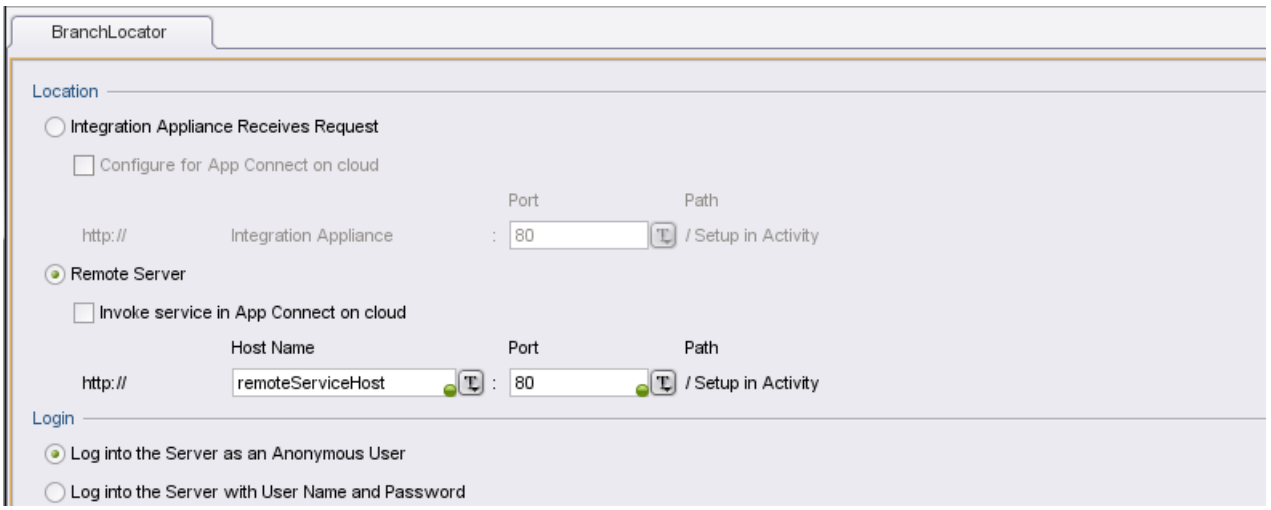
__ a. remoteServiceHost: Hardcode this value to **`banka.mybluemix.net`**

__ b. salesforce.user: Set to your salesforce.com user ID used in previous labs.

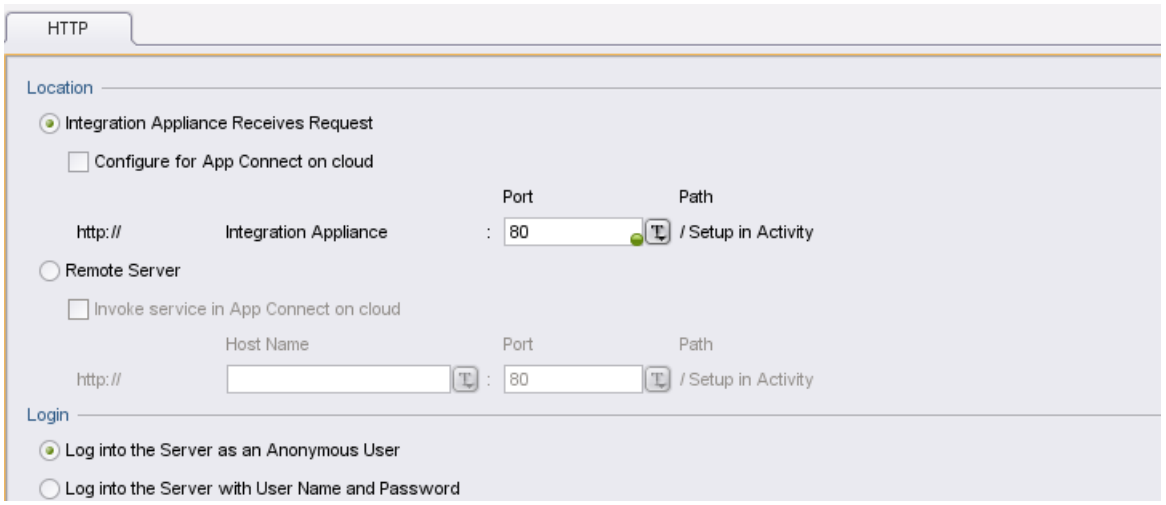
__ c. salesforce.pw: Set to your salesforce.com password+token ID used in previous labs.

__ 4. This Project contains three endpoints.

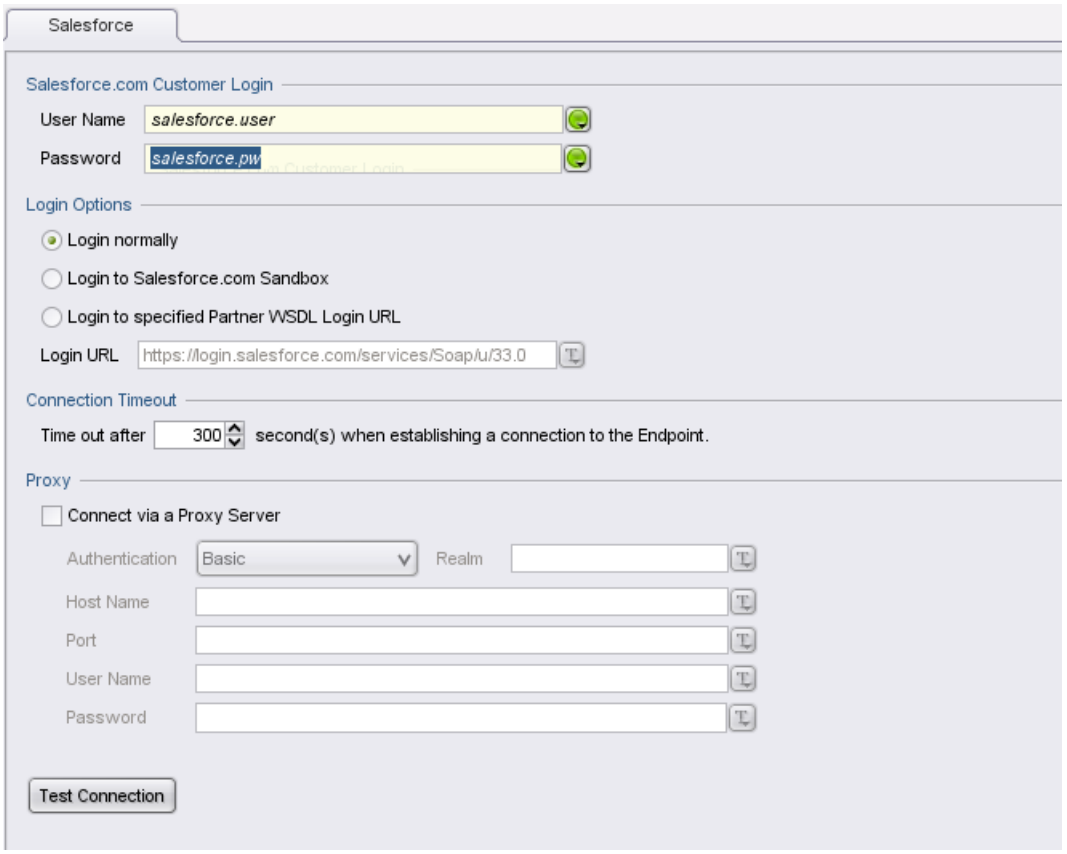
__ a. The first endpoint is an HTTP endpoint. Name it `BranchLocator`. Set this up as a *remote* server call. Set the host name to the remoteServiceHost configuration property. Use the default port 80. Ensure the Login section is set to the default “Log into Server as an Anonymous User”.



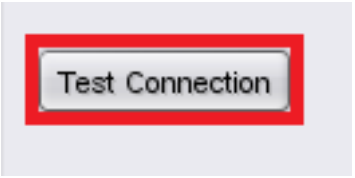
__ b. The second endpoint is the default HTTP Receive endpoint. Leave the default name of “HTTP”.



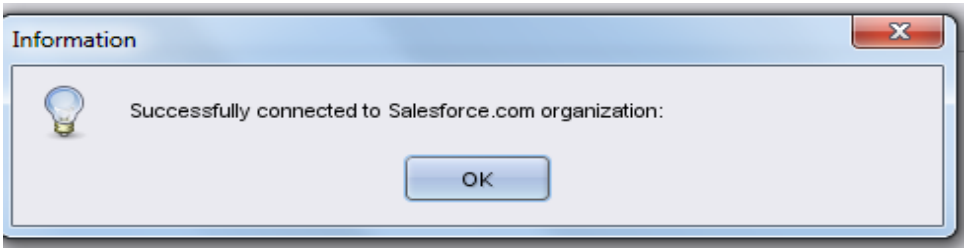
__ c. The third endpoint is Salesforce.com. Leave the default name of “Salesforce” for this endpoint. Use the salesforce.user and salesforce.pw configuration properties in the endpoint configuration as depicted in the following figure as done in previous exercises.



__ d. Test the Salesforce endpoint. Click the **Test Connection** button.



__ e. If successful, you see the following response in IBM App Connect Studio.



__ 5. Drag the following activities on to the getAndSyncBranch orchestration canvas in the following order:

__ a. **HTTP Receive Request**

__ b. **HTTP Get Request**

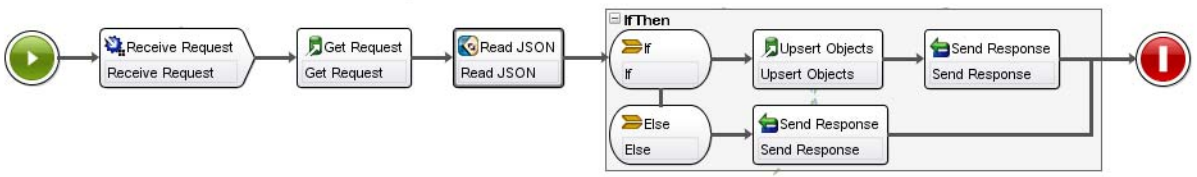
__ c. **Read JSON**

__ d. **IF/Then** block

__ e. With in the **If/Then** Block: For the top path drag in a Salesforce **Upsert Objects** and then HTTPSendResponse

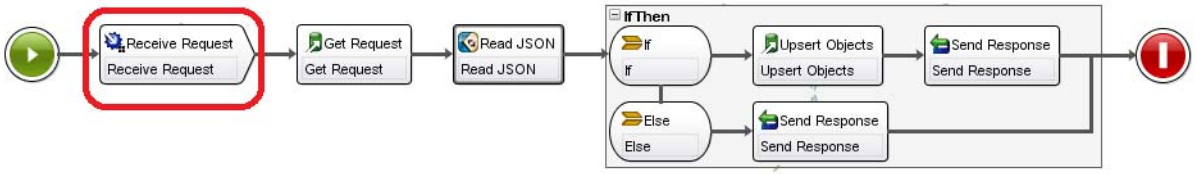
__ f. On the bottom (else) path of the **If/Then:** Drag in a second **HTTP Send Response**.

__ g. Confirm your getAndSyncBranch orchestration:



__ 6. Configure the Activities within the orchestration

__ a. Start with the **HTTP Receive Request** activity. Click the activity to start the configuration.



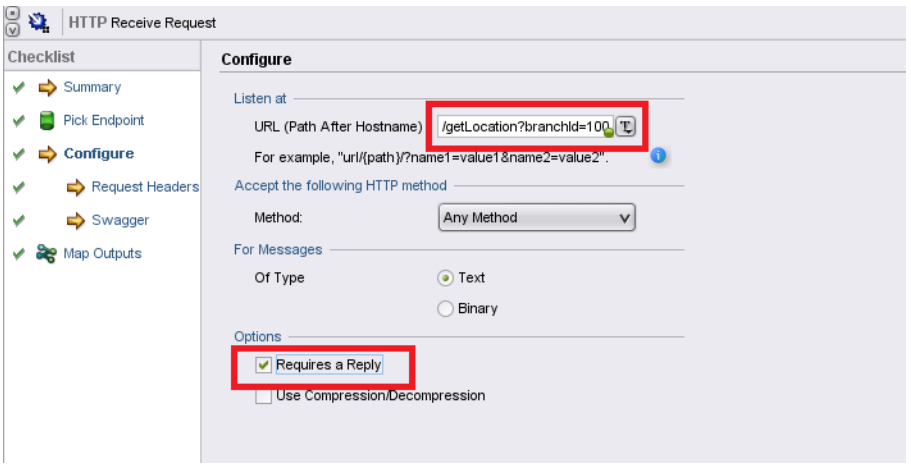
__ b. Complete the **HTTP Receive Request** activity checklist.

i. Under the “Pick Endpoint” step of the checklist, select the HTTP endpoint. This represents the CI Runtime environment as the hosting endpoint for the orchestration.

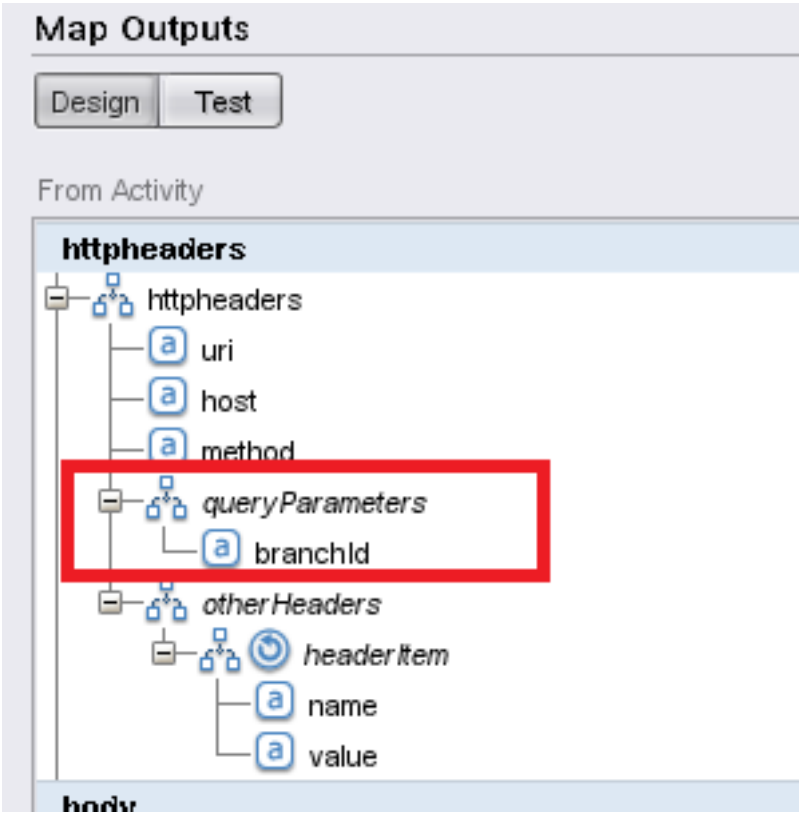
ii. On the “Configure” step of the checklist, Configure the step as follows.

1) Set the URL to this value: `/getLocation?branchId=100`. This does not mean you are hardcoding the branch ID value in the URL. Rather, you are creating a variable called **`branchId`** that the invoker of this orchestration can use to pass in variable data as part of the URL that you can use in the orchestration further down stream.

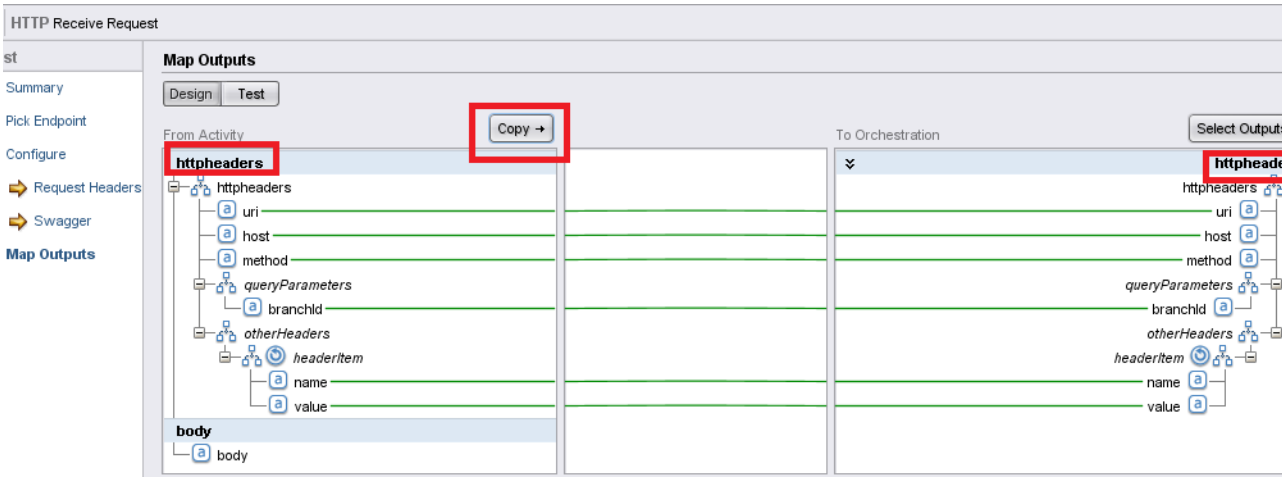
2) Ensure the **Requires a Reply** check box is selected.



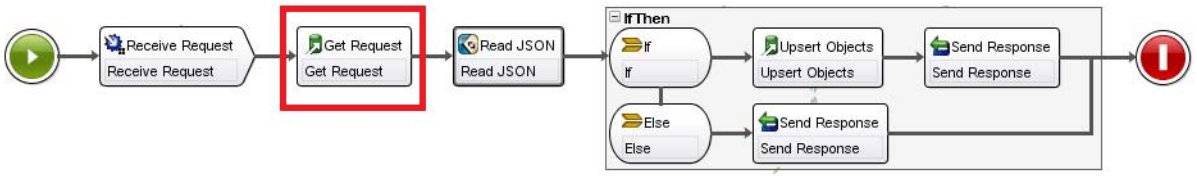
3) Go to the **Map Outputs** step. Notice that the `branchId` variable is set up under the `queryParameters` object. You can easily capture and map the query parameters that are passed to the incoming request.



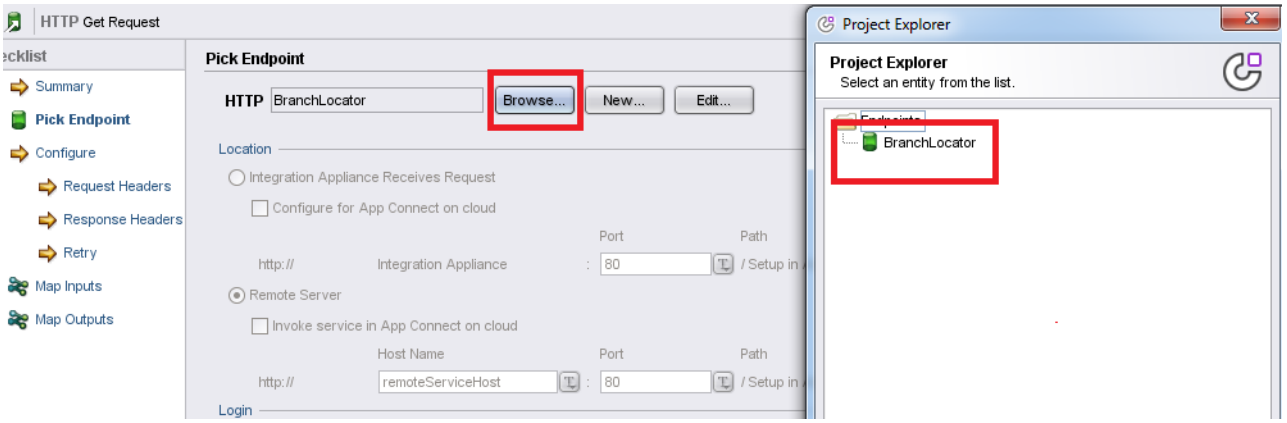
iii. You can now map the entire contents of the http headers variable over by using the **Copy** button.



__ c. Complete **HTTP Get** activity checklist. Click the **HTTP Get** activity and proceed to the checklist.



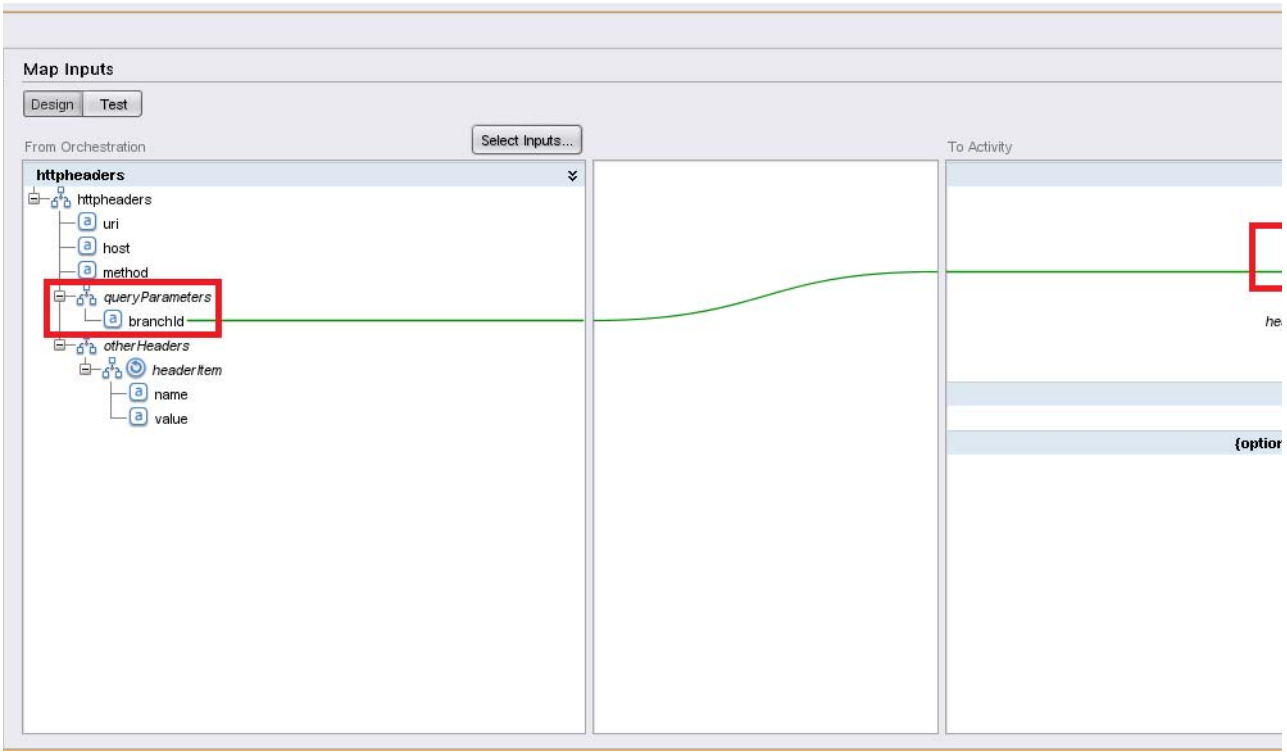
i. On the Pick Endpoint step, select the BranchLocator HTTP endpoint from the list by clicking **Browse**.



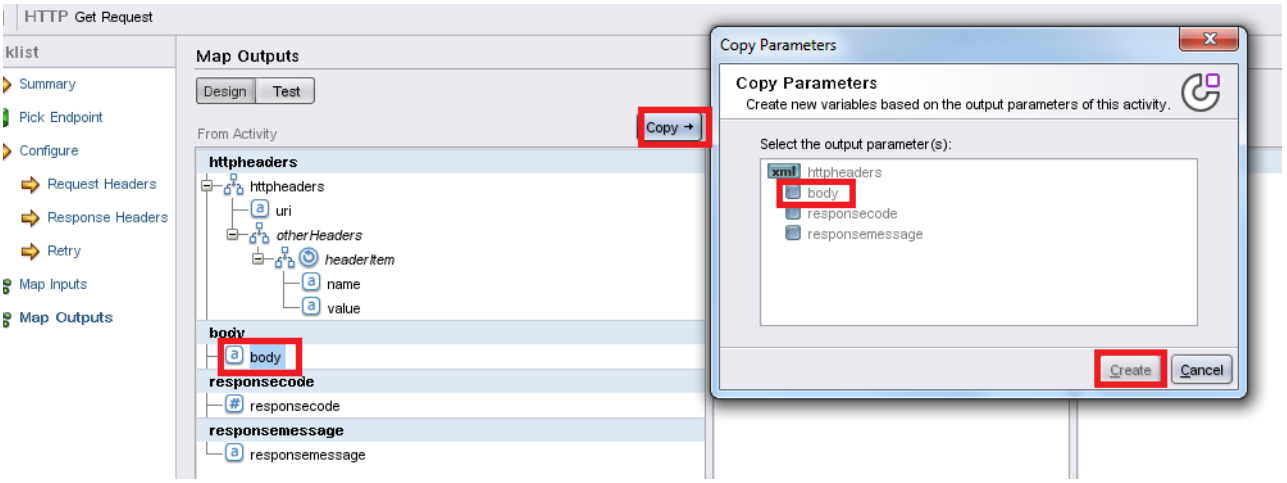
ii. On the Configure step, use the following text as the URL for the HTTP Get:
`/branches/v1/{branch}/location`
The `{}` notation in the URL represents a variable component of the URL of the HTTP Get. In this way, you can map in a parameter. For more information, see the IBM Knowledge Center:
https://www.ibm.com/support/knowledgecenter/SSGR73_7.0.0/com.ibm.wci.doc/http_get_request_activity.html

iii. Accept the defaults for the Request/Response Headers and the Retry steps of the checklist.

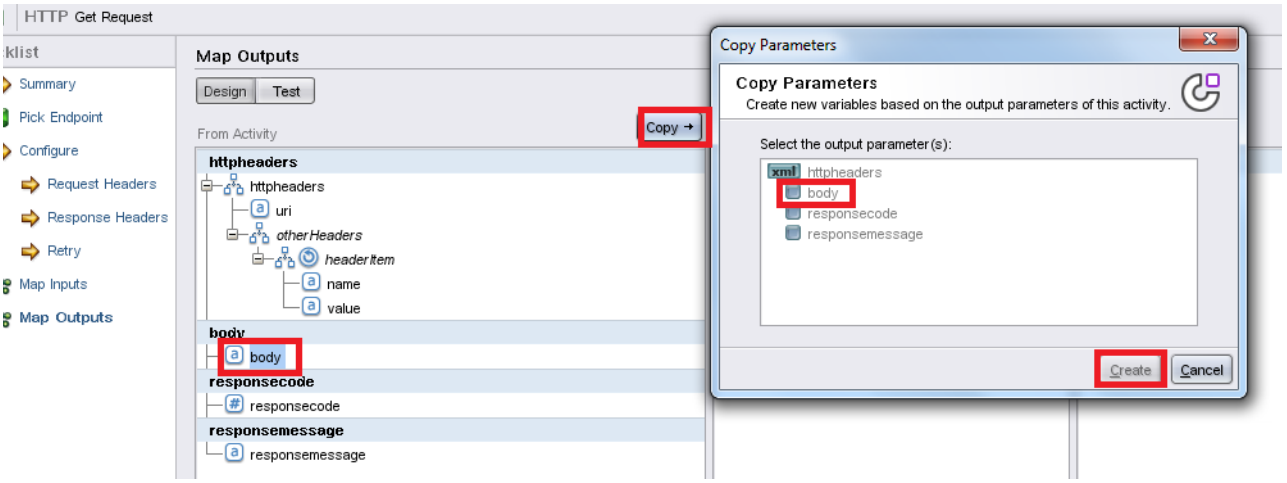
iv. On the Map Inputs step, on the “From Orchestration” on the left side, click **Select Inputs** and select **httpheaders**. Then, drag the branchId variable from the queryParameters to the branch variable under pathParameters on the “To Activity”.



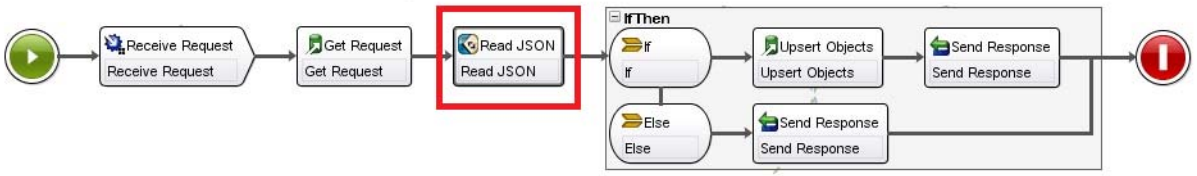
v. On the Map Outputs step, copy the “body” object to a variable, by clicking the **Copy** button, then in the “Copy Parameters” window, select **body** and click **Create**.



vi. When complete, your map outputs step looks like the following figure:



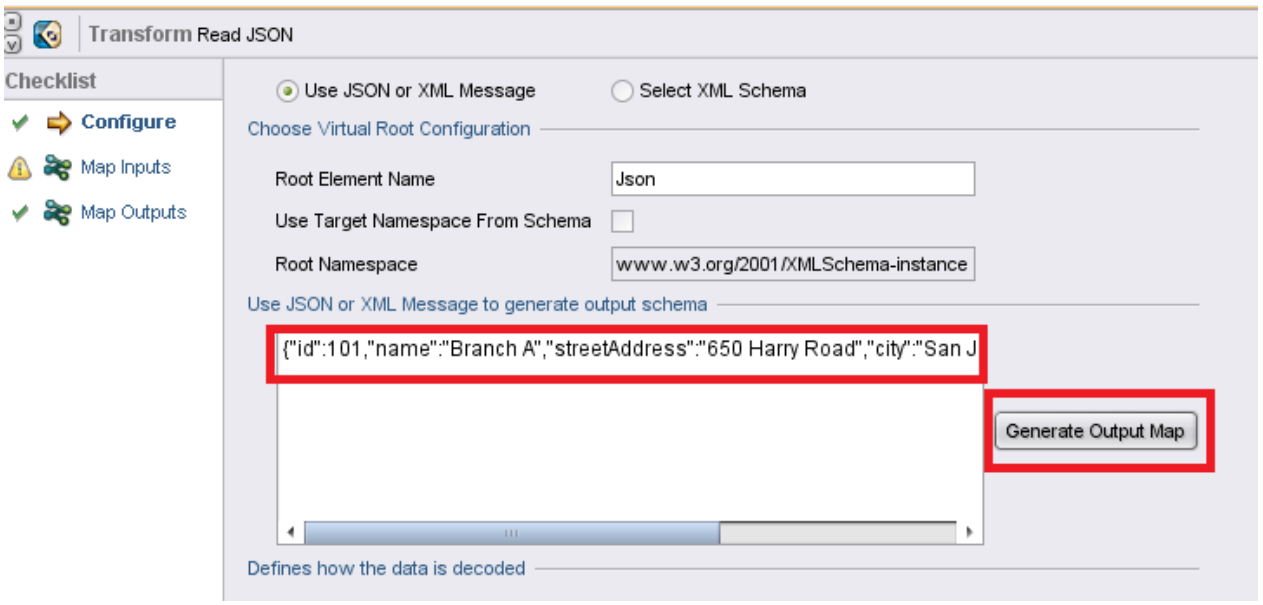
__ d. Complete the **Read JSON** activity. Click the activity and then start with the checklist.



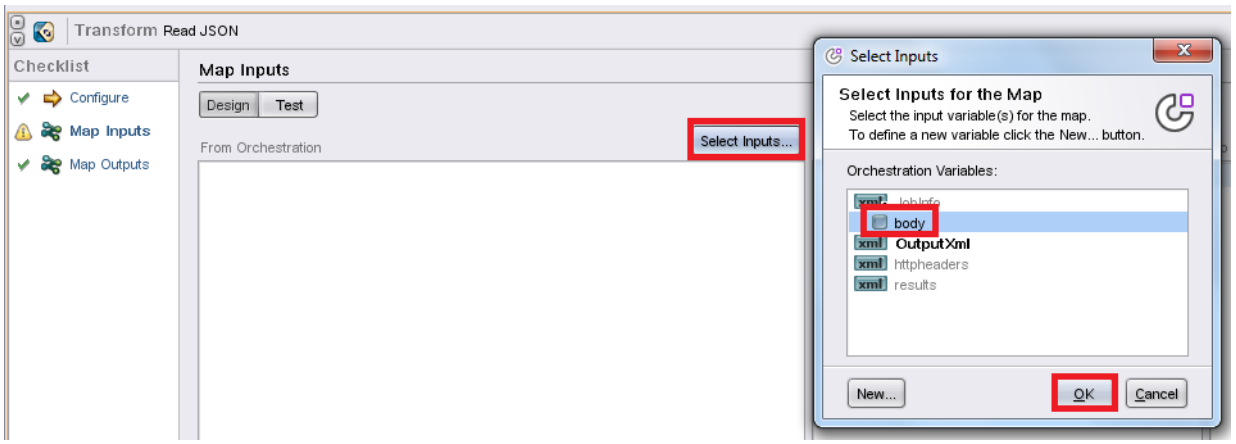
i. Start with the Configure step in the checklist. Copy and paste the following Sample JSON into the lower white space of the activity. This data is to be used to reflect the output from the BranchLocator service. Then, click the **Generate Output Map** button.

```
{"id":101,"name":"Branch A","streetAddress":"650 Harry Road","city":"San Jose","state":"CA","zipCode":"95120-6099"}
```

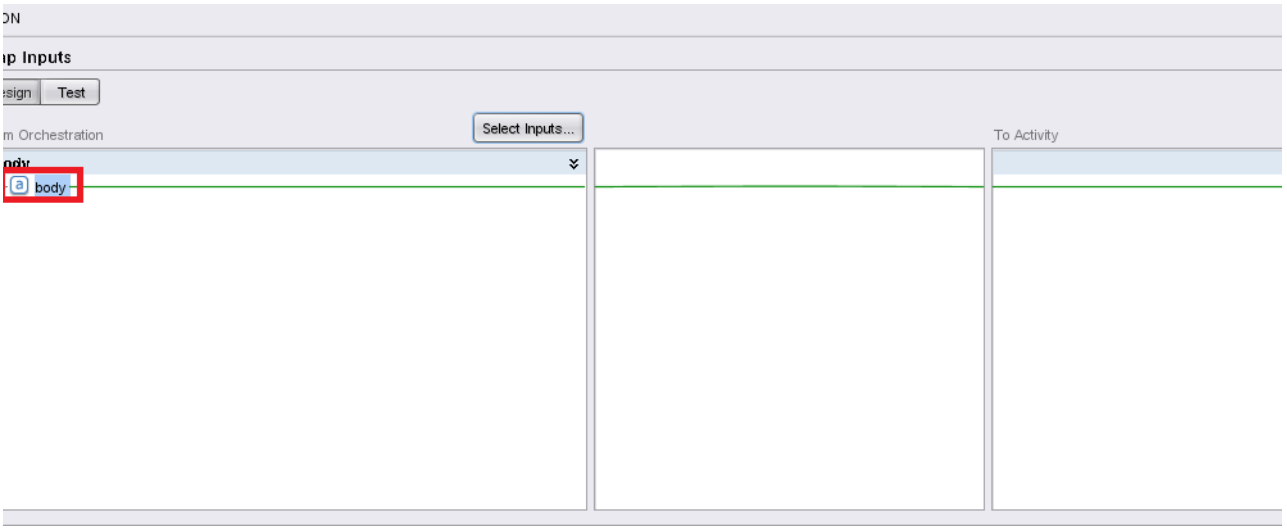
Alternatively, you might get the JSON data sample by running the service manually in a browser by going to the following URL:
http://banka.mybluemix.net/branches/v1/101/location



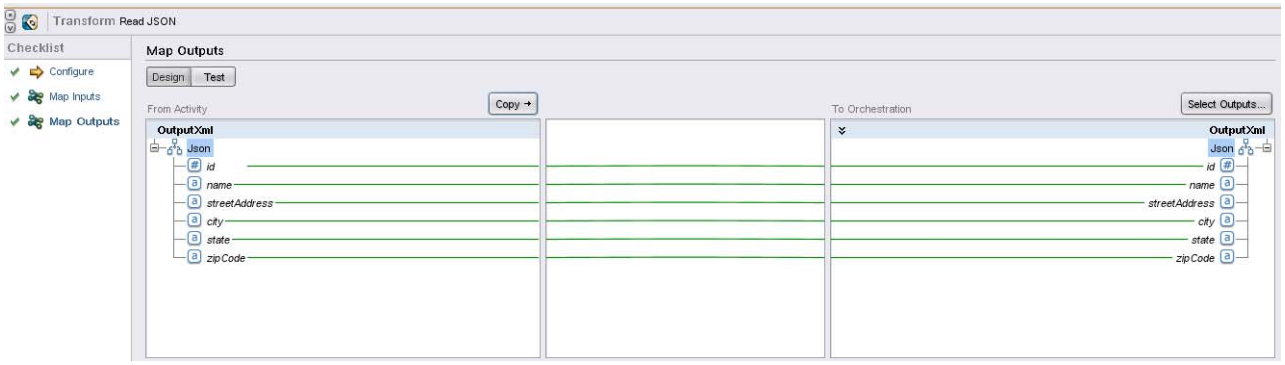
ii. Moving to the Map Inputs step. Add the body variable by clicking **Select Inputs**, highlighting the body variable, and then clicking **OK**.



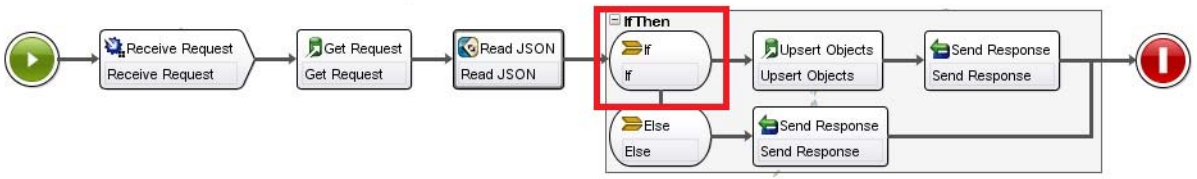
iii. Drag the body variable on the “From Orchestration” (left) side over to the JsonText variable on the “To Activity” (right) side.



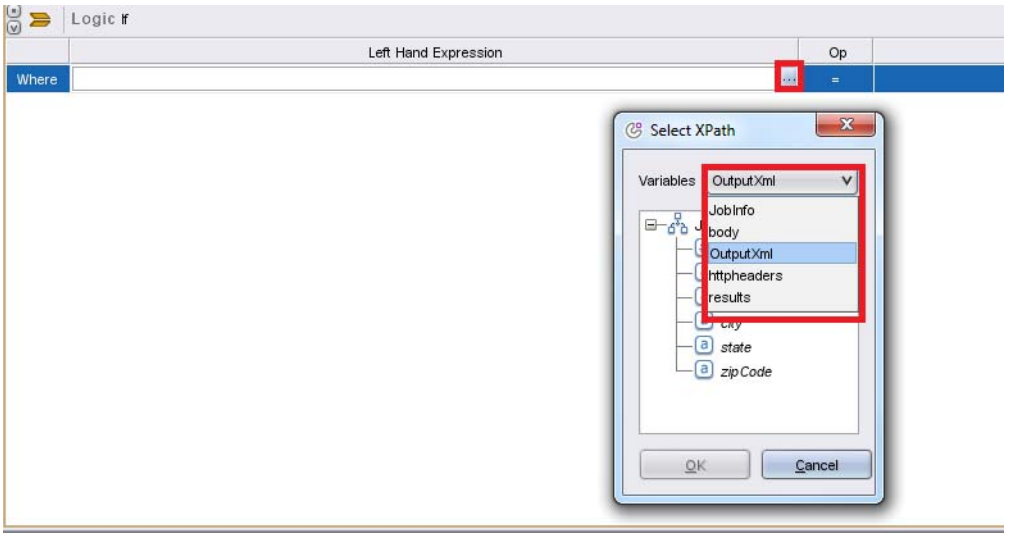
iv. Click the Map Outputs step. Click the **Copy** button, then select the OutputXML variable, and then click **Create** to map the variable that represents the parsed JSON data.



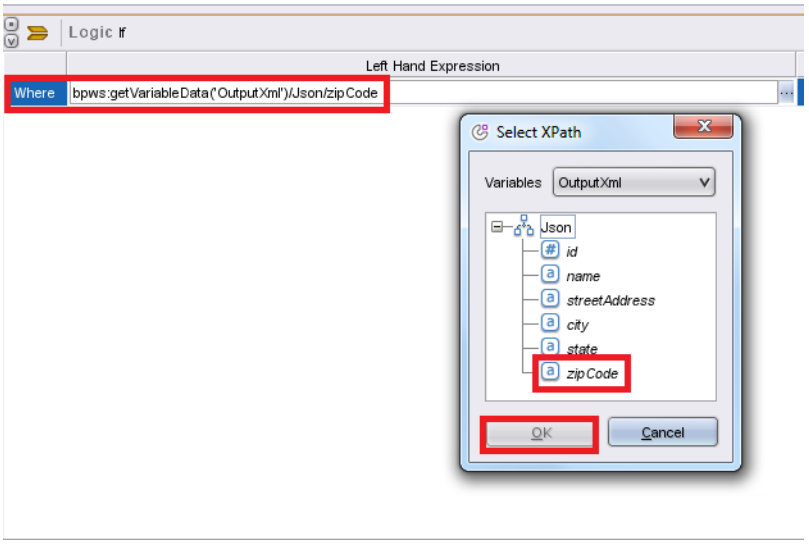
__ e. Configure the **IfThen** activity. Click the activity and then proceed to configure.



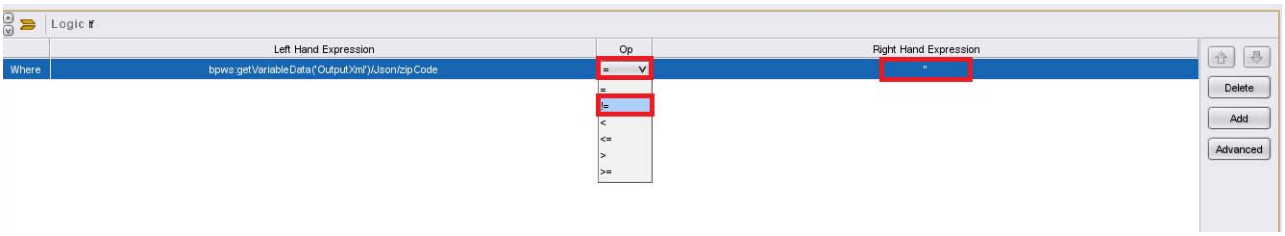
i. At the right of the left hand expression white space, click the box with the three dots on it to open the Select XPath window. Here, you can select the variables to evaluate. Select the OutputXML variable.



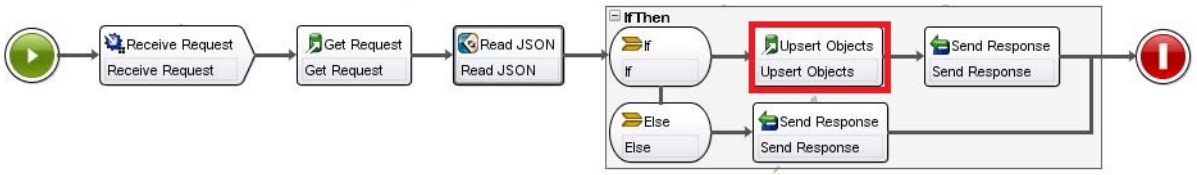
ii. Select the **ZipCode** field and then click **OK** to automatically populate the left hand expression with the requisite xpath.



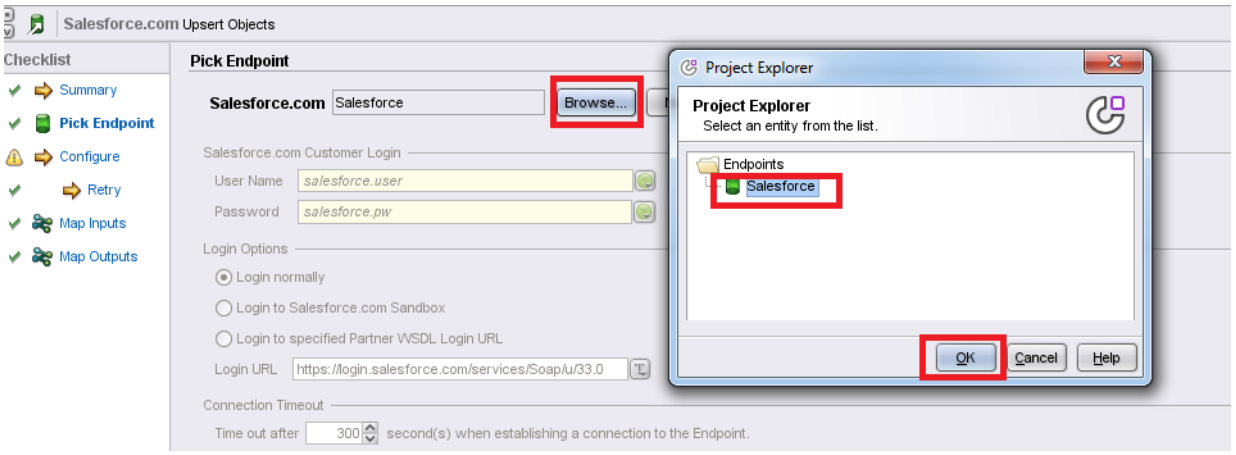
iii. Set up your expression next where you are going to evaluate the ZipCode field to be Present or Not Equal to None. Select the Not Equals operation (**!=**) from the **Op** list. Next, type in two single quotes (**''**) into the right hand expression.



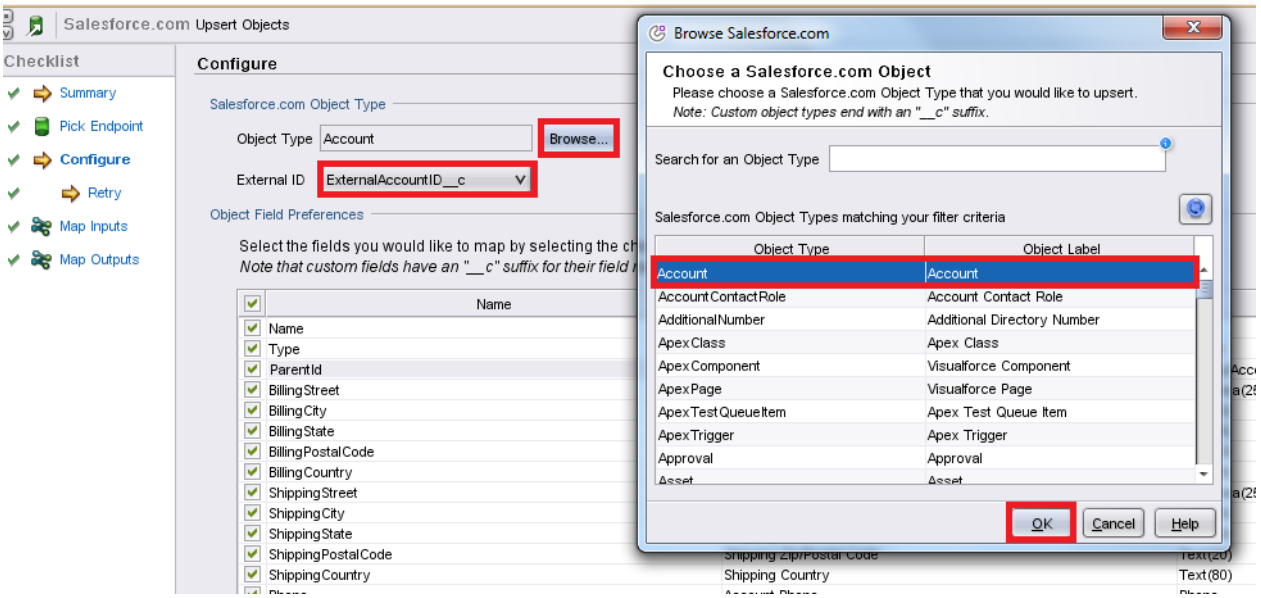
__ f. Configure the **Upsert Objects** activity. Click the activity and then proceed to configure.



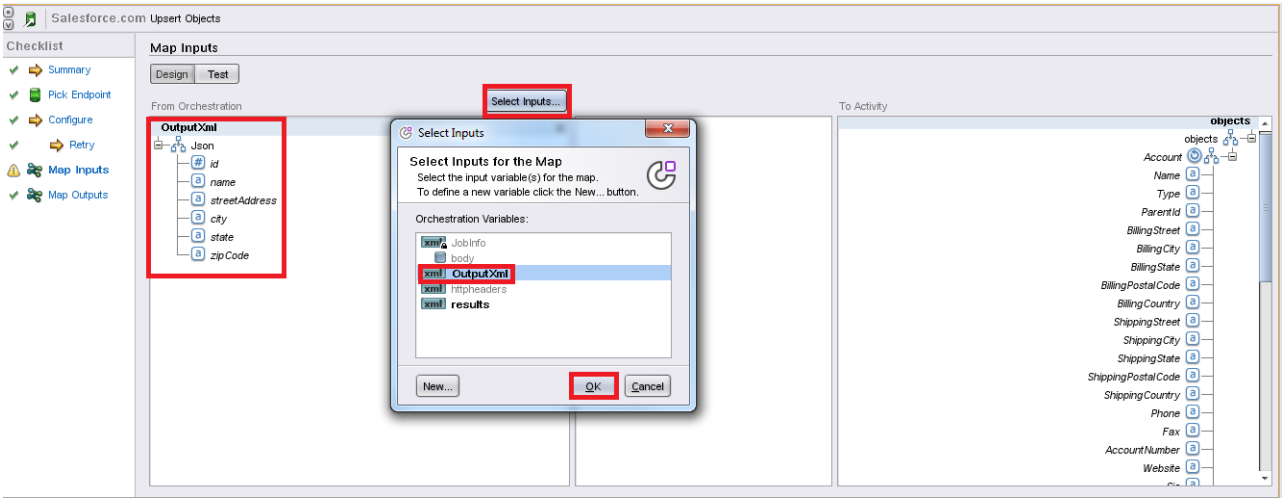
__ g. Configure the Pick Endpoint step by clicking the **Browse** button and selecting the Salesforce endpoint from the list. Click **OK** to complete the step.



__ h. Go to the Configure step in the checklist and then click the **Browse** button so that Studio connects to Salesforce to get a list of your objects. Select the **Account** object as indicated in the following figure and then click **OK** to bring back the metadata for your account, including the fields and their respective properties. Next, ensure that the External ID is selected to the ExternalAccountID__c custom field.



__ i. Go to the Map Inputs step. Click the **Select Inputs** button and then select the **OutputXML** variable from the list. Click **OK**. The OutputXML is now the input data to be mapped into Salesforce.

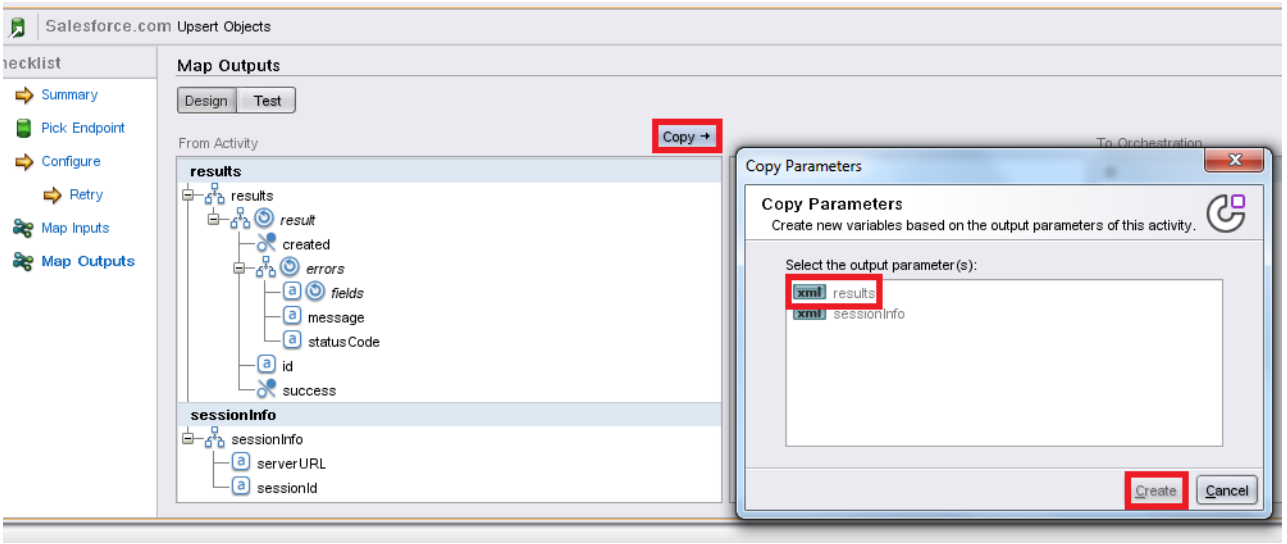


__ j. You can now start to map fields from the input. Use the following Mapping Guide.

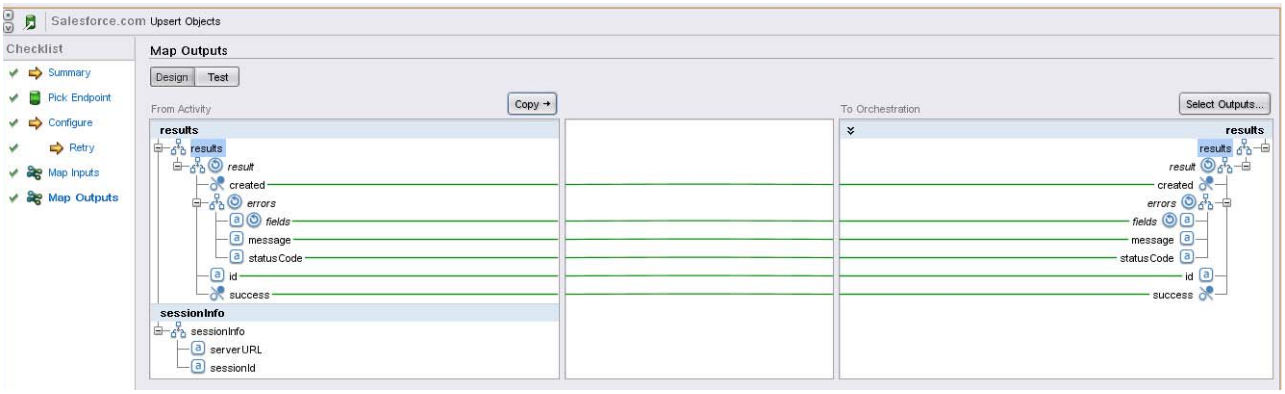
Table 4.

Input Field	Output Field	Transformation Rules
Id	ExternalAccountID__c	Direct mapping, no transformation
Name	Name	Direct mapping, no transformation
streetAddress	BillingStreet	Direct mapping, no transformation
City	BillingCity	Direct mapping, no transformation
Street	BillingCity	Direct mapping, no transformation
ZipCode	BillingPostalCode	Direct mapping, no transformation

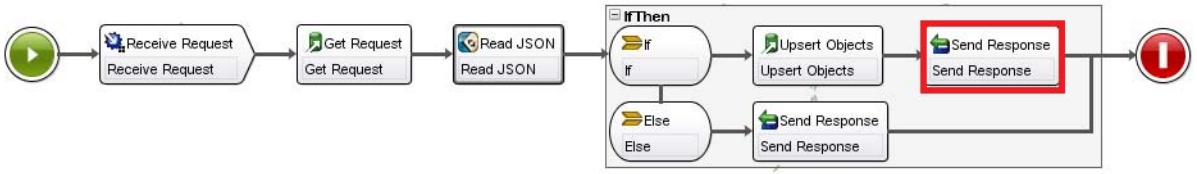
__ k. Go to the Map Outputs step. Here you map the result set returned from Salesforce.com after the upsert, and then proceed to map the output to a variable. Click **Copy**, then in the Copy Parameters pop-up, select the results variable, and click **Create**.



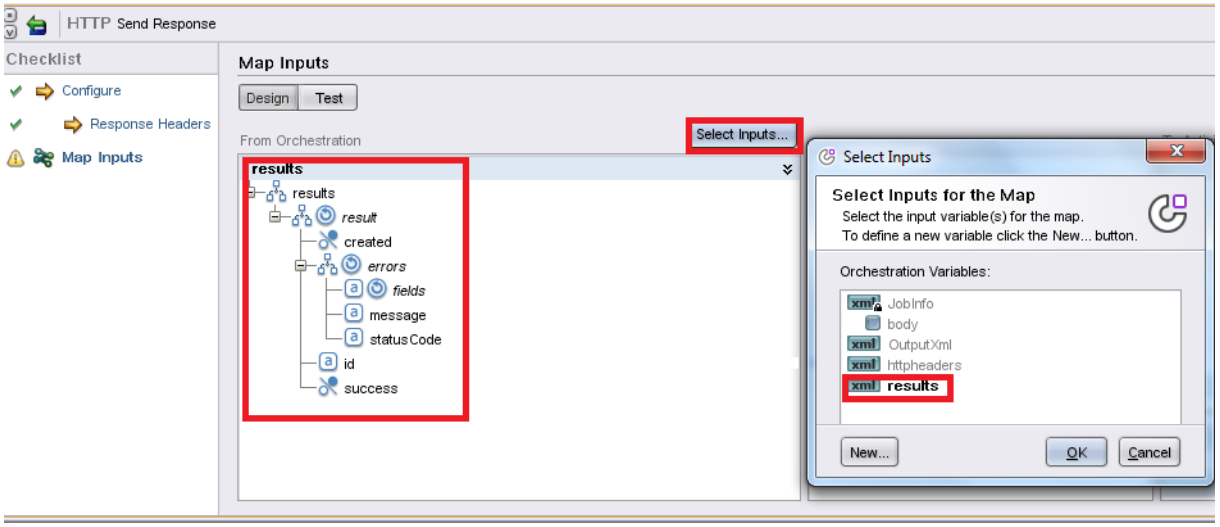
__ l. As soon as it is complete, your Map Outputs step should look as follows:



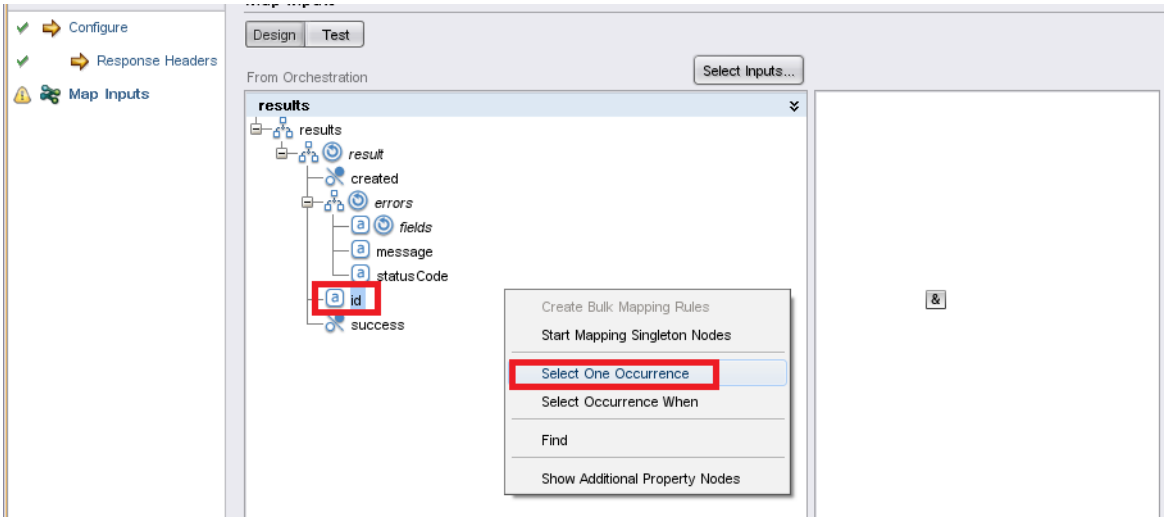
__ m. Configure the **Send Response** activity for the top branch of the **IfThen**. Click the activity and then proceed to configure.



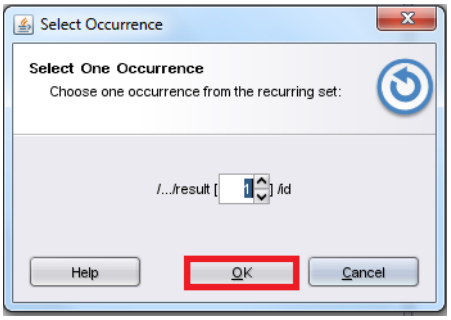
__ n. In the Map Inputs Step, click the **Select Inputs** button and in the window, select the results variable. Click **OK**, and the results variable with its elements appears on the left side of the window.



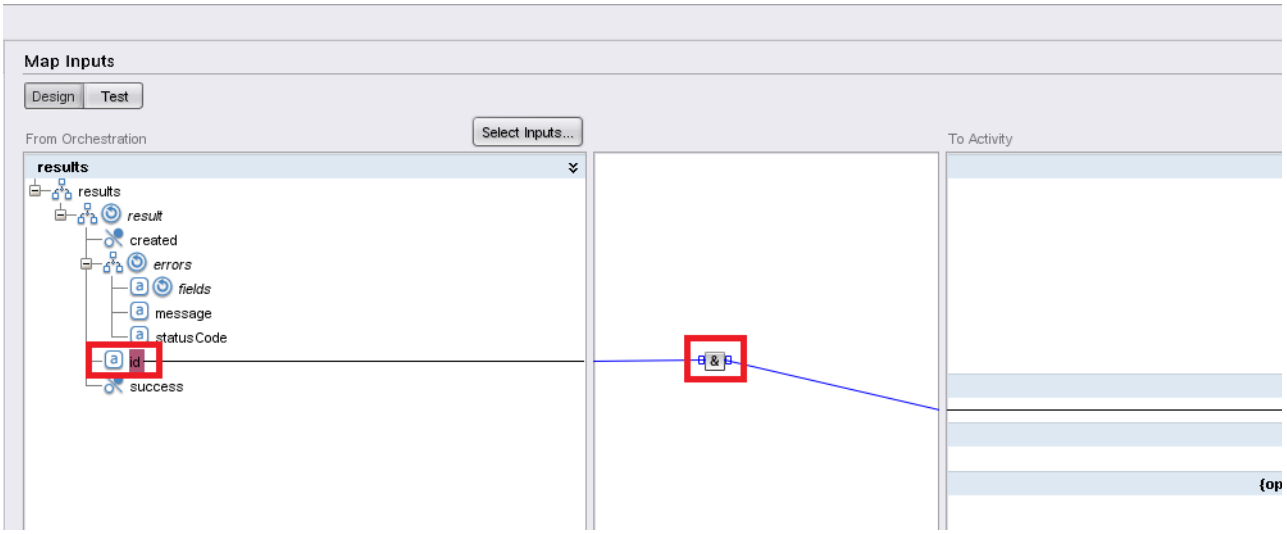
__ o. Next, click the **Functions** tab on the right on side of the workspace. Drag the **Concatenate** function from the String Functions folder onto the middle mapping pane. To map over the `id` value from the results variable, you must select a specific occurrence of that variable since the result object is a repeating structure. To do so, right-click the `id` variable and click the **Select One Occurrence** option.



__ p. When the next window opens, click **OK** to finish the selection.



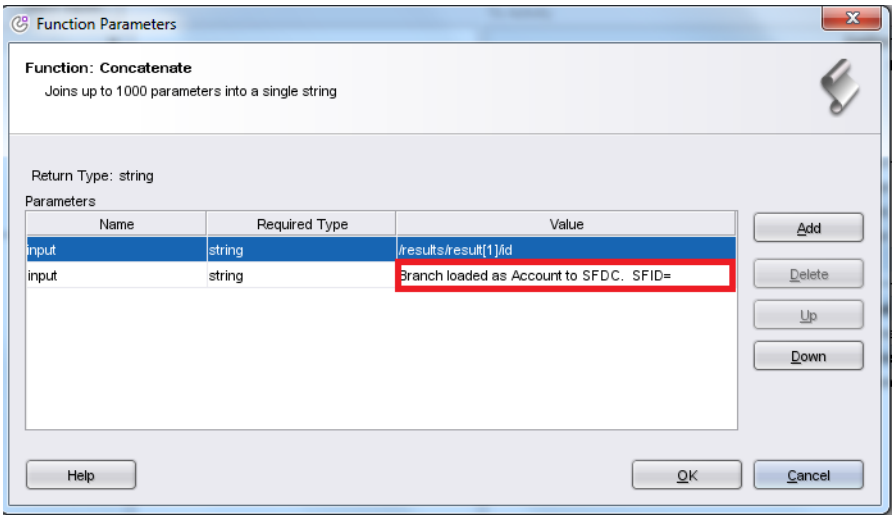
__ q. When the step is complete you can drag the **id** field into the *concatenate* function, and then click and drag the concatenate to the body field on the output side.



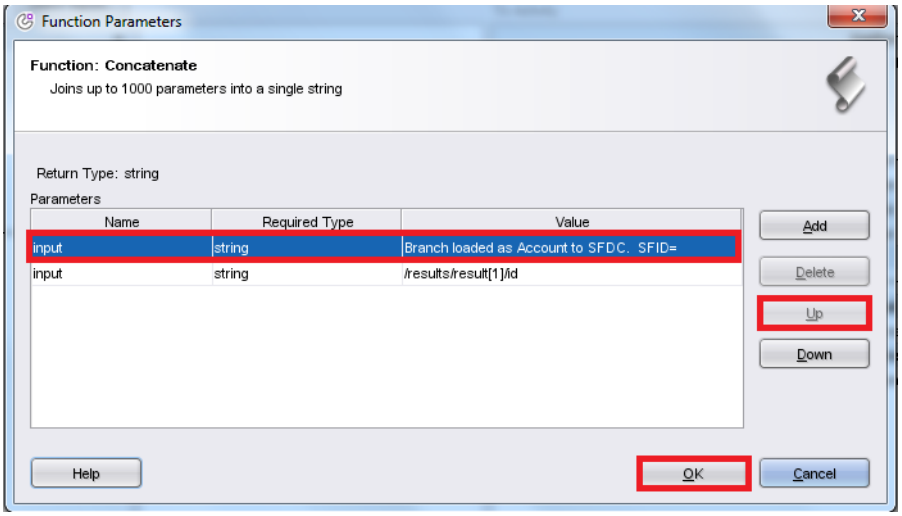
__ r. Next, double-click the concatenate function that was placed on the middle mapping plane in the step above to configure it. Notice that the id field is in the concatenate list, along with an empty field is to be concatenated together along with the id. Type in the following literal value into the field:

```
Branch loaded as Account to SFDC. SFID=
```

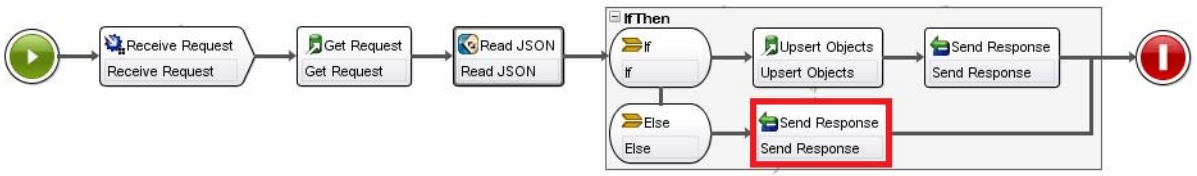
When complete, the function should look as follows.



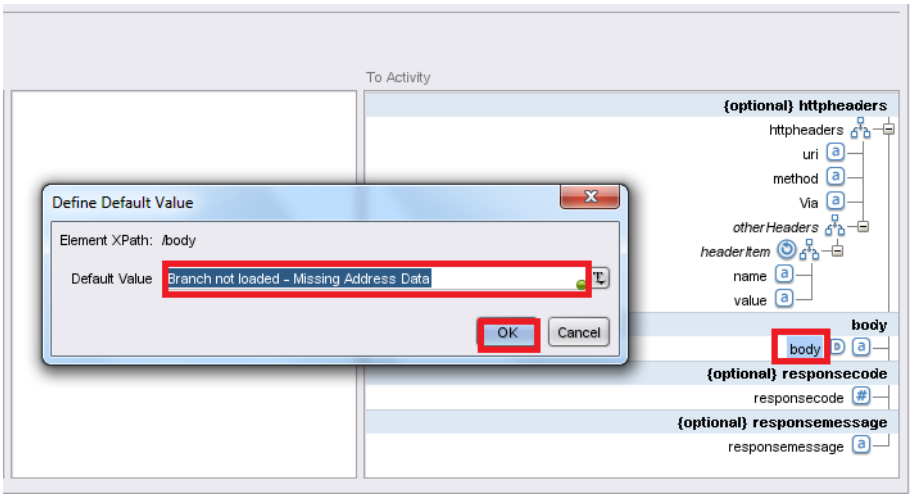
__ s. To reorder the field listing in the *concatenate* function, select the field that you just entered, and then move it up to the top by clicking the **Up** button so that it is situated ahead of the id field. When this step is done, click **OK**.



__ t. Configure the **Send Response** Activity for the bottom branch of the **IfThen**. This represents your response if the data retrieved from the BranchLocator service is missing the Zip Code and therefore is not valid. Click the activity and then proceed to configure.



__ u. Configure the map inputs step by providing a literal value for the body of this HTTP reply to reflect that the branch information was not loaded. Use the literal value indicated in the screen capture. Click **OK** when complete.

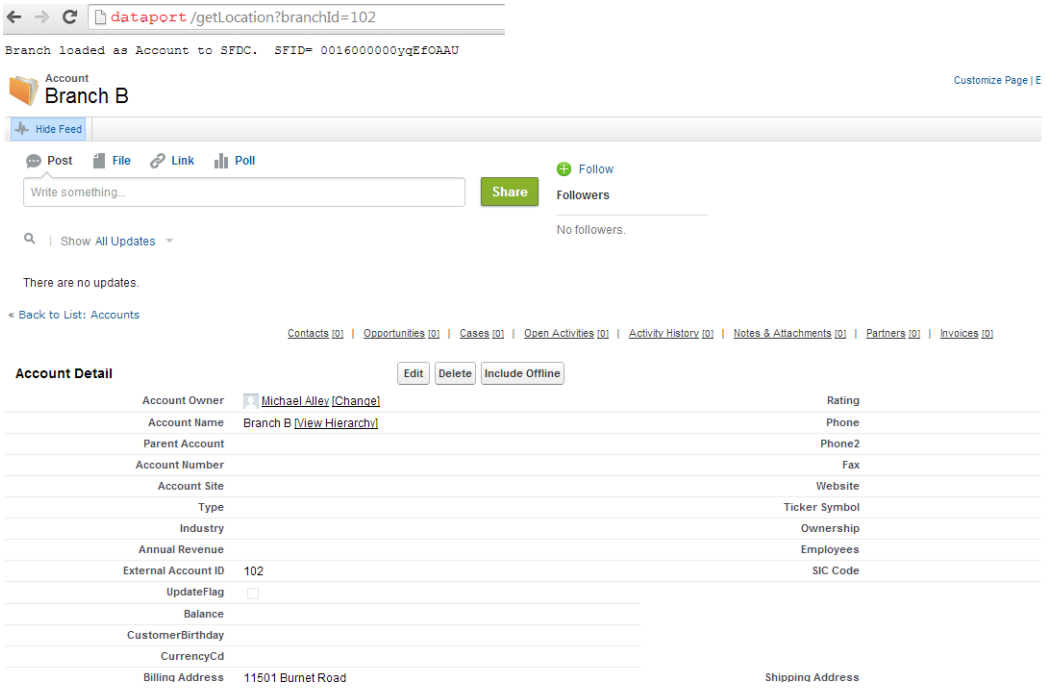


__ v. This completes the orchestration configuration. Deploy this project to your runtime environment according to the previous exercises.

__ w. Run the orchestration by directing a browser windows or HTTP Post utility to this URL:

```
of appliance>/getLocation?branchId=101
```

__ x. The output of the orchestration looks as shown in the browser. Also, notice a screen capture of the output inside Salesforce, in the Account object. Feel free to run more tests. Valid values for this service are branch IDs of 101, 102, and 103.



End of exercise

Exercise review and wrap-up

This lab is built around illustrating IBM App Connect Professional's ability to expose and work with REST Based services as part of an infrastructure that integrates seamlessly with software as a service, in this case it is with the Force.com API exposed by Salesforce.com.

Exercise 7. Developing with OData

Estimated time

00:45

Overview

OData as an integration standard is gaining more traction in the marketplace today. Vendors like Salesforce.com and SAP are recommending OData as an integration standard into their respective platforms. For more information, see (http://www.odata.org/blog/salesforce-external-object-integration-using-lightning-connect-with-odata/ and https://blogs.sap.com/2017/03/13/what-is-odata/). The main appeal of OData as a standard is that it allows for integration into backend systems without having to persist data in multiple places.
General information about the OData standard can be found here: https://www.odata.org/
The focus of this lab is meant to give some practical knowledge of how to use an App Connect Professional Template Integration Project (TIP) for OData exposure of a backend database.

Objectives

After completing this exercise, you should be able to:

- Create an application by using OData

Introduction

This lab assumes some basic knowledge of App Connect Professional. If you are new to App Connect Professional, it is recommended that you look at the tutorials on Developerworks to get you started:
https://developer.ibm.com/integration/docs/app-connect/tutorials-for-ibm-app-connect/migrating-account-data-from-a-flat-file-to-salesforce-com/

As soon as the TIP is implemented, you set up an external data source in Salesforce that points to the OData service and create an external object that references the data source. In this way, you can view that external data natively within the Salesforce UX.

The overall use case is about taking a database table that is running on MySQL, and expose that as OData. The MySQL table represents shipments that are being stored in a backend system. You want to be able to see this shipment data within your Account object layout inside of Salesforce.com.

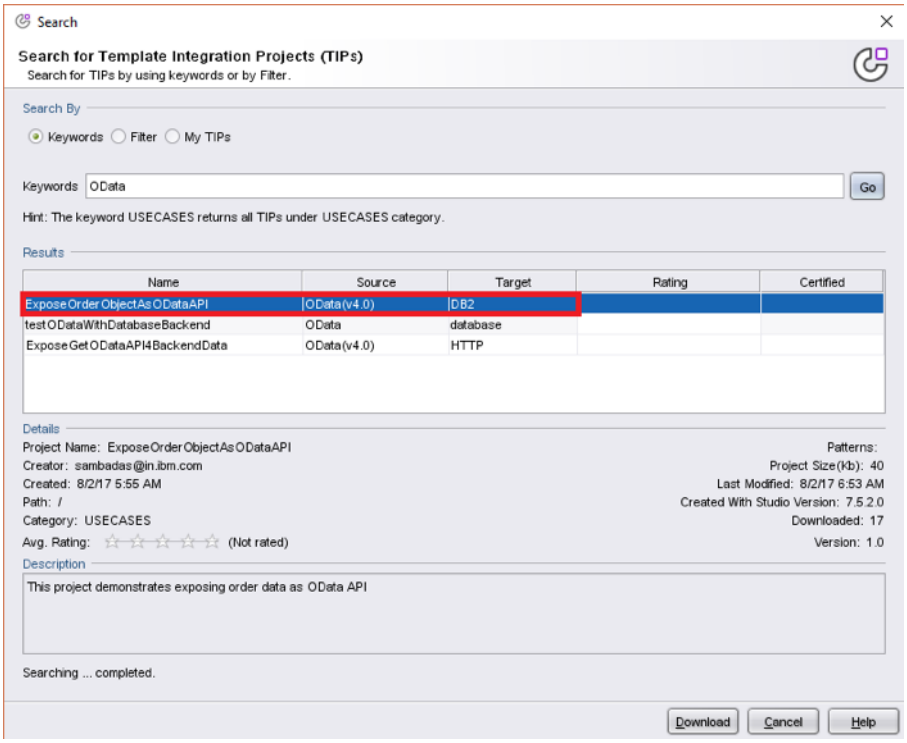
Exercise instructions

7.1. Download the OData TIP

__ 1. Open the App Connect Studio. Create a project by clicking the **Create Project** icon on the splash screen or by clicking **File > New > New Project**.

__ 2. Ensure that you are logged in to the TIPs repository. In the lower-right corner, you should see that you are logged in already as the **TIPSReadOnly** ID. If you are not logged in, click the **Login** button and accept the default credentials for the **TIPSReadOnly** ID. If you are using App Connect Live, you are automatically logged in.

__ 3. In Studio, go to **Solutions > Search for TIPs**. A search window opens where some text can be entered. Enter: `OData`
After a few seconds, a list of TIPS is returned. Select the **ExposeOrderObjectasODataAPI** Tip as indicated.



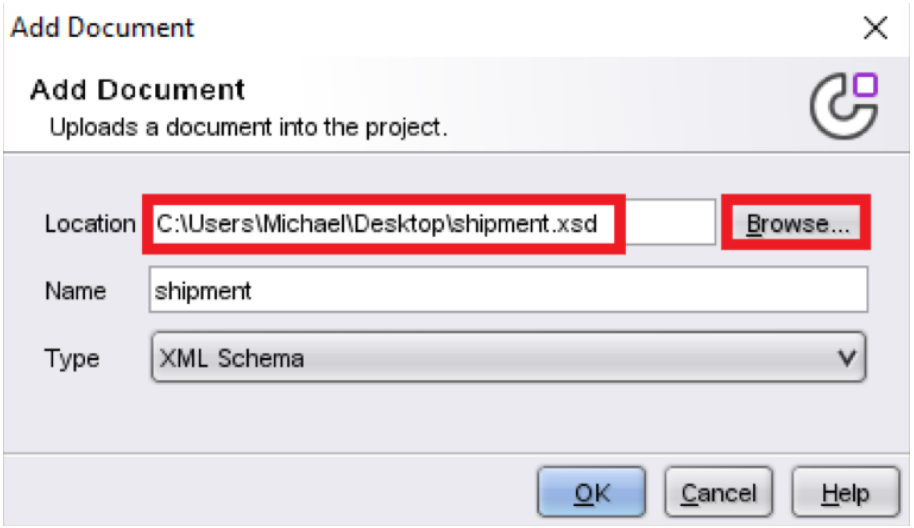
__ 4. Select a download location for the TIP. The TIP downloads and then opens when it is ready. When the TIP opens, it starts in the configuration wizard. You can step through the wizard to see what each main step is, but for this lab, you perform explicit steps to configure the TIP. Click the **Close** button, which is at the lower-right corner of the configuration wizard. Upon doing so, it takes you to the orchestration.

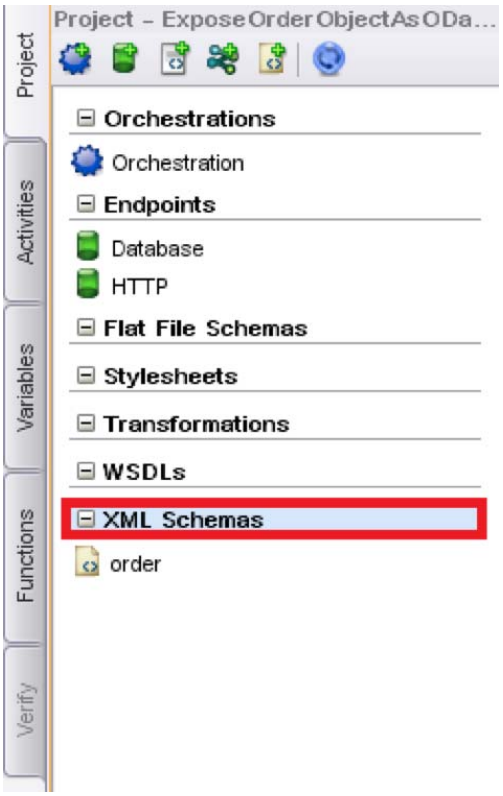
__ 5. Do not be concerned about the amount of activity in the orchestration, as your focus is specifically on the key areas that you need to modify. Before you continue, you must download the XML schema for the shipment object. The TIP must have a schema for the object that you are exposing. Download this file locally, you are going to use it shortly: https://github.com/ibm-cloudintegration/techguides/blob/master/shipment.xsd.

7.2. Configure the OData TIP

__ 1. Before you continue, enable the Activity IDs inside this project to help ensure that you are modifying the proper activity as you go through each step. From the main menu in Studio, select: **Edit > Preferences > Orchestration**. Ensure that the **Show Activity ID** check box is selected. Click **OK** when complete.

__ 2. Add this schema to the App Connect Project by right-clicking the XML **Schemas** heading on the **Project** tab and selecting **Add Document**. Browse for the `Shipment.xsd` file that you downloaded. It appears in the list of schemas in addition to the order schema that came with the TIP.



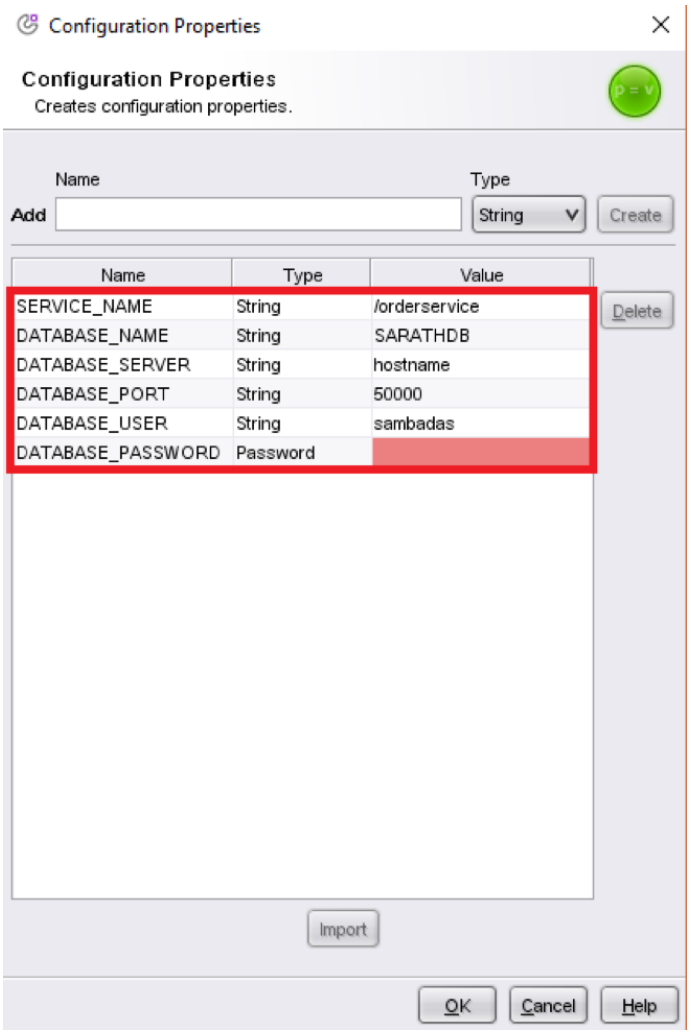


__ 3. Note the name of the root object in the schema that you created. The path to the URI is affected by making the name of the object that you selected. You encounter more on this subject later.

__ 4. Next, you need to configure the database endpoint by clicking the **Database** icon under **Endpoints** to open the database configuration screen.

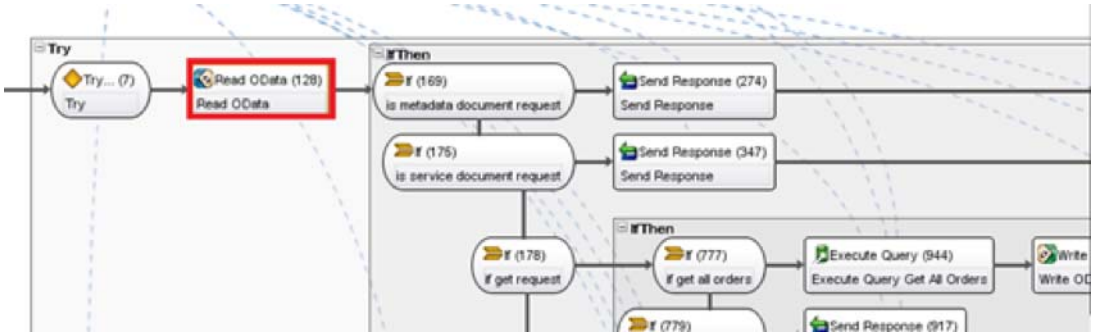
__ 5. Change the **Database Type** to **MySQL**. Notice that some of the parameters are listed with a yellow background. These are configuration properties. You must change them to reflect your database environment. To change the configuration properties, from the menu on the top of the screen on Studio, select **Project > Configuration Properties**. Replace the properties provided here. Click **OK** when done.

```
SERVICE_NAME = /expose (This is hard coded - this will be the root location
of the OData Service)
DATABASE_NAME=<Your DB Name>
DATABASE_SERVER=<Server Location of your MySQL DB)
DATABASE_PORT=<Your Port, usually for MySQL its 3306>
DATABASE_USER=<DB Username>
DATABASE_PASSWORD=<DB Password>
```



__ 6. Rename the **HTTP Request (activity ID #4)** from `orderService` to **`shipmentService`** by double-clicking the text box and typing the new name.

__ 7. Next, you configure the Read OData activity. In the Orchestration pane, click the **Read OData (activity ID #128)**:.



__ 8. You need to change the schema reference for the OData payload to reflect the **shipment** schema. In the checklist that appears on the left side of the screen, click the **Configure** step.

__ a. Click the **Browse** button to select the appropriate variable. The variable for the **Shipment** object doesn't exist yet, so you need to create it.

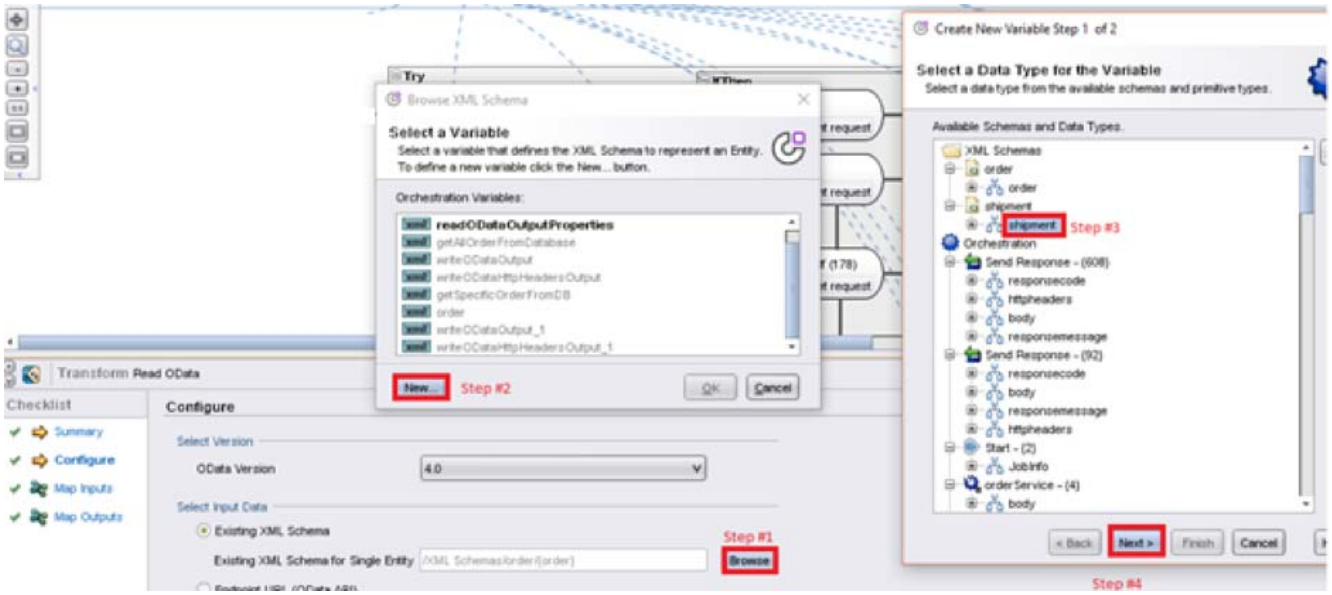
__ b. Click the **New** button in the window that comes up.

__ c. Select the **Shipment** variable from the list of variables.

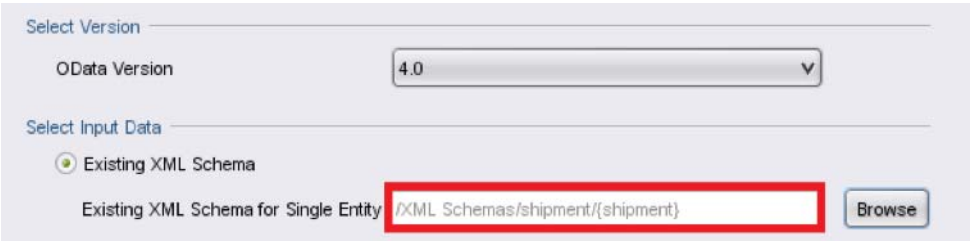
__ d. Click **Next**, click **Finish**, and then click **OK** to complete the process.

Note

The URI of your exposed OData services is going to end up behind the combination of the **SERVICE_NAME** configuration property that you set in step 10 and the plural value of the object that you defined in your schema. In this case, **shipment** made plural is **shipments**, so your final URL that you call is: `http://<appliance_dataip>/expose/shipments`



__ 9. When completed properly, the **Existing XML Schema** reference looks as follows:



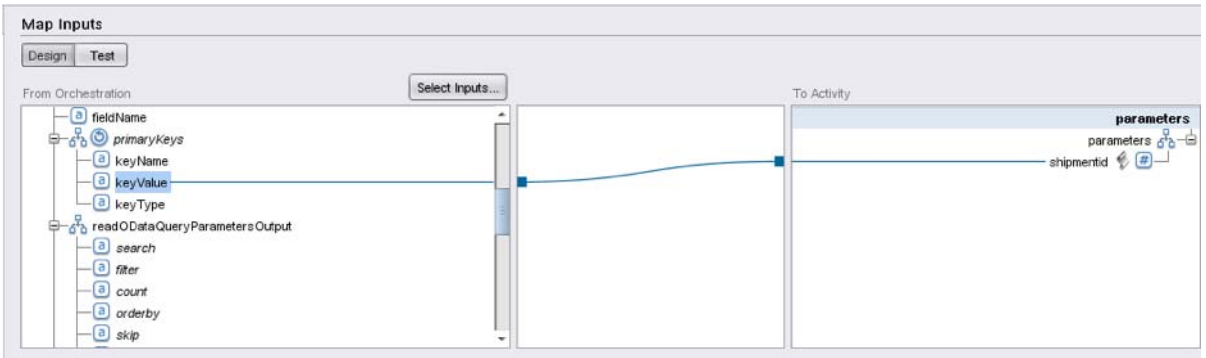
__ 10. Next step is to modify the **If/Then (activity ID #777)** to reflect the new object you are working with. Select the activity labeled **If get all orders**. Rename this activity to **If get all shipments**. Also, modify the right hand expression of the activity to reflect the proper plural name of the **shipment** object that you defined previously, which in this case, is **/shipments**. Here is the figure before updates:

Here is the figure after updates:

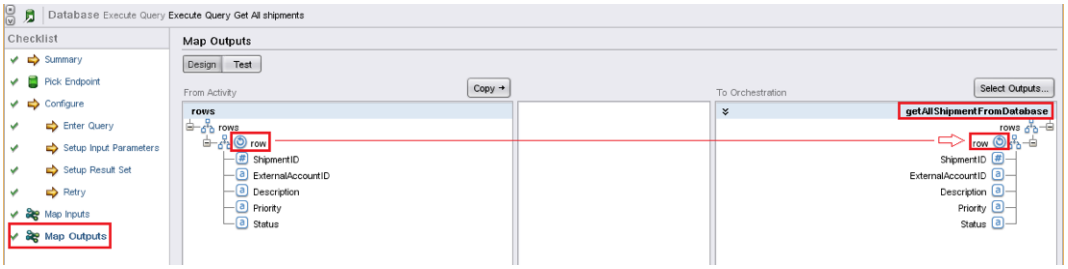
__ 11. Rename the **Execute Query (activity 944)** from `Execute Query Get All Orders` to: `Execute Query Get All shipments`

__ 12. Modify the query in the **Execute Query** activity to reflect the shipment table. In the activity checklist, click the **Enter Query** and change the query to be `Select * from shipment`. Click the validate query to update the query metadata from your database.

__ 13. In the **Map Inputs** step, drag over the primary key value from the OData header to the query. Find the *keyvalue* variable underneath the *primaryKeys* group. Right-click and click **Select One Occurrence** and then click **OK**. This is needed because the *primaryKeys* group is a repeating variable, and you cannot map a repeating object to a single object. Drag the singleton variable over the **ToActivity** side into the *shipmentid* variable. You receive a warning that there is a data type mismatch. If you want to resolve the type mismatch, you can drag a **Number** function from the **Functions** tab, and then link those together, but it is not mandatory.



__ 14. Update the query result set metadata in the **Map Outputs** step and map those results to a variable. Click **Map Outputs**. Change the variable name on the output side by clicking it and changing it to: `getAllShipmentFromDatabase`
Drag the row object on the left side to the row object on the right side. It maps over all of the variable automatically.



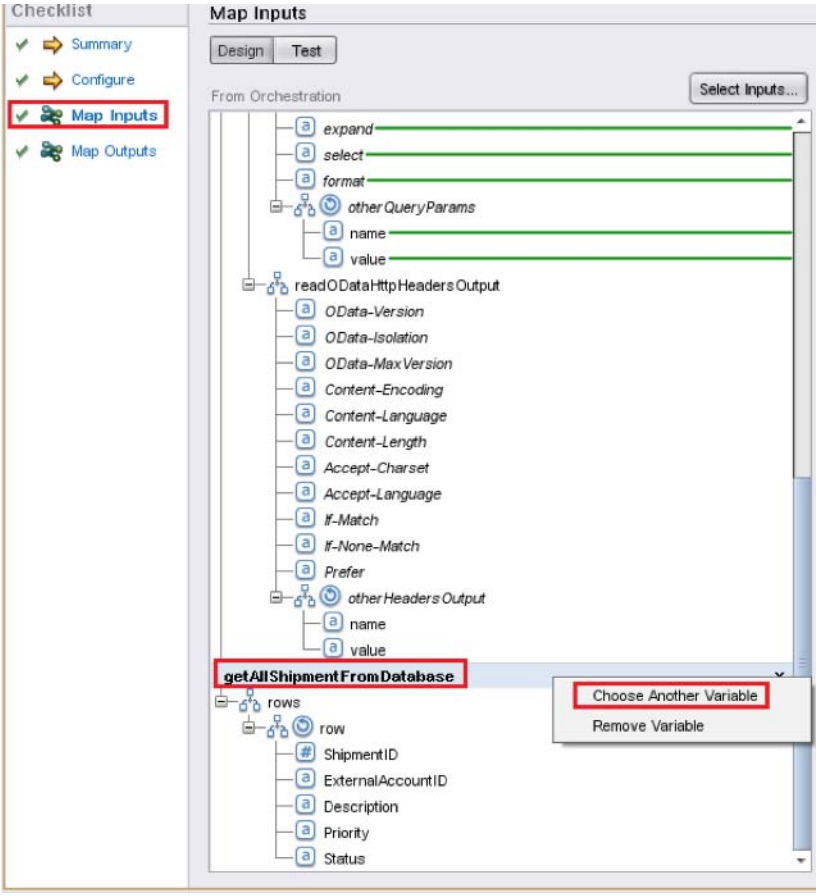
__ 15. The next activity to modify is the **Write OData (Activity #971).** In the **Configure** step, modify the schema being used to the root object of the shipment schema. This is similar to what you did previously, but you can use the existing variable.

__ a. Click the **Browse** button to select the appropriate variable. Select the existing **shipment** variable from list.

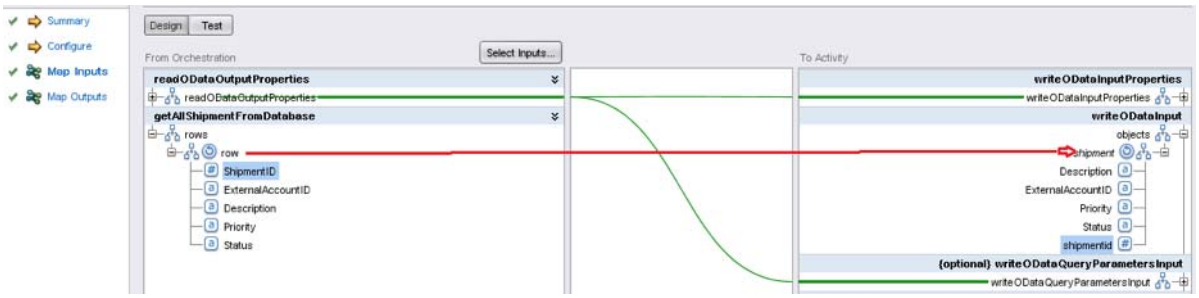
__ b. Click **OK** to complete the process.

__ 16. On the **Map Inputs** step of **Write OData (Activity #971)**, you then map the result set of the Shipment query to a variable on the output side. Complete the following steps:

__ a. Replace the variable for the database query response by scrolling down and replacing the original variable for the database query with the *getAllShipmentFromDatabase* variable defined in the previous step. Do this by scrolling down to the bottom of the left pane of the Map Inputs screen and find the down arrow. Click it and select **Choose Another Variable**. Select **getAllShipmentFromDatabase**.



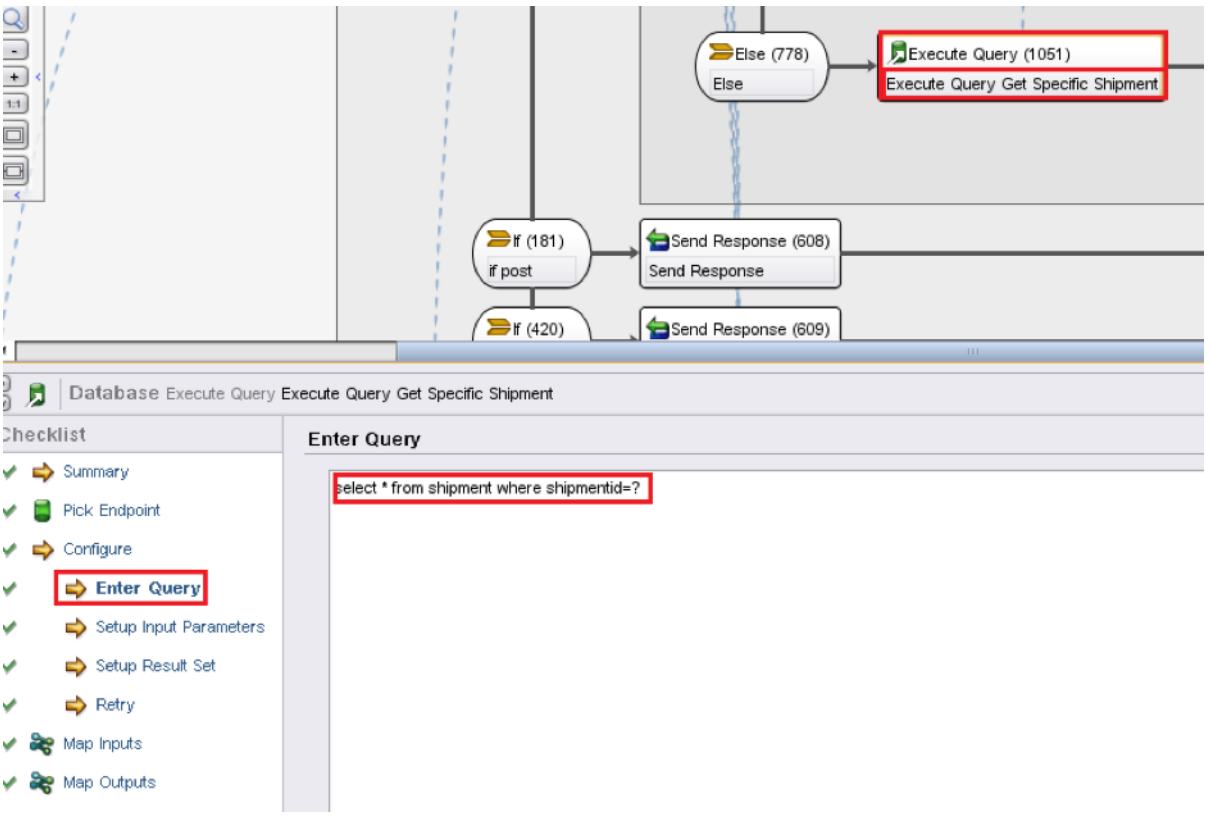
__ b. Now you can map over the **row** object over to the **shipment** object in the **Write OData Input** variable by using a click or drag and drop. When complete, the green lines automatically map from the object on the left to the object on the right.



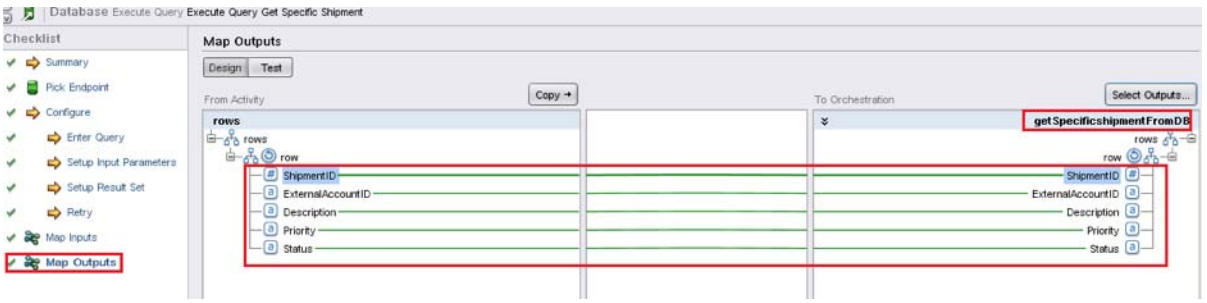
__ 17. Next, you need to change the **Execute Query (Activity #1051)**, which controls how the TIP extracts each **shipment** by its primary key, **shipment id**.

__ a. First, click the activity and change the activity name to `Execute Query Get Specific Shipment`.

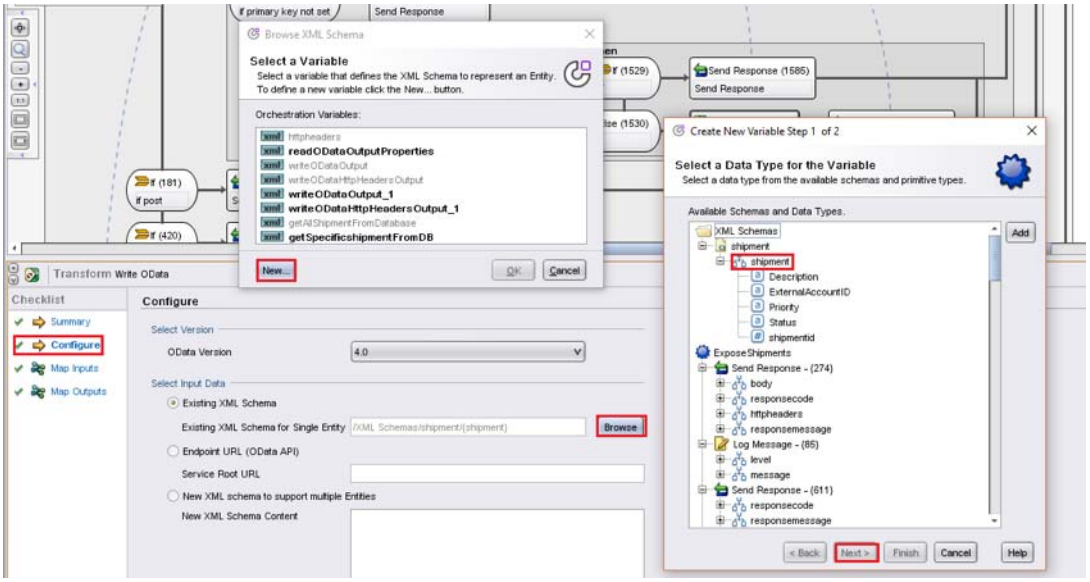
__ b. Next, click the **Enter Query** entry from the checklist. Change the query from to `Select * from shipment where shipmentid=?`. Click the **Validate Query** button in the lower-right corner.



__ 18. Click the **Map Outputs** step in the checklist. Map the data over to a new variable called `getSpecificShipmentfromDB`. This action automatically maps the fields from the input to the output side as follows:



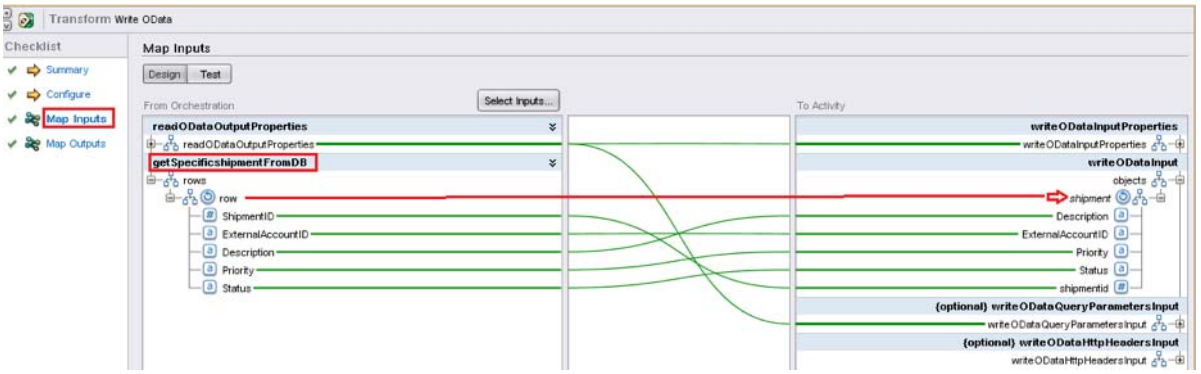
__ 19. Change the **Write OData** (**Activity #1104)** to use the shipment schema. Click the **Browse** button in the **Configure** step of the checklist, then **New**, then browse for the **shipment** variable, and then click **Next > Finish.**



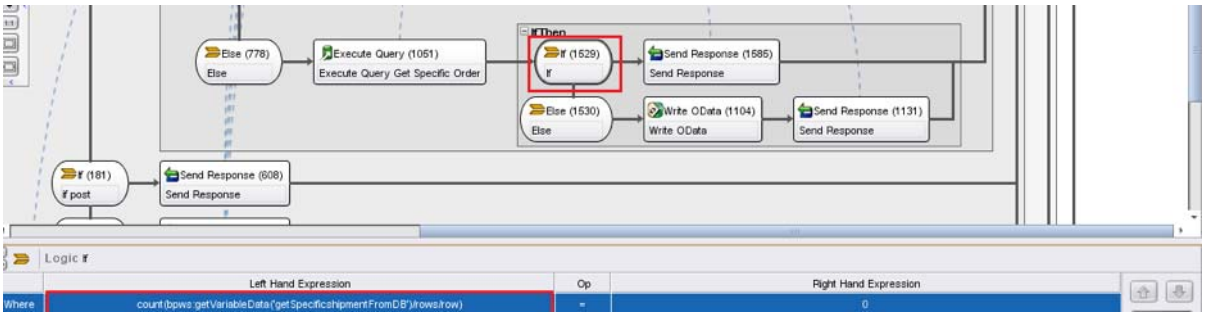
__ 20. Now you need to change the **Map Inputs** of **Write OData** (**Activity #1104)** to map the data over from the variable that holds the output for the Database Query to the OData activity. Follow these steps:

__ a. Replace the variable for the Database query response by scrolling down and replacing the original variable for the database query with the *getSpecificShipmentFromDatabase* variable defined previously. Do this by scrolling down to the bottom of the left pane of the Map Inputs screen and find the down arrow. Click it and select **Choose Another Variable**. Select **getAllSpecificFromDatabase**.

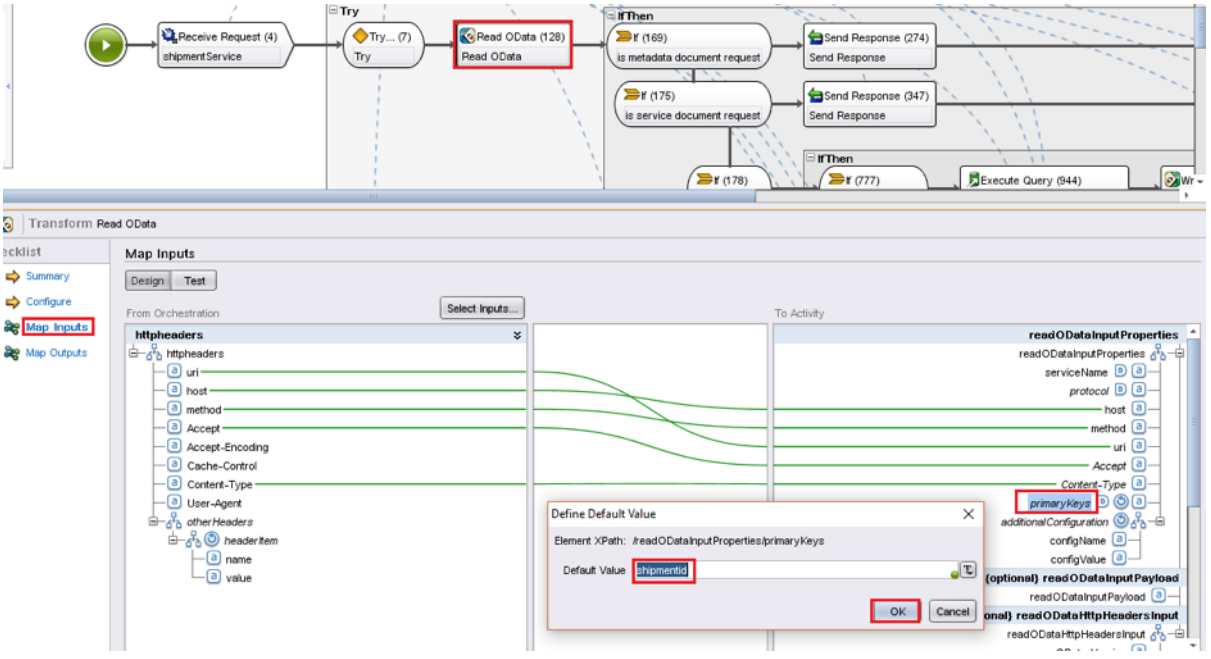
__ b. Now you can map over the **row** object over to the **shipment** object in the **Write OData Input** variable by using a click or drag and drop. When complete, the green lines automatically map from the object on the left to the object on the right.



__ 21. Change the **IfThen (Activity #1529)** to evaluate the proper response variable. Click the activity and then change the left expression to the following code:
`count(bpws:getVariableData('getSpecificshipmentFromDB')/rows/row)`



__ 22. Ensure that the Primary key in the main **Read OData (Activity#128)** is set to properly reflect the primary key that is defined in the previous step. Click **Read OData (Activity#128)** and go to the **Map Inputs** step. Find the field on the output side called `primaryKeys`. It is set to a default value that came with the original TIP. Change it by right-clicking it, and then setting it to the value of: `shipmentid`
Click **OK** to complete the process.



7.3. Deploy to your runtime environment and test

__ 1. Save your project and then deploy to your runtime environment.

Hint

Rename your project to something meaningful for you. You can also rename the orchestration if you like by right-clicking the orchestration on the **Project** tab and then clicking **Rename**.

__ 2. Publish the project to your runtime.

Hint

Enable logging on the orchestration to assist with troubleshooting if something doesn't work right before starting it.

__ 3. Start the project.

__ 4. Test by using a tool like Postman. There are a few basic commands you execute to test things out and get a feel for what the OData integration can do. Running this orchestration by using App Connect Live (SaaS) requires a few extra arguments passed to the GET to process properly. For example, use the following URL to call this orchestration from Live: https://provide.castiron.ibmcloud.com:443/env/<yourEnvironment>/expose
You also must pass the login credentials for your live instance either as HTTP headers or as query parameters (ciPassword, ciUser).

__ a. First, send an HTTP GET to the root of the service at this address. It provides a response like the one provided here:

```
{
    "@odata.context": "http://localhost/expose/$metadata",
    "value": [
        {
            "name": "shipments",
            "url": "shipments"
        }
    ]
}
```

__ b. To see the metadata description of shipments, you can pass the following command: http://<runtime_data_ip/expose$metadata

It responds with the structure of the data:

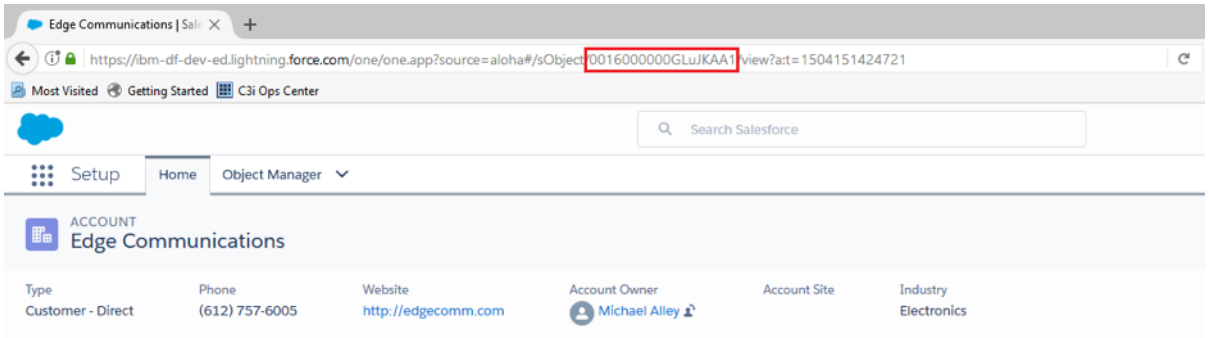
```
<?xml version='1.0' encoding='UTF-8'?>
<edmx:Edmx Version="4.0"
xmlns:edmx="http://docs.oasis-open.org/odata/ns/edmx">
    <edmx:DataServices>
        <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm"
Namespace="com.ibm.castiron.odata.v4.Expose">
            <EntityType Name="shipment">
                <Key>
                    <PropertyRef Name="shipmentid"/>
                </Key>
                <Property Name="Description" Type="Edm.String"
Nullable="false"/>
                <Property Name="ExternalAccountID" Type="Edm.String"
Nullable="false"/>
                <Property Name="Priority" Type="Edm.String"
Nullable="false"/>
                <Property Name="Status" Type="Edm.String"
Nullable="false"/>
                <Property Name="shipmentid" Type="Edm.Int32"
Nullable="false"/>
            </EntityType>
            <EntityContainer Name="ExposeContainer">
                <EntitySet Name="shipments"
EntityType="com.ibm.castiron.odata.v4.Expose.shipment"/>
            </EntityContainer>
        </Schema>
    </edmx:DataServices>
</edmx:Edmx>
```

__ c. Load some data into your shipment table by using whatever method you are comfortable with. Issue this command in Postman to view this data by using OData: `http://<runtime_data_ip>/expose/shipments`. The response looks similar to the following segment:

```
{
    "@odata.context": "http://localhost/expose/$metadata#shipments",
    "value": [
        {
            "Description": "Automated via IBM Cloud Integration for
Salesforce",
            "ExternalAccountID": "0016A0000037M02QAE",
            "Priority": "Standard",
            "Status": "Started",
            "shipmentid": 1
        }
    ]
}
```

__ 5. If all of these work as expected, you have completed this part of the lab.

__ 6. Before moving on to the next step, It is important that you load the Account ID data into your shipment table that matches one of the accounts you have inside of Salesforce. To find the account ID of an account, simply select one of the accounts and look at the URL. Here is an example that uses one of the sample accounts **Edge Communications** that is created with all demonstration instances. Use the same for your sample also.



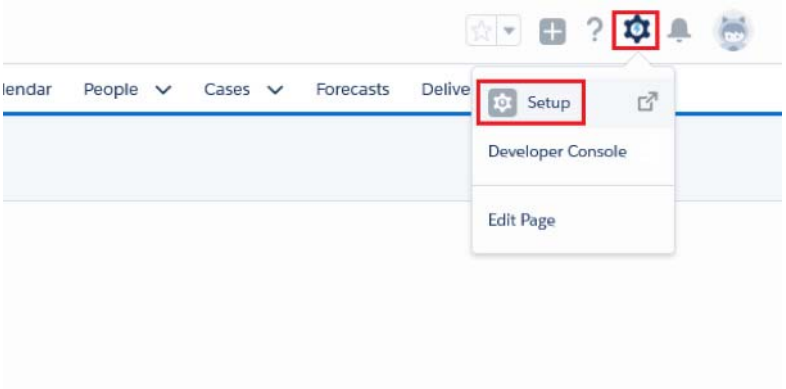
__ 7. The Account ID is highlighted in red in the URL in the browser. This is the Salesforce ID. Update your **ExternalAccountID** column **Shipment** table in your database with this value. Load the value exactly, it is case-sensitive. Use copy and paste for best results.

7.4. Set up external object in Salesforce

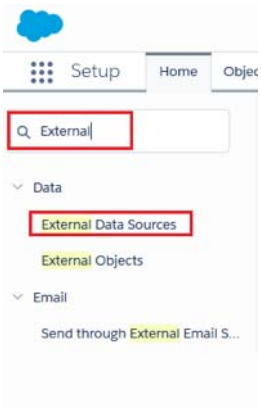
__ 1. Log in to your Salesforce.com instance.

__ 2. These instructions assume you are using the Salesforce Lightning Experience. If you are not, then you can switch to the Lightning experience by using **<yourname> > Setup > Switch to Lightning Experience**.

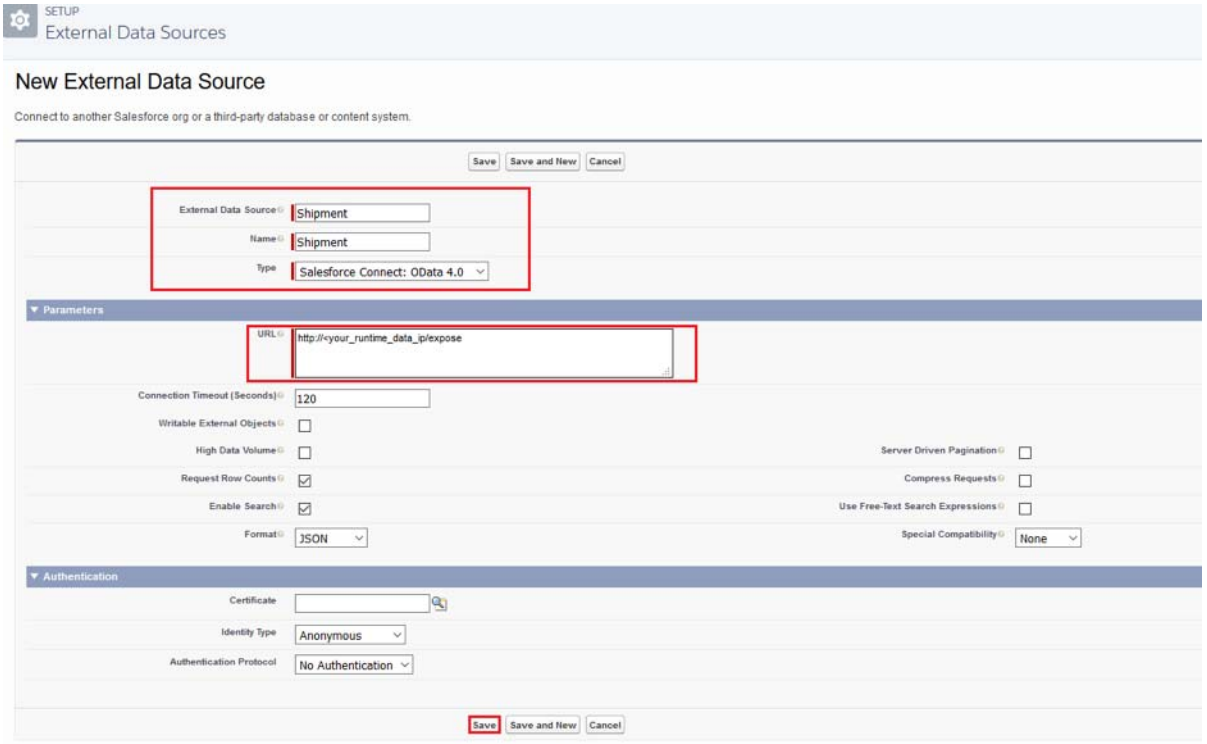
__ 3. In the upper-right corner, click the cog wheel and then click **Setup**.



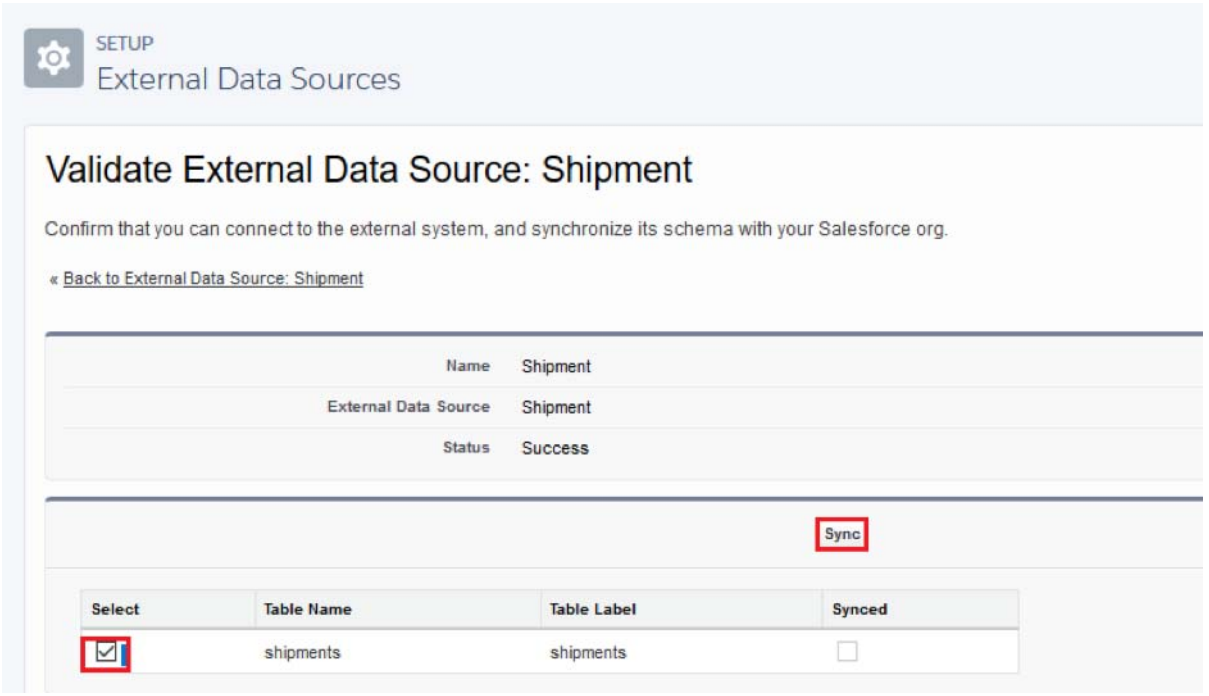
__ 4. From the search window on the left, type the following text: `External`
It returns the following information. Click the **External Data Sources** option below.



__ 5. Click the **New External Data Source** button to create a data source. Enter in the information. Replace the URL with the location of your runtime environment and the orchestration endpoint you used to test (for example http://<your_runtime_data_ip>/expose). Ensure the type is set to **Salesforce Connect: OData 4.0.** Click the **Save** button. If you are using AppConnect Pro Live, then you can change the **Identity Type** to **Per User** and then select the **Authentication Protocol** to **Password Authentication**. Enter the IBM ID for AppConnect Pro here to avoid having to pass the **ciUser** and **ciPassword** query parameters in the execution URL. Typically, in normal runtime scenarios an application ID type user would be used here rather than a personal ID. You can set up more login IDs inside of AppConnect, but for this lab, a personal ID works.

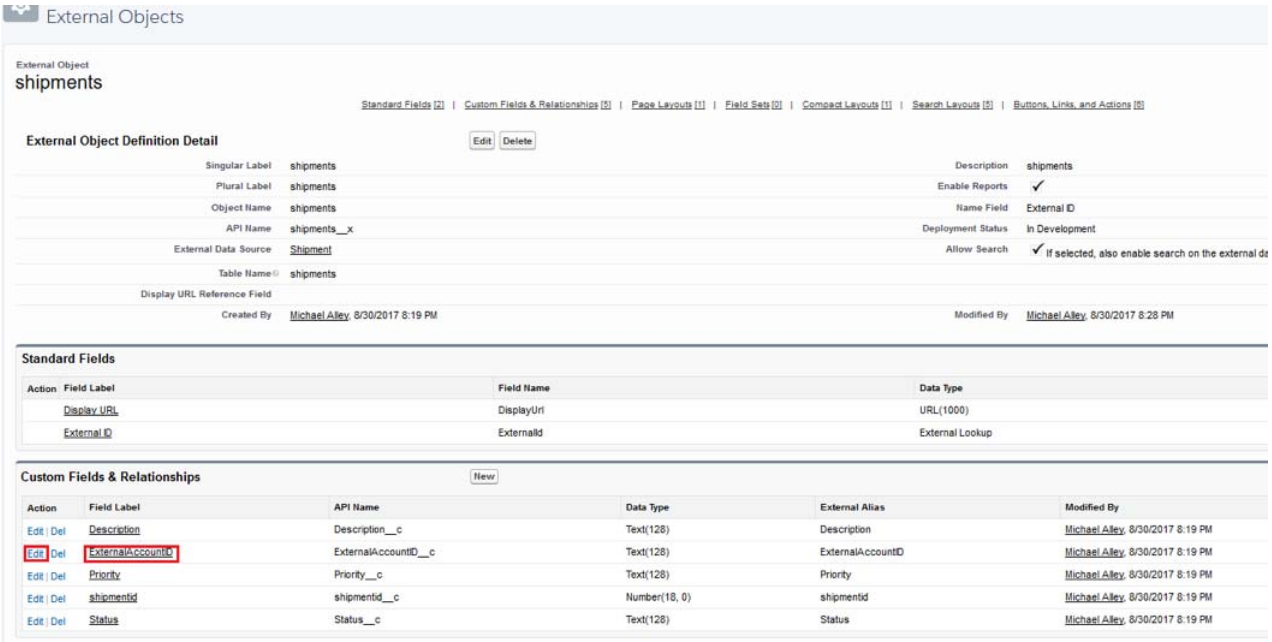


__ 6. As soon as the save is complete, a new button called **Validate and Sync** is visible. Click the button to continue. Salesforce reaches out to the OData endpoint to test the connectivity. When that is complete. Select the check box under the **Select** heading then click the **Sync** button to bring in the metadata. As soon as that is complete, you can now add it to the view of your Account object such that it is visible.



__ 7. From the top menu, select the arrow button just to the right of **Object Manager**. In the submenu, click the newly created object **shipments**.

__ 8. Here you see the object that you created. Find the **ExternalAccountID__c** field. Click **Edit**.



__ 9. Click **Change Field Type**.

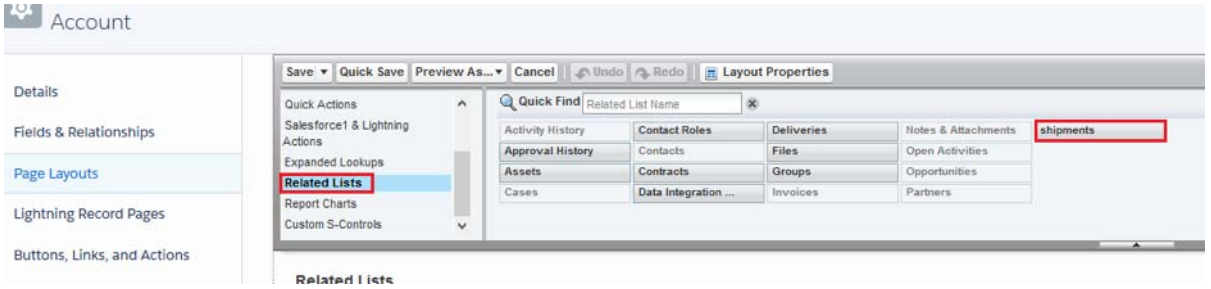
__ 10. From the list, select the **Lookup Relationship** option**.** Click **Next** in the lower-right corner of the screen.

__ 11. From the drop-down, select the **Account** object. Click **Next**. Then, click **Save.**

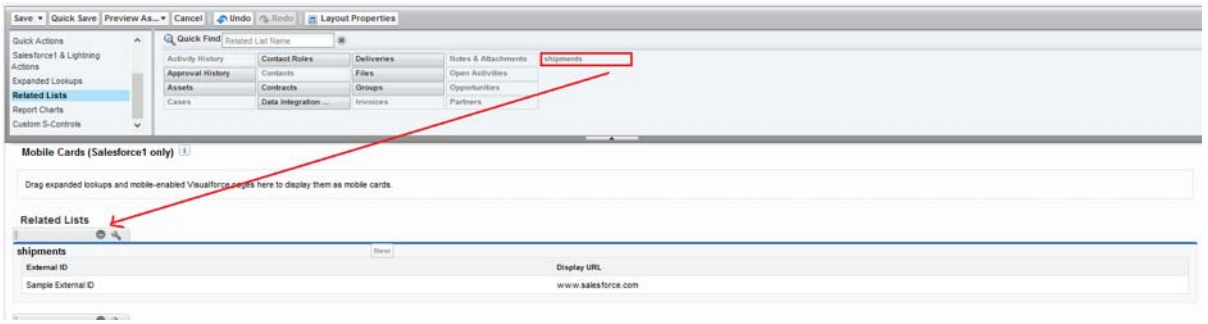
__ 12. From the top menu again, select **Object Manager,** then click **Account.**

__ 13. Select **Page Layouts**. You see several layouts. Here, click **Account Layout**.

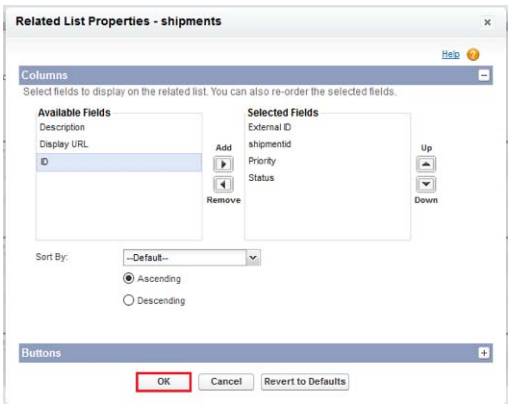
__ 14. At the top of the Account Layout screen, there is the Account Layout box. From the scroll bar inside that box, select the option called **Related Lists**. Note that the **shipments** object is there.



__ 15. Click and drag the **shipments** object to the top of the **Related Lists** section. It adds the **shipments** external object to your account object.

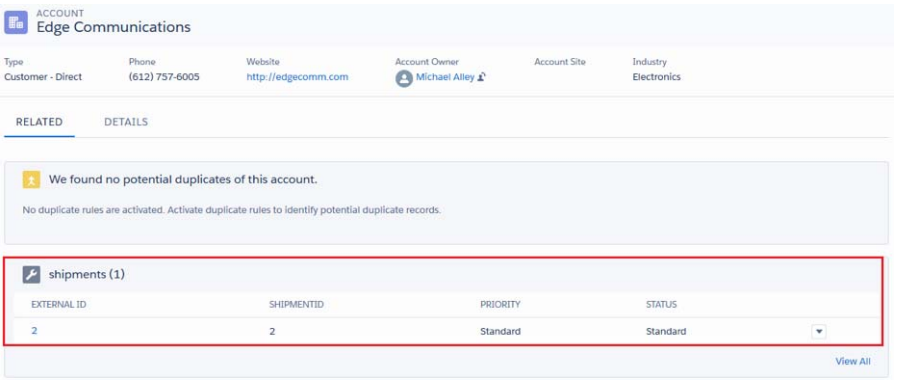


__ 16. Click the small wrench icon on the object you just dragged. You can then configure the layout of what is displayed. Using the **Add**, **Remove**, and **Reorder** buttons, arrange the layout to look as such. Click **OK** when complete:



__ 17. Click the **Save** button in the upper-left corner of the layout box.

__ 18. Return to the **Accounts** screen through the main menu. Select the Account object. You should then see your **Shipment** object that is related to your **Account** when it is complete



End of exercise

Exercise review and wrap-up

In this exercise, you exposed a database table as an Odata service that is consumed by Salesforce Connect and linked that to a standard object inside of Salesforce.

Exercise 8. Combining multiple sources with IBM App Connect Designer

Estimated time

00:60

Overview

In this exercise, students combine data from multiple sources and process for each record returned, and also check for and indicate if a row does not exist.

Objectives

After completing this exercise, you should be able to:

- Combine data from multiple sources
- Process each record that is returned
- Verify that records exist

Introduction

A sales representative for Sliding Otter Enterprises has a list of contacts in Salesforce. However, the email address for many of those leads lies in Insightly, a different CRT. The sale representative would like to gather up those email addresses from Insightly and report on any contacts that do not exist in Insightly as well.

Requirements

- You have a Salesforce developers account
- You have an Insightly developers account
- You have an IBM Bluemix Account
- You have a Slack Account

Exercise instructions

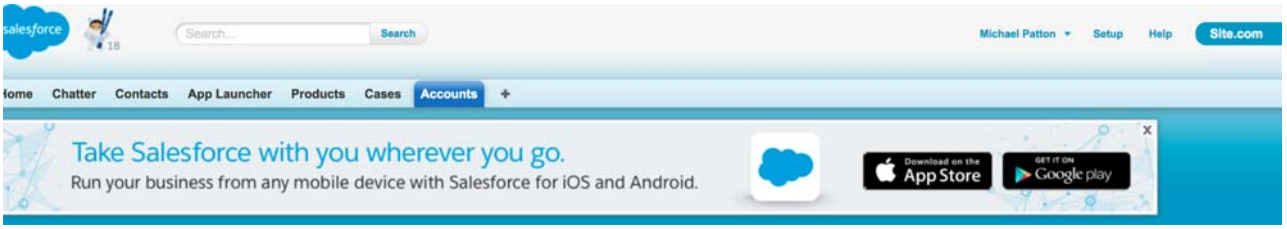
8.1. Prework

__ 1. Set up contacts in Salesforce.

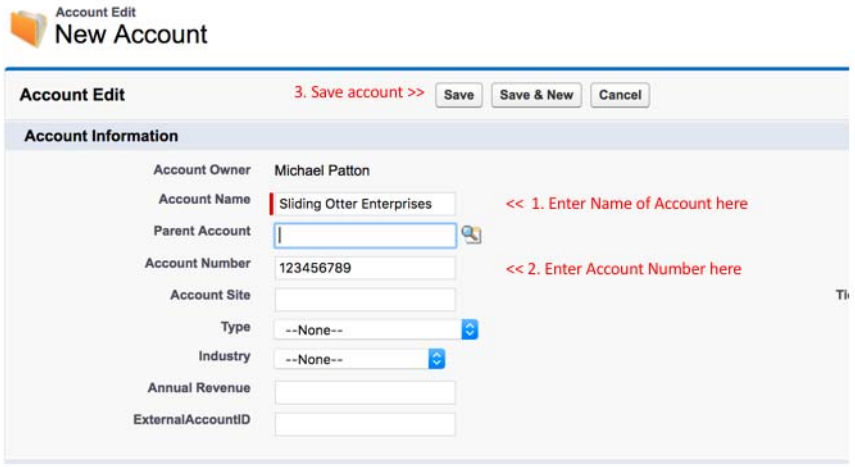
__ a. Navigate to your Salesforce developers account and log in.

https://developer.salesforce.com/

__ b. Navigate to the Accounts section of your account

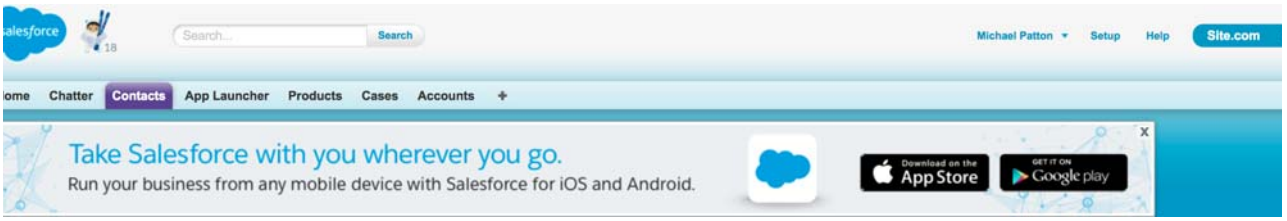


__ c. Create an account, name it anything that you like, and add an **Account Number** (Note, this is different from the Account ID, which is auto generated by Salesforce).

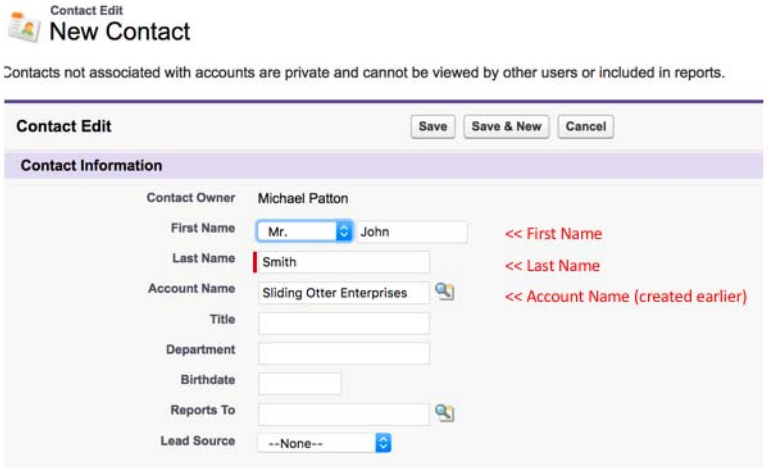


__ d. Make a note of the Account Number, you need this later.

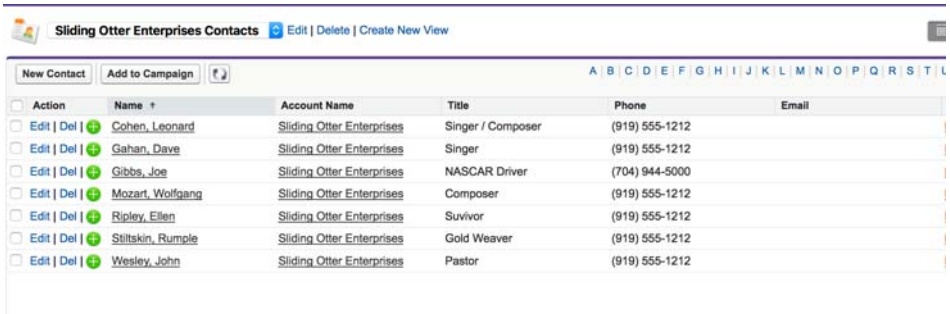
__ e. Navigate to the Contacts section of your account.



__ f. Create five or more contacts and ensure that each contact is in the account that is created in step 1c. Five contacts should be good for this exercise. Do not reuse last names. (This lab is not going to address multiple records with an identical field).



__ g. Check that your contacts are all listed together. Note, there is no email that is listed for any of the contacts.

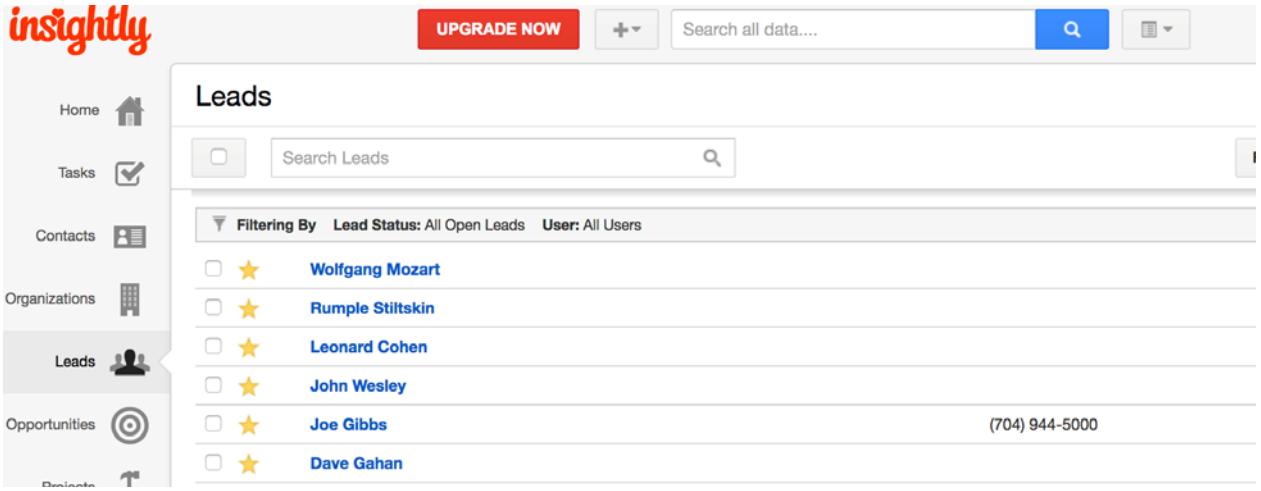


__ 2. Set up Leads in Insightly.

__ a. Navigate to your Insightly developers account and log in.

https://www.insightly.com/product/connect-to-apps/for-developers/

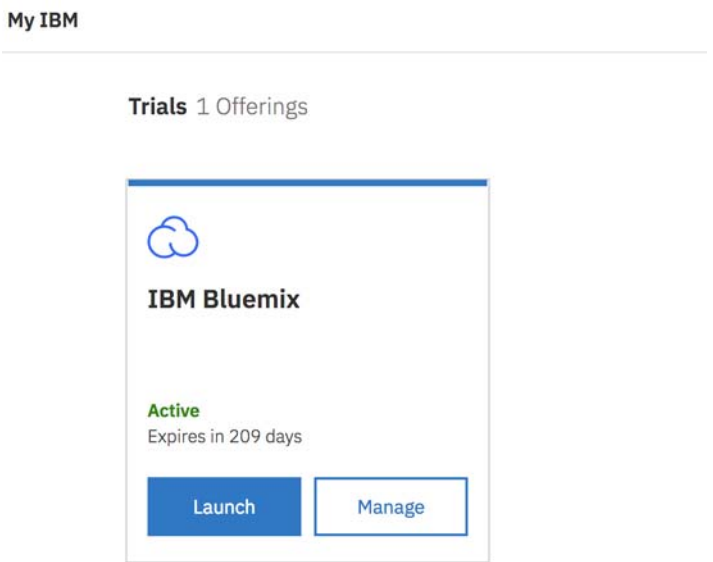
__ b. Navigate to the Leads on the left menu bar, and add all but one of your contacts again into Insightly. You only need the first name, last name, and an email address.



__ 3. Configure App Connect to connect to both Salesforce and Insightly.

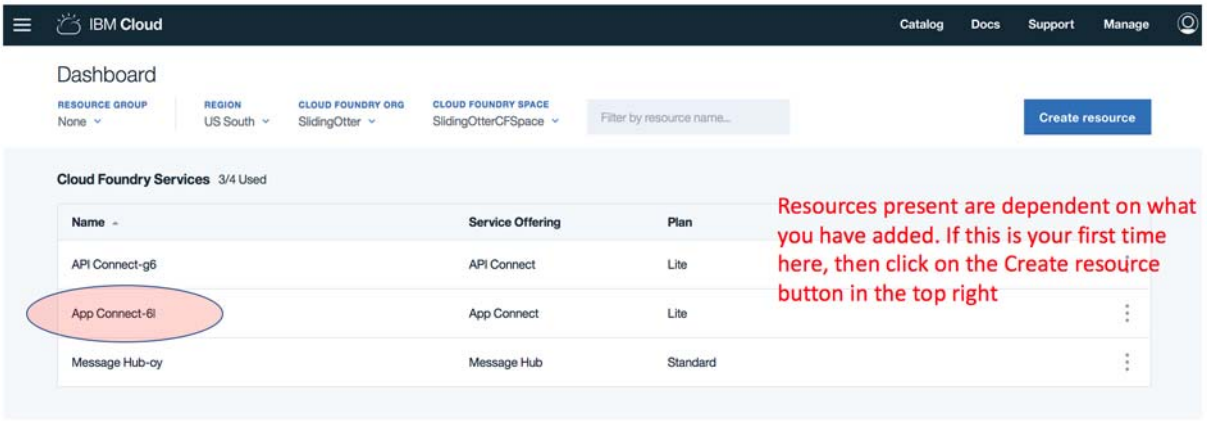
__ a. Log in to your IBM Cloud account.

https://myibm.ibm.com/products-services/products



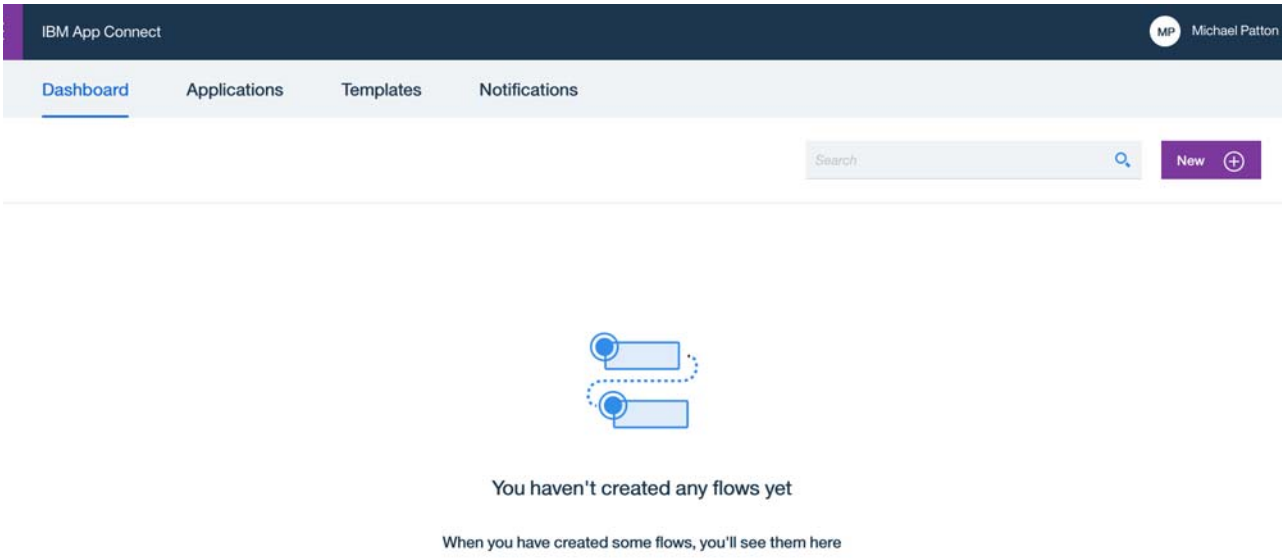
__ b. Launch the IBM Cloud resource.

__ c. Launch App Connect on IBM Cloud (If you do not have App Connect, create the resource from the button in the upper-right corner).

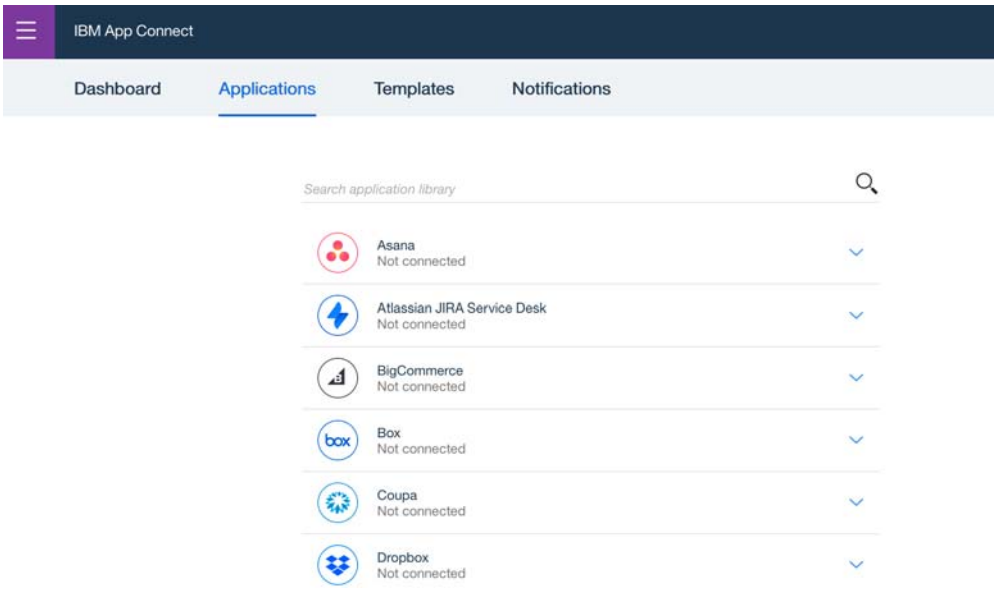


__ d. Click **Launch App Connect** again on the next window.

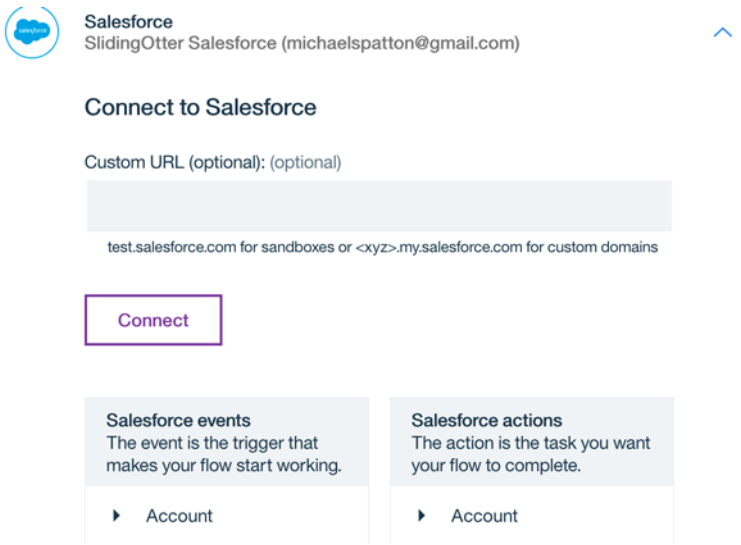
__ e. The page might indicate that you do not have any flows.



__ f. Click the **Applications** tab (Second from left, next to the Dashboard). You are going to configure the Salesforce, Insightly, and Slack applications to communicate with App Connect.



__ g. Scroll down to the Salesforce Application, expand, and select **Connect**.



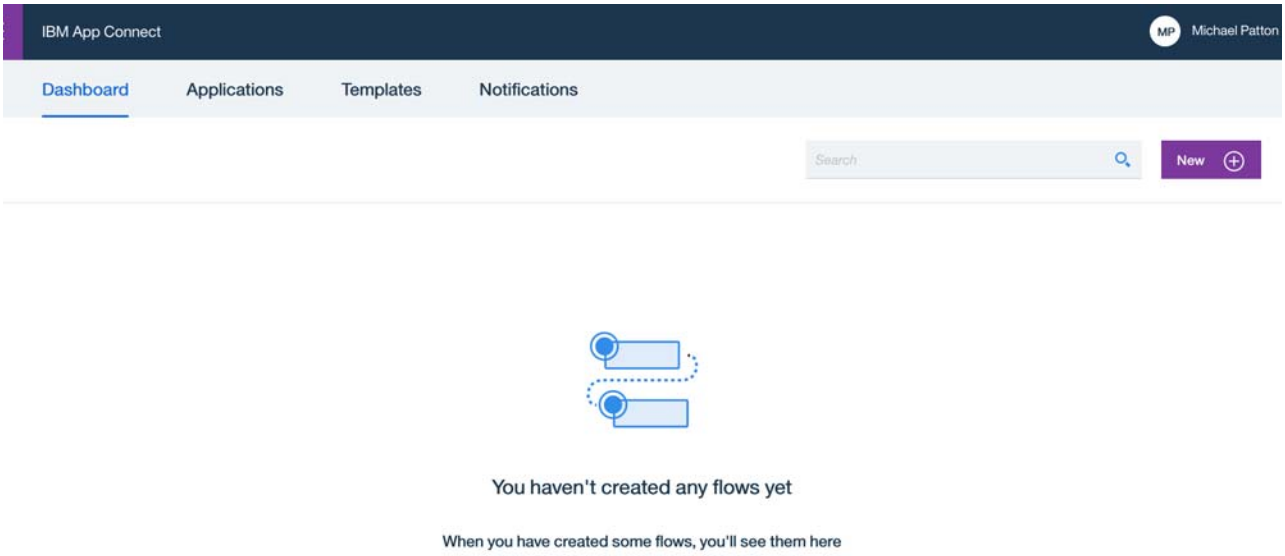
__ h. Scroll up to the Insightly Application, and repeat the earlier steps.

__ i. Scroll down to Slack, and repeat the steps again.

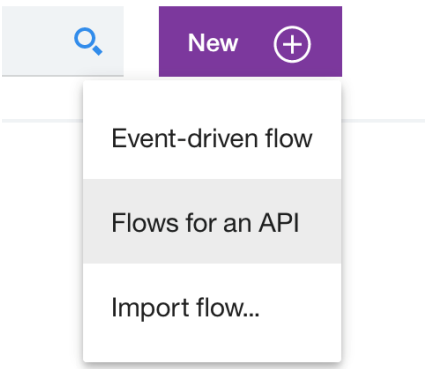
You are now ready to start building App Connect Flows.

8.2. Build App Connect Flow

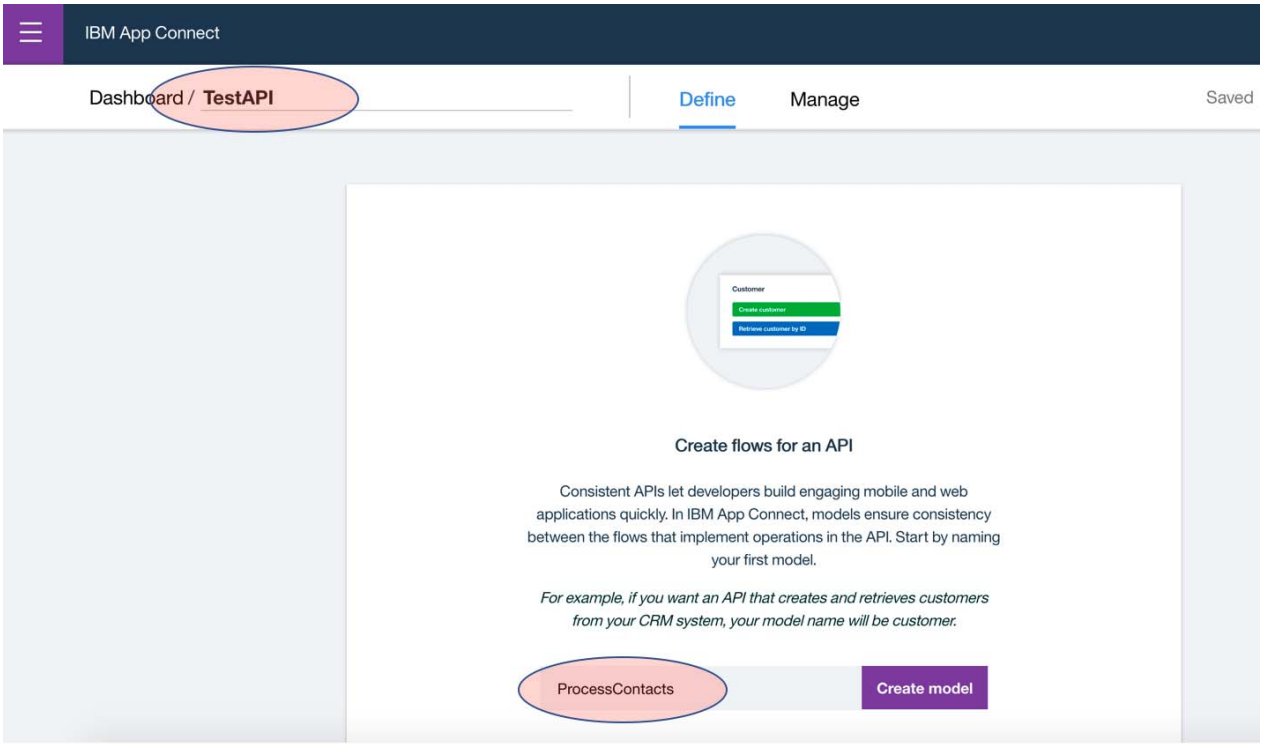
You should be on App Connect (you might not have any flows there already).



__ 1. Click **New** and select **Flows for an API**.

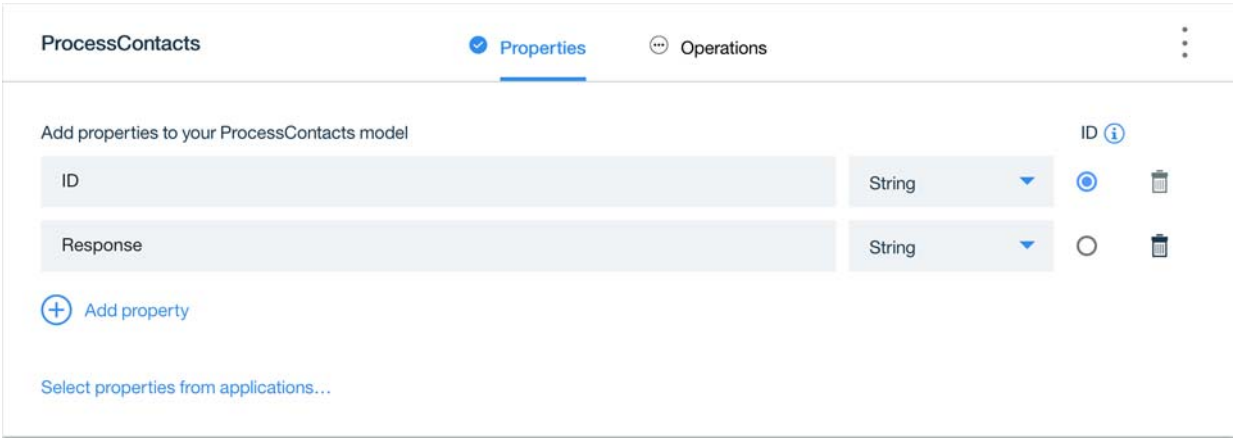


__ 2. Enter **TestAPI** for the Dashboard, and **ProcessContacts** for the Model. *ProcessContacts* become part of the API Path to invoke this flow a little later.

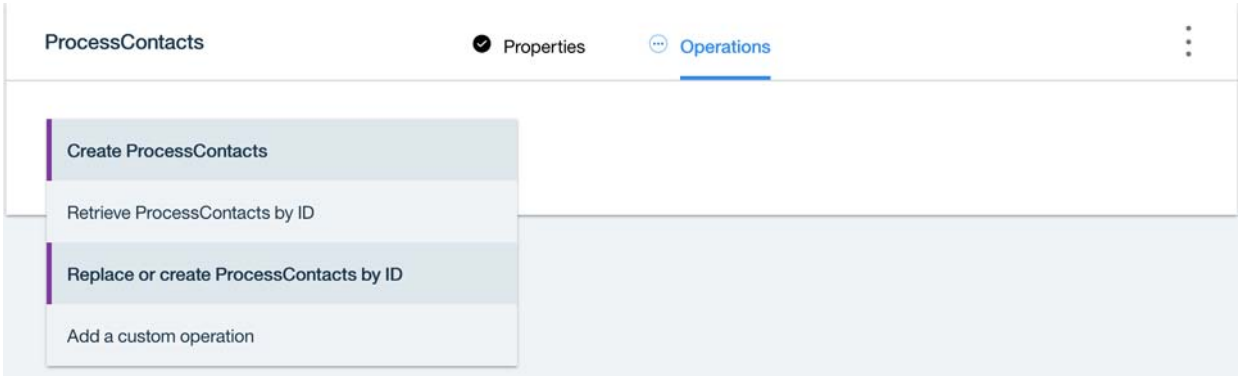


__ 3. Click **Create model**.

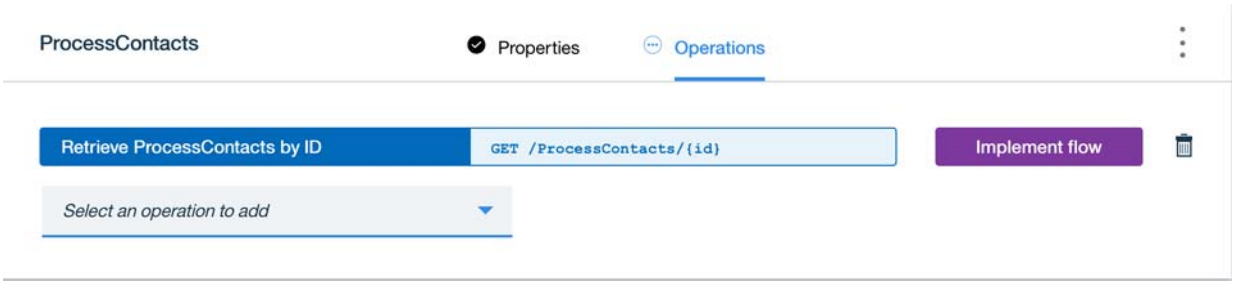
__ 4. Create two properties for this flow, an ID field that is used to find records, and a “Response” field for when the flow completes. The ID field has the radio button selected, as it is a part of the API to invoke the flow.



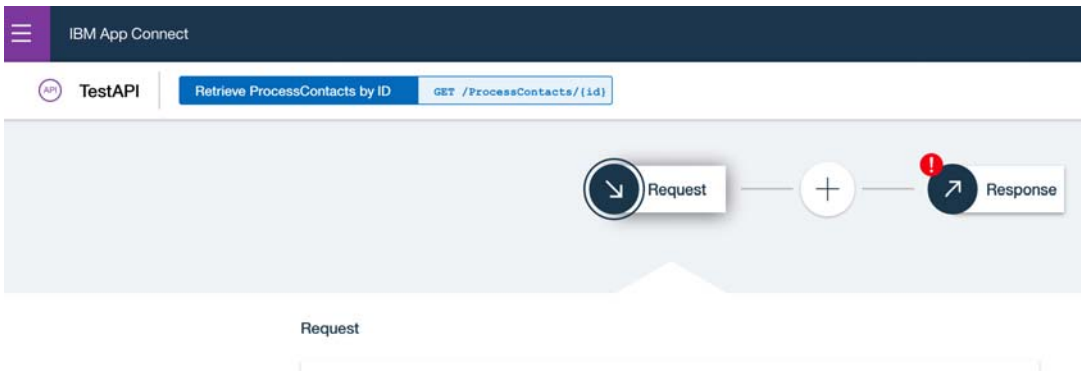
__ 5. Click the **Operations** tab just to the right of Properties. Then, from the **Operations to Add** list, select **Retrieve ProcessContacts by ID**.



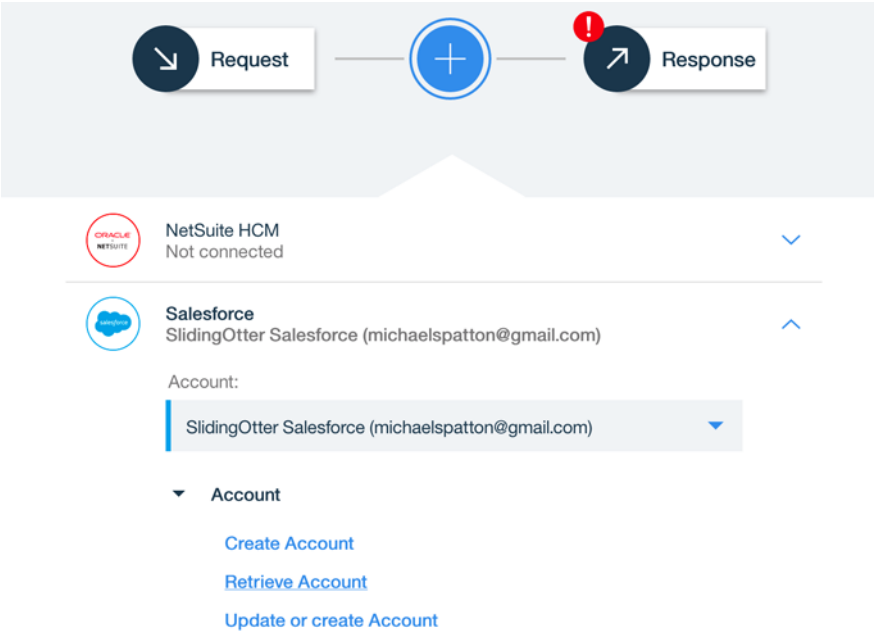
__ 6. You should then see the following output:



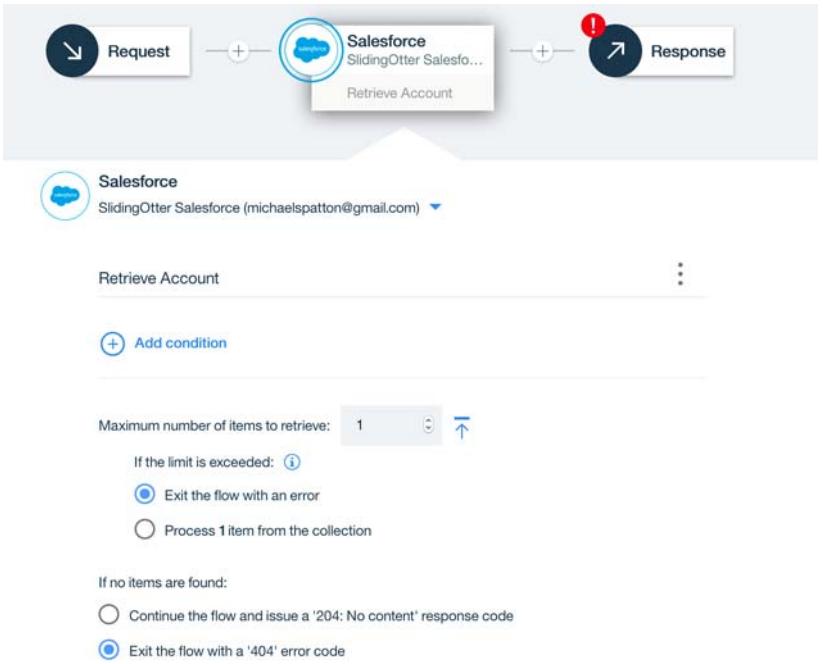
__ 7. Click **Implement flow**.



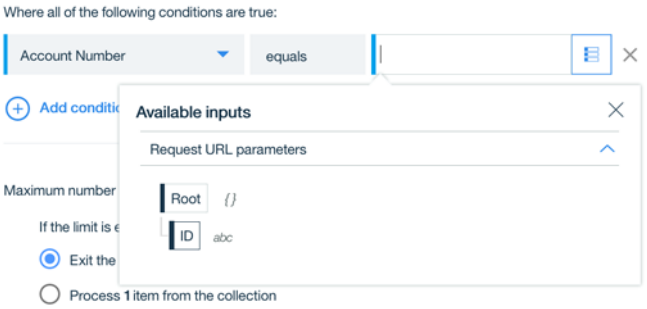
__ 8. Click **+** between the request and response nodes. Then, expand **Salesforce Application** to find **Accounts**. Select your Connection to Salesforce (in this case, it is “SlidingOtter Salesforce”, but yours might read “Account 1”), then select the Account twistie, and then “Retrieve Account”.



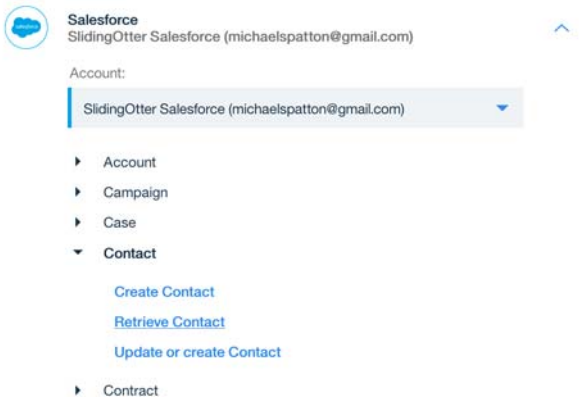
__ 9. The Salesforce node slides into place and a screen to add a condition to search for in Salesforce Accounts is displayed.



__ 10. Click **Add Condition** and select **Account Number** for the left side, and then click in the right side, and select ID from the inputs.

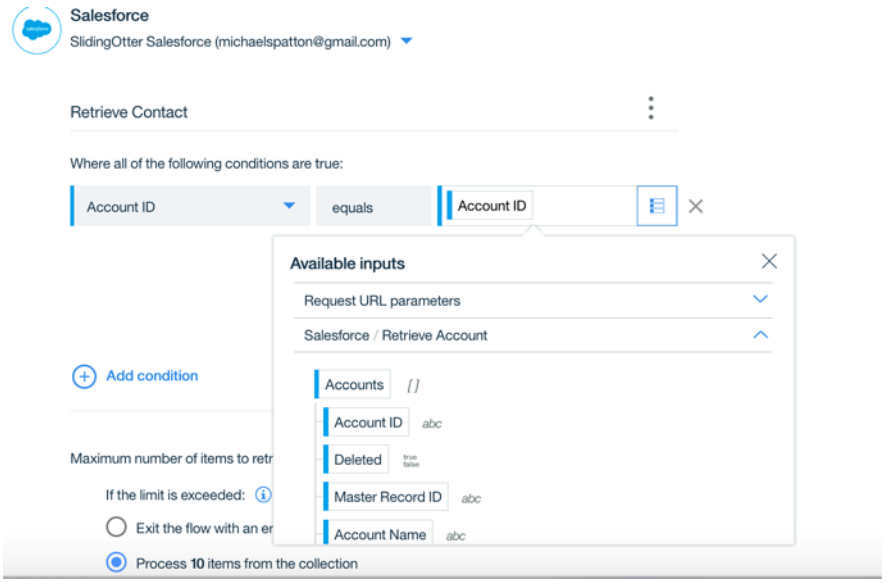


__ 11. Click the next **+** sign, to the right of the Salesforce “Retrieve Account” node and repeat for Salesforce again, but this time, retrieve Contacts.

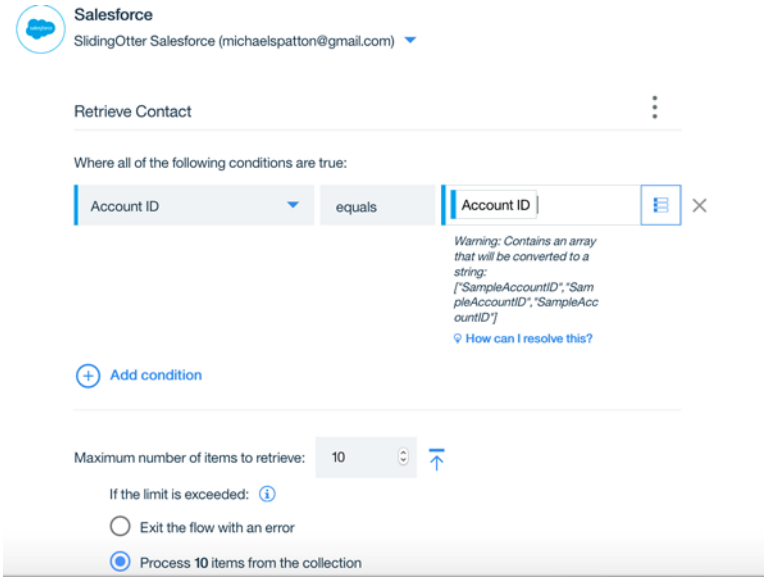


__ 12. You need to alter the configuration of the node for this one because you are expecting multiple records back.

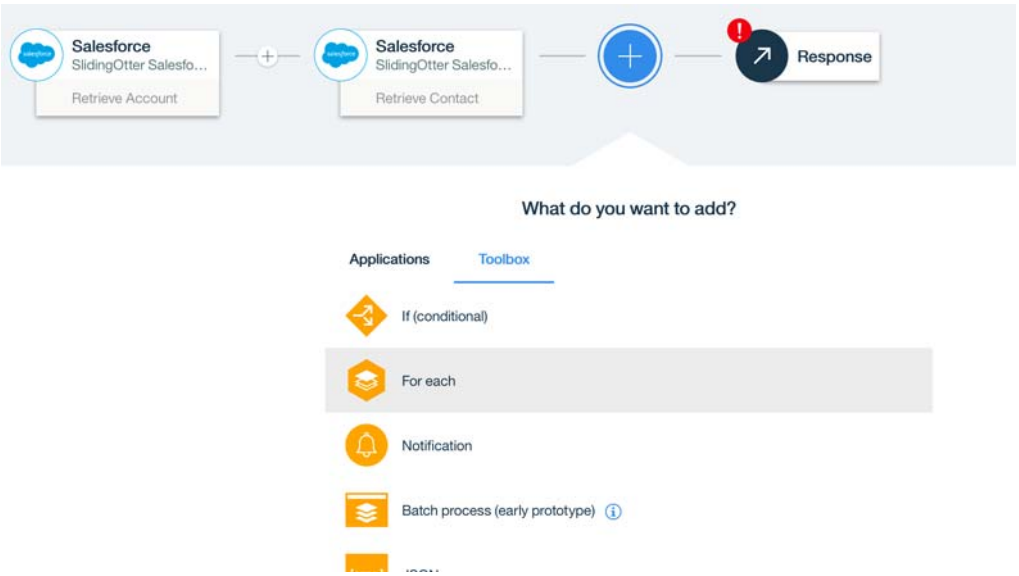
__ a. Set the left side of the condition to “Account ID” and the right side of the condition from the returned “Account ID”, from the previous node. It displays a list of available parameters to choose from, select the ones that you want.



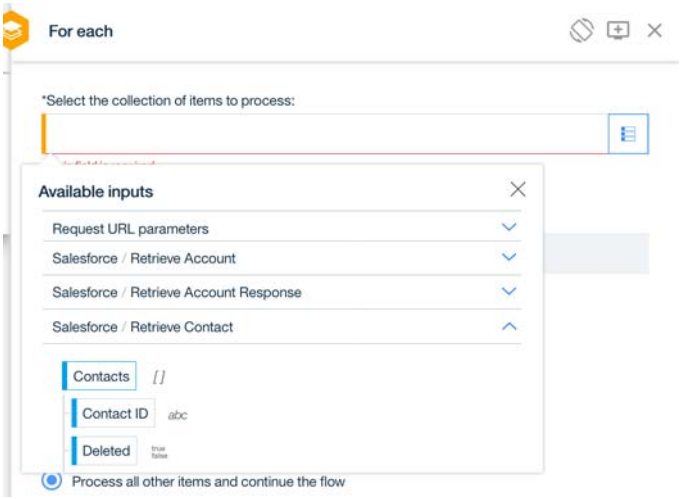
__ b. Since you are expecting more than one record, change the parameter to allow for more than one to be returned, limit it to 10 (ten). Select the radio button to indicate it is to process 10 records if more than ten are found.



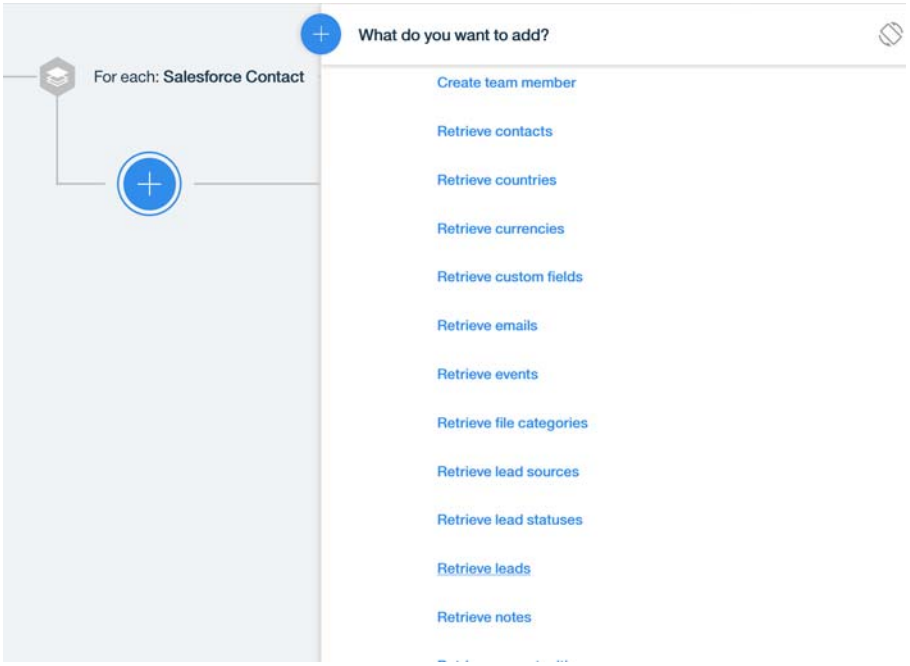
__ 13. You now have multiple records in your flow, and you process them one at a time. Insert a “For Each” operation by clicking the **+** sign to the right of the second Salesforce Node, then on the **Toolbox** tab, and select **For Each**.



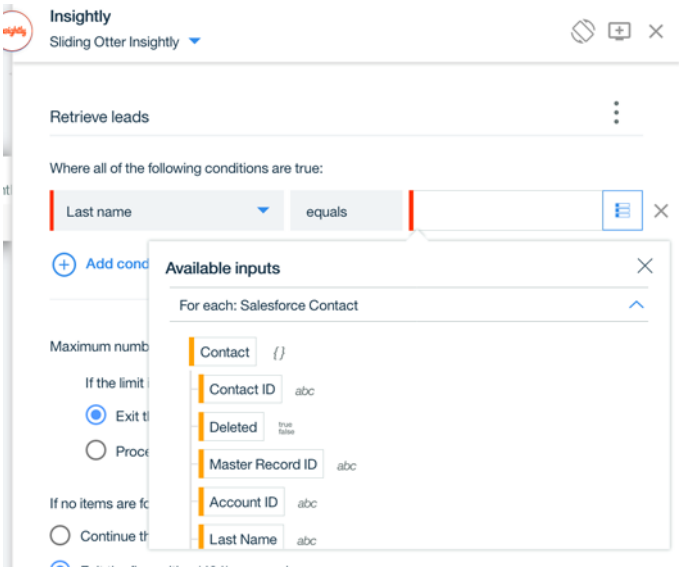
__ 14. Now, select **Select the collection of items to process**, choose the **Salesforce / Retrieve Contact** record array, and the structure to be looped on.



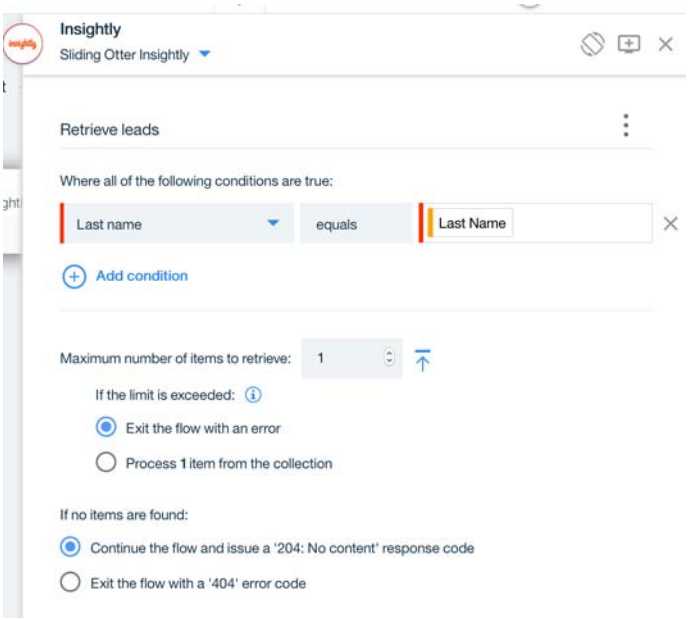
__ 15. You now loop through all contacts that are returned from Salesforce to get the email address from Insightly. Click the first **+** sign under the For Each loop, select the Insightly application, then click **Retrieve Leads**.



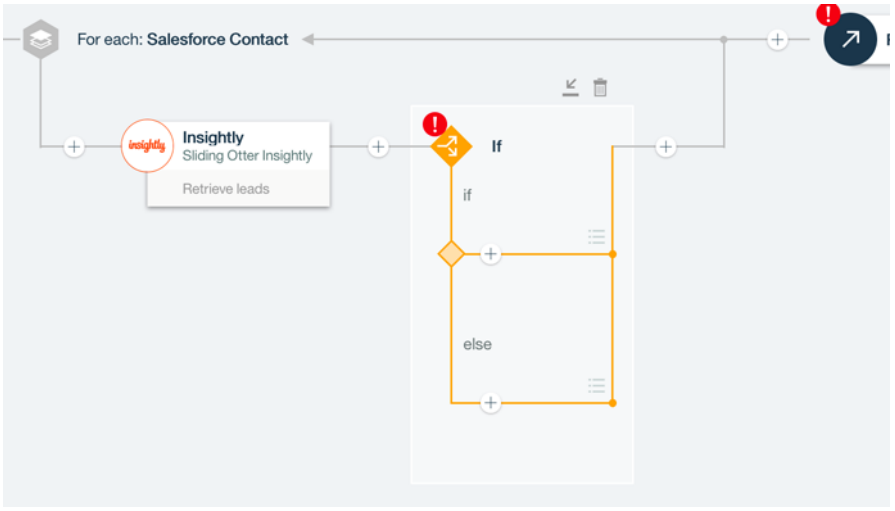
__ 16. You are again going to search for a record, and need a condition to match on. You have the lead's last name as one of the last names in the Contact array from Salesforce, and are stepping through those contacts one at a time. On the left side of the condition, select **Last Name**. On the right side, *and this is important*, select **Last name** from the **For each: Salesforce Contact** contact.



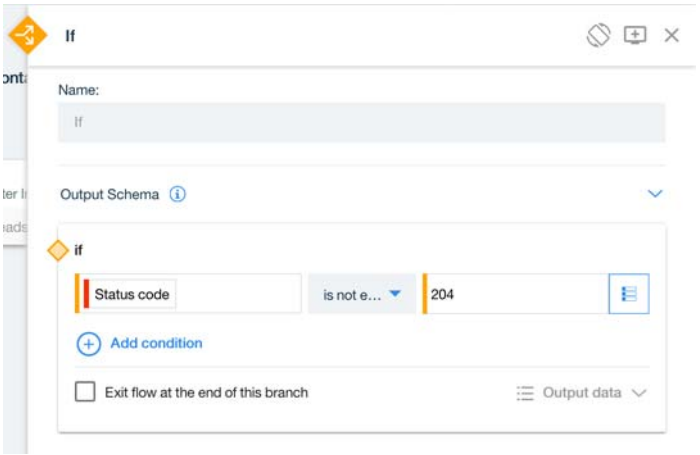
__ 17. Select the **If no items are found** option to continue the flow and issue a 204 code.



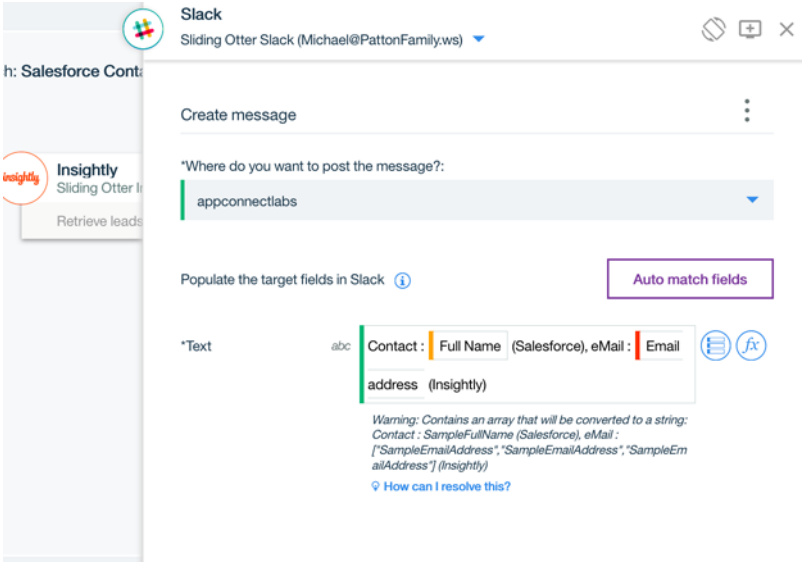
__ 18. To the right of the **Insightly** tab, click **+**, then on the toolkit, select the **IF** condition.



__ 19. On the “IF” condition check, select **Insightly Status Code**, change the comparison operator to **is not equal to**, then hardcode the value 204 in the right side. You are checking for a non existent record here.

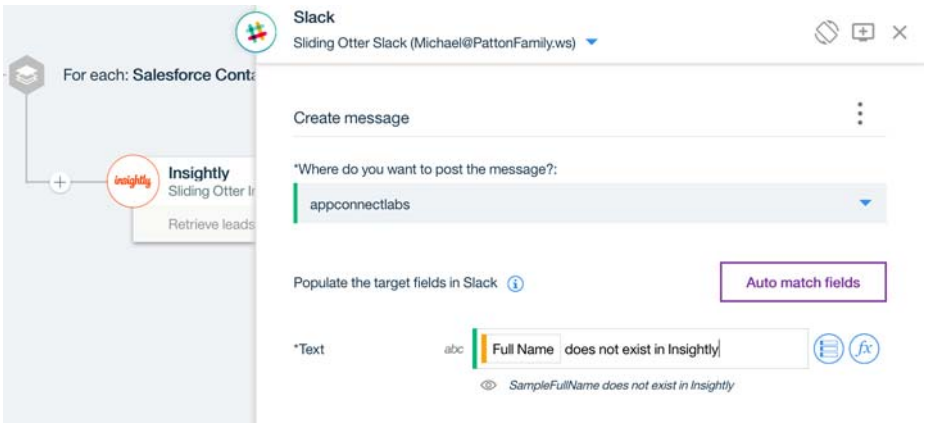


__ 20. Close the window, then select the first **+** under the IF statement. This node is executed if the condition is true, that is, the status code is not equal to 204. Select **Slack** as your output for the flow, where you send a message to show contact name and their email address. Select **Slack** from the applications, select your slack account, choose a slack channel to send the message to, and then select **create message**.

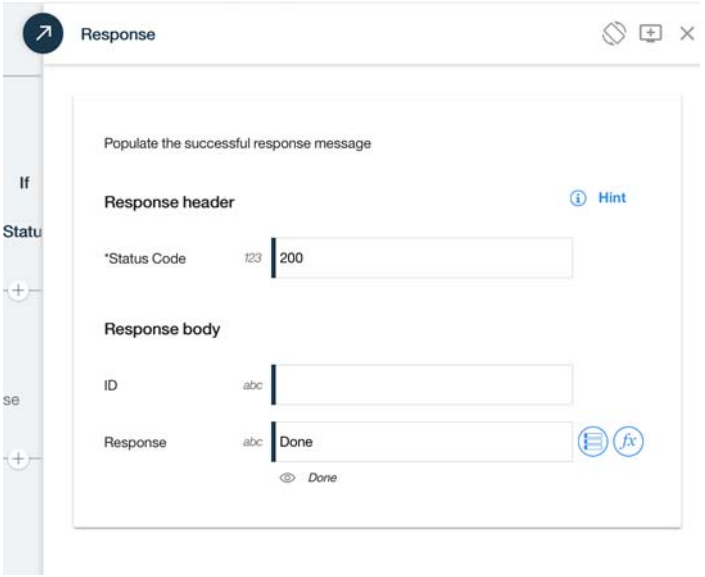


The message must be constructed again, with the full name from the “For Each: Salesforce Contacts” just like before, but since there is no array of email address, you can select the Insightly lead / email address. Fill any surrounding text to make the message read better.

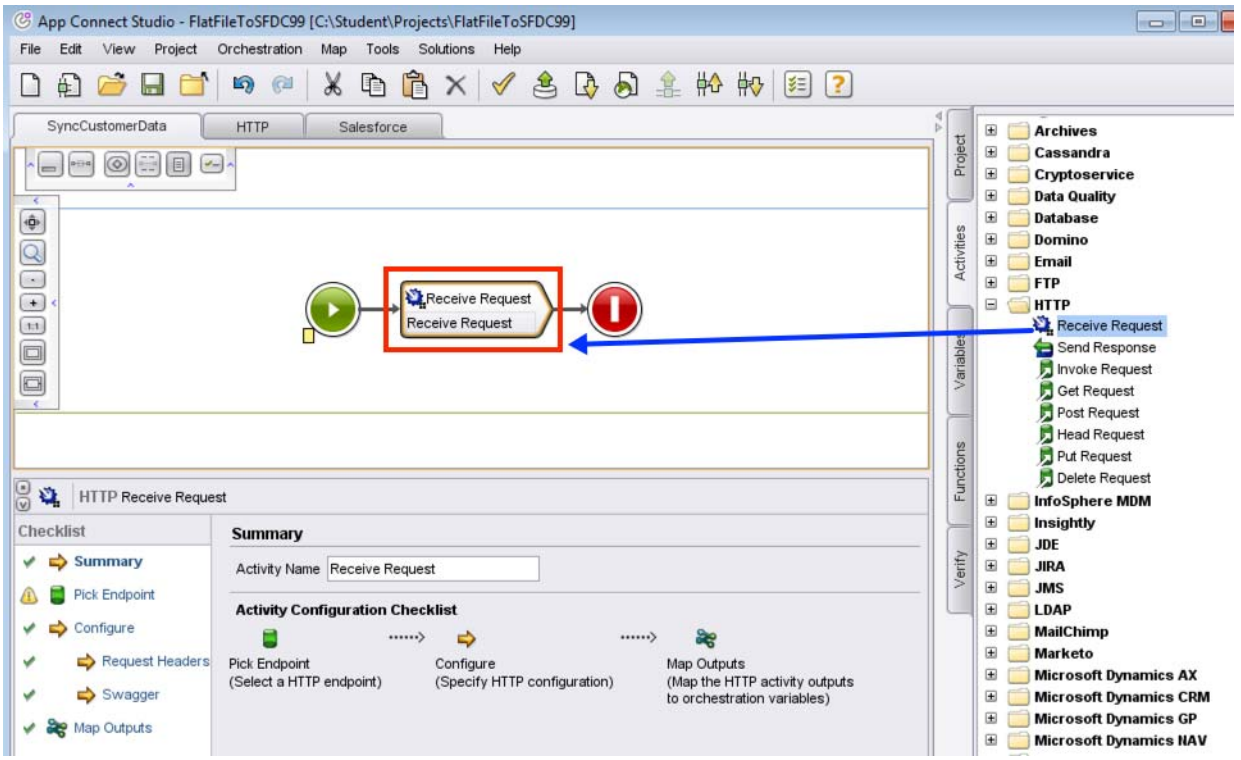
__ 21. Click the second **+** under the IF statement, this is the “else” portion of the if statement. Then, select **Slack**, and navigate just as before, but since you do not have a record, you change the message to indicate that there was no record.



__ 22. Finally, you can add a reply to your API Flow so that the calling application knows that you have done something. Click the **Response** node on the far right of the flow. It probably still has a red exclamation point with it. When opened, enter **Done** in the response field to tell your calling app that you have finished with the processing.



__ 23. You are now ready to start the test. Click **Done** in the upper-right corner.

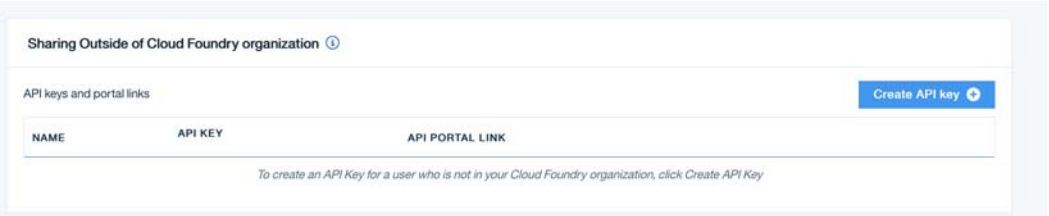


__ 24. You are returned to the same view that you are in on step 6.

8.3. Test the API

__ 1. Click the **Manage** tab. You are now in the API Testing functions for the flow. First, start the flow by clicking the 3 dots to the right of the word “Stopped” in the upper-right corner, and select **Start API**. You see the status “Running” with a green dot if the service started successfully.

__ 2. Scroll to the bottom of the page until you get to “Sharing Outside of Cloud Foundry Organization”, and click **Create API Key**. Provide a meaningful name.



__ 3. When an API Key is created, click **API Portal Link URL**. The testing interface must be configured for you. Click the green **Try It** text from the right.

__ 4. The tester prompts for your ID, enter the Account Number that you made when you were setting up the Salesforce Account. In this case, use the account number 123456789.

__ 5. You should see several messages in Slack from App Connect, giving you the name and email address for all but one of your contacts. For that missing contact, it gives you the message that you entered a nonexistent record in Insightly.

```
Ellen Ripley does not exist in Insightly
Contact : Joe Gibbs (Salesforce), eMail :
Joe.Gibbs@SlidingOtterEnterprises.com (Insightly)
Contact : John yousley (Salesforce), eMail :
John.yousley@SlidingOtterEnterprises.com (Insightly)
Contact : Leonard Cohen (Salesforce), eMail :
Leonard.Cohen@SlidingOtterEnterprises.com (Insightly)
Contact : Dave Gahan (Salesforce), eMail :
Dave.Gahan@SlidingOtterEnterprises.com (Insightly)
Contact : Rumple Stiltskin (Salesforce), eMail :
Rumple.Stiltskin@SlidingOtterEnterprises.com (Insightly)
```

End of exercise

Exercise review and wrap-up

In this exercise, you combined data from multiple sources and process each record returned

Exercise 9. Salesforce account API

Estimated time

00:60

Overview

In this exercise, the student updates the Salesforce accounts and contracts to be created. To simplify the process of creating these accounts, a utility program is supplied that creates the objects. This utility program needs to be configured with the Salesforce account details and then automatically create the objects that are required.

Objectives

After completing this exercise, you should be able to:

- Enhance their Mobile and web apps to include customer account information.

Introduction

My Great Retailer wants to enhance their Mobile and web apps to include customer account information. To enable this, they want to expose existing information in Salesforce. However, the application developers consider the standard Salesforce API complex. The IT department is unable to immediately help due to other priorities, but the application teams would like to create a simplified API, that can be consumed easily by the web and mobile applications. They use App Connect to create an Account API that connects to Salesforce.

Exercise instructions

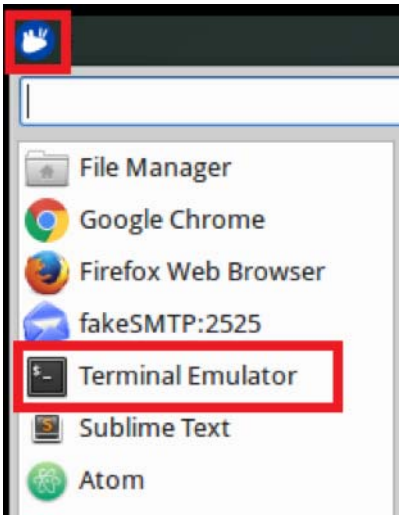
To begin this activity, ensure you have the following accounts:

- An IBM App Connect account
- A Salesforce developer account preloaded sample accounts in Salesforce

9.1. Prerequisites for loading data into Salesforce

The scenario requires Salesforce Accounts and Contracts to be created. To simplify the process of creating these accounts, a utility program is supplied that creates the objects. This utility program needs to be configured with the Salesforce account details and then automatically creates the objects that are required.

__ 1. Open a terminal, click the **Xubuntu** icon, and select **Terminal Emulator**:

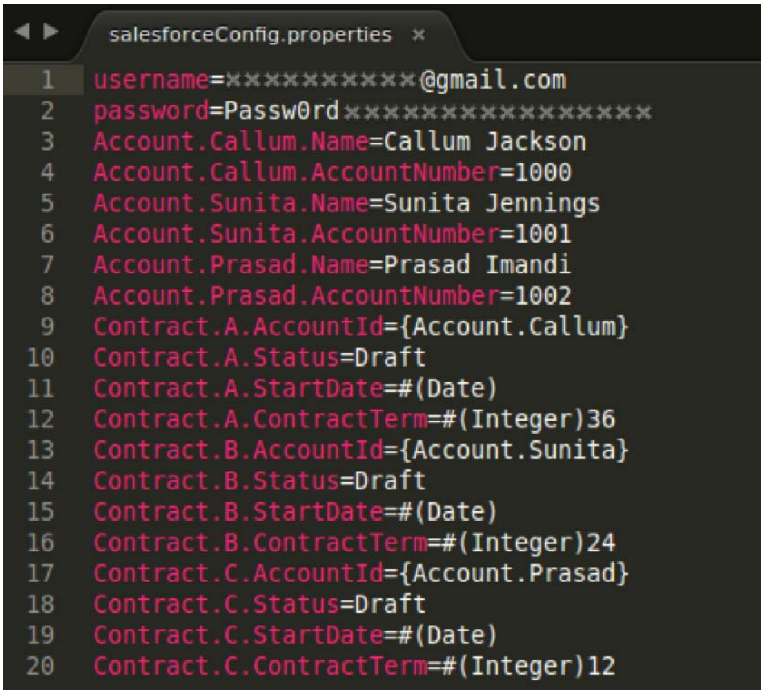


__ 2. Navigate to /home/student/ais-quickstart/integration-engine/utils:

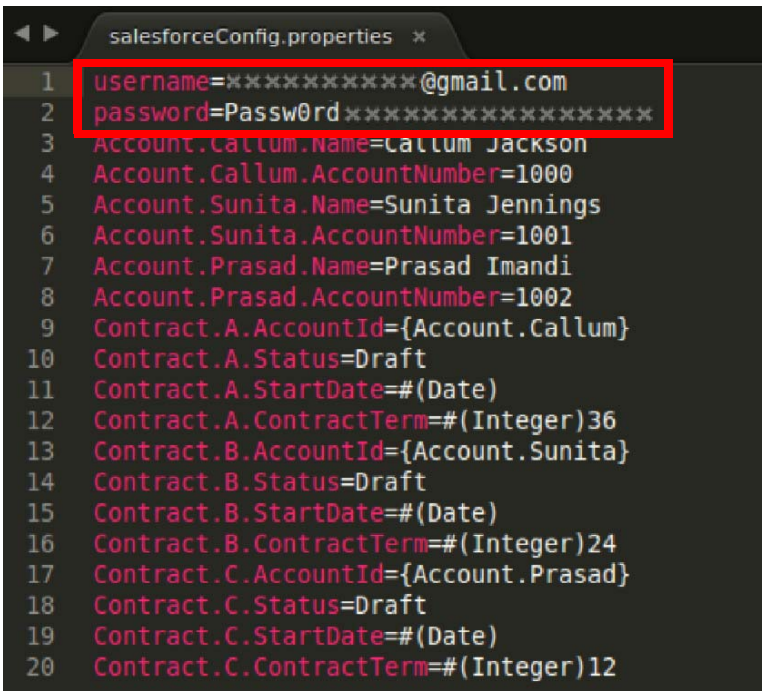
```
cd /home/student/ais-quickstart/integration-engine/utils
```

__ 3. Open the utility program configuration file for editing:

```
sub1 salesforceConfig.properties
```

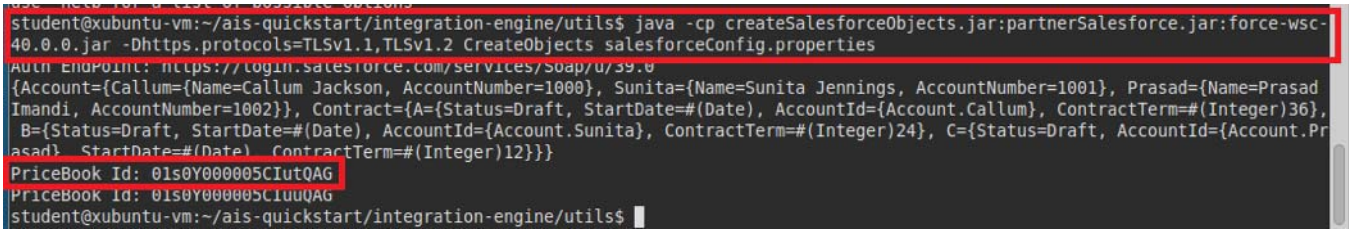


__ 4. Customize the configuration file to include the correct <SALESFORCE_ID> and <SALESFORCE_PASSWORD><SALESFORCE_SECURITY_TOKEN> as highlighted in the following figure:



__ 5. Return to the terminal window, make sure that the current directory is `~/ais-quickstart/integration- engine/utils`, and run the following command to run the utility:

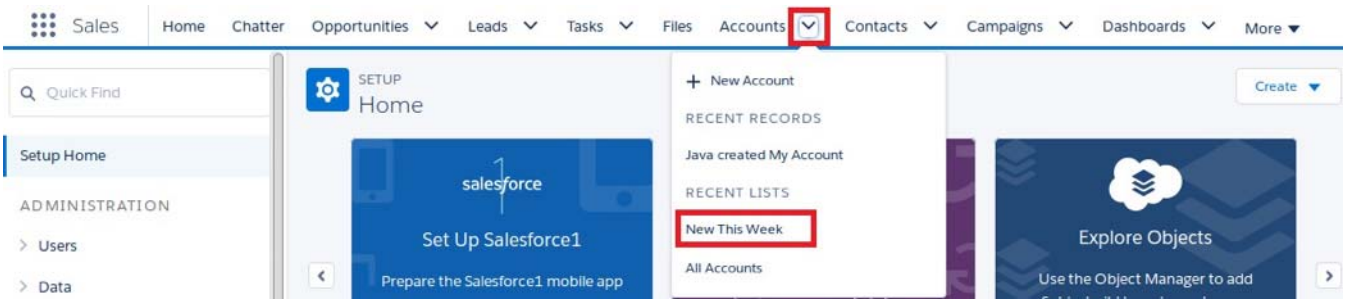
```
java -cp createSalesforceObjects.jar:partnerSalesforce.jar:force-wsc-40.0.0.jar
-Dhttps.protocols=TLSv1.1,TLSv1.2 CreateObjects
salesforceConfig.properties
```



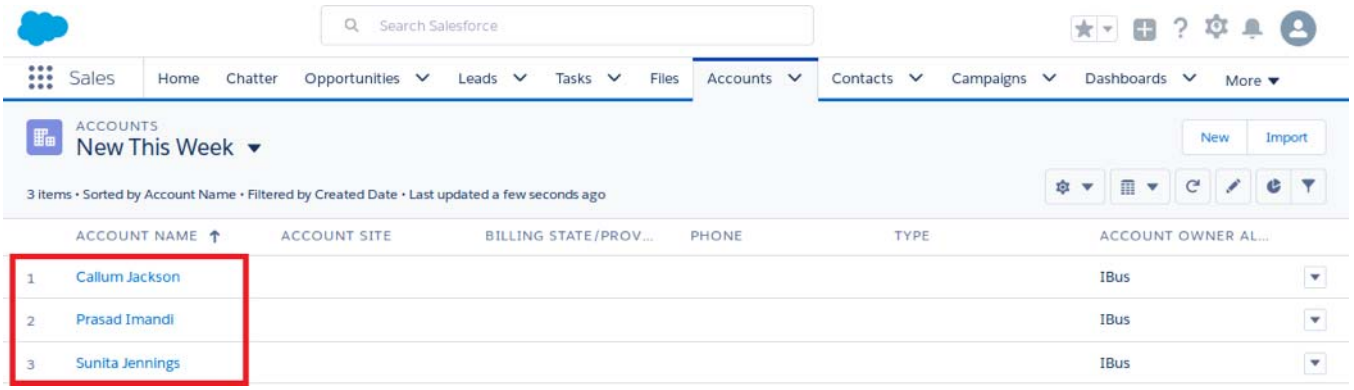
__ 6. In addition to creating the required Salesforce objects, it prints out the default PriceBook identifier (highlighted above). Copy this identifier as it is required in future steps.

__ 7. To verify that the utility program created the objects, navigate back to Firefox and log in to Salesforce: https://login.salesforce.com with the login details.

__ 8. To view the accounts, click the down arrow for the account.



__ 9. View that three accounts have been created:



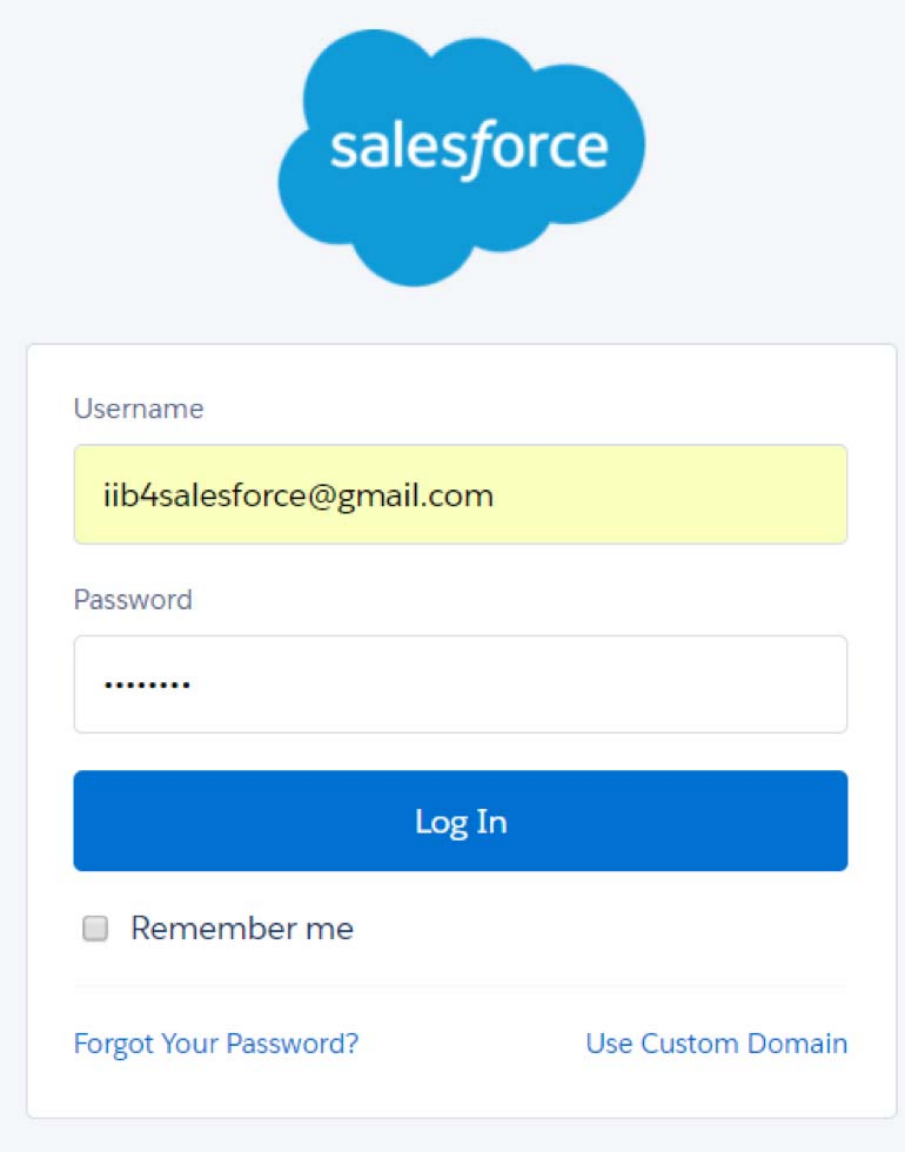
Connect to Salesforce

__ 1. Log in to https://designer.appconnect.ibmcloud.com with your IBM ID.

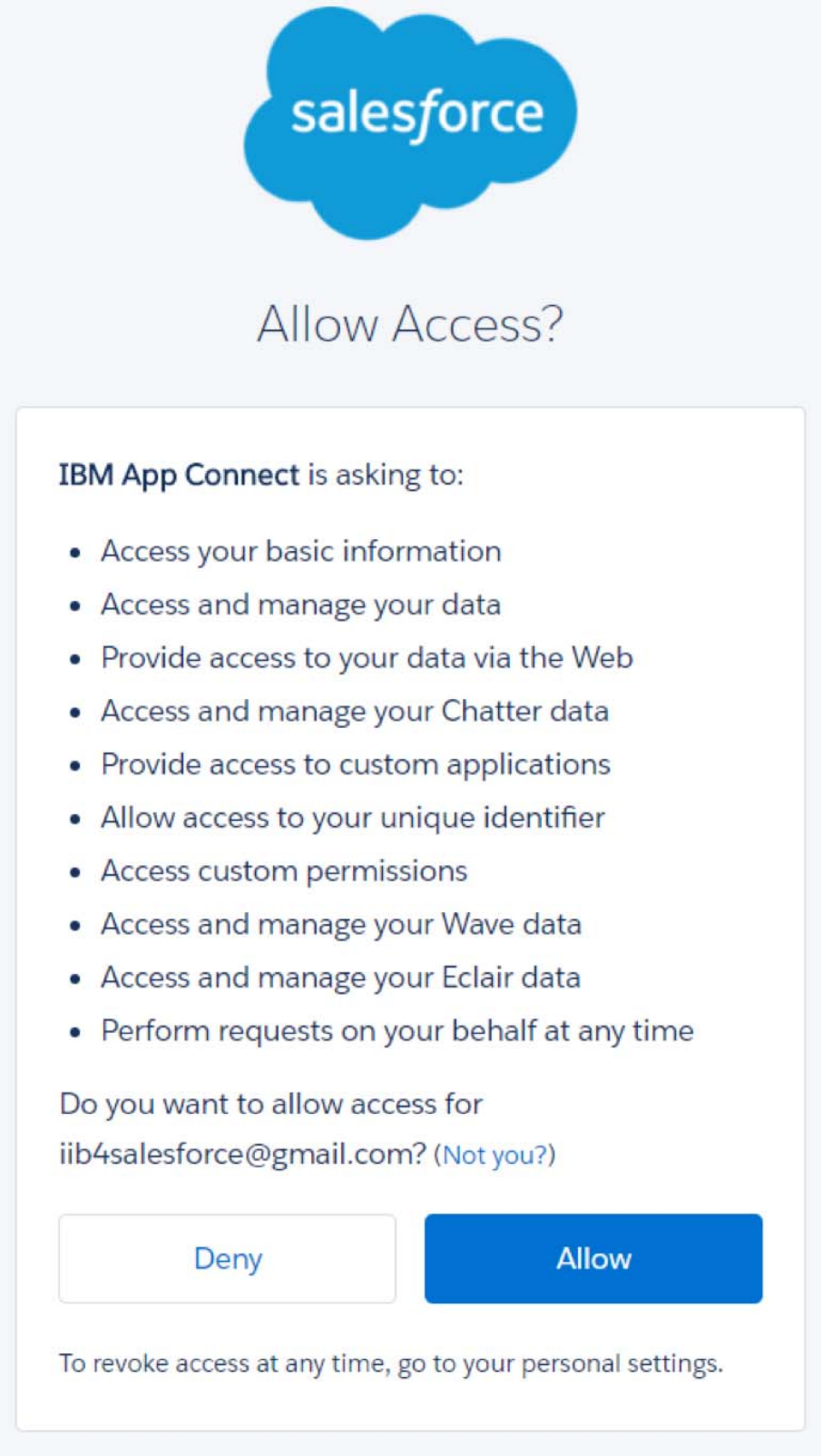
__ 2. Open the **Applications** tab and search for Salesforce.

__ 3. A warning regarding your Salesforce account being API enabled might appear, click **Continue**.

__ 4. You are redirected to the Salesforce login page, enter your user name and password, and click **Log In**.

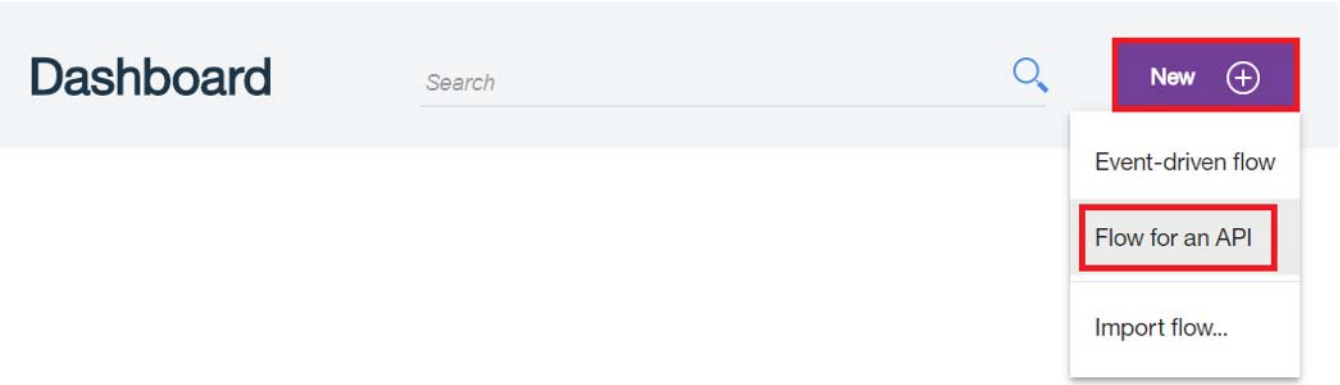


__ 5. In Salesforce, verify your credentials and confirm if you want to provide access to App Connect. Click **Allow**.

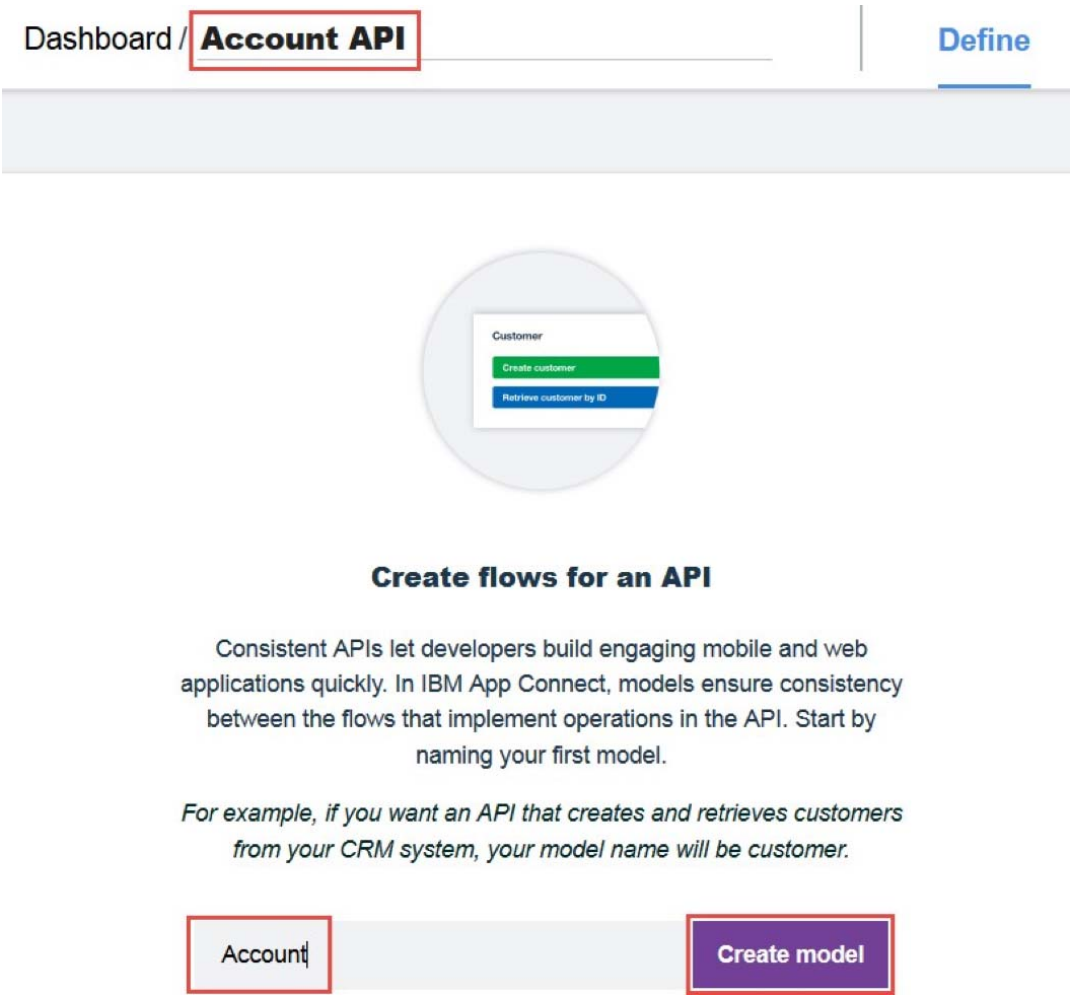


Define your API

__ 1. Return to the Dashboard and start creating your API and its model.

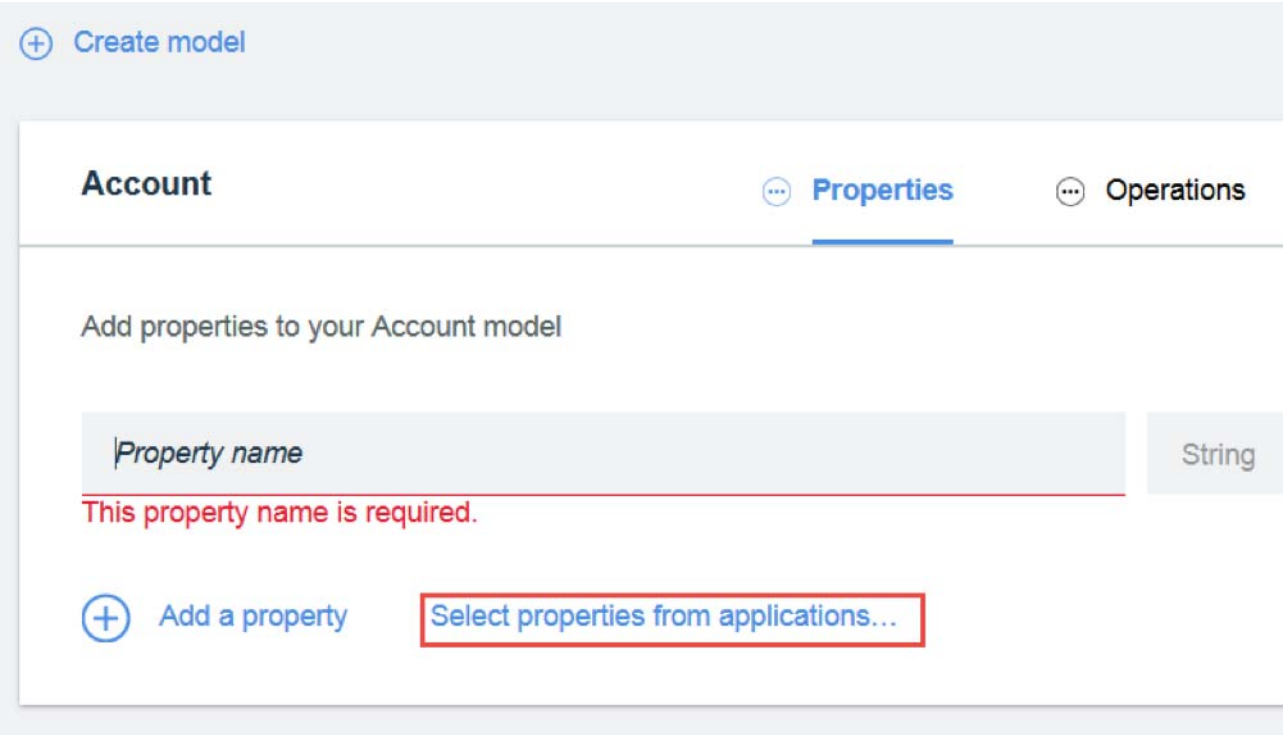


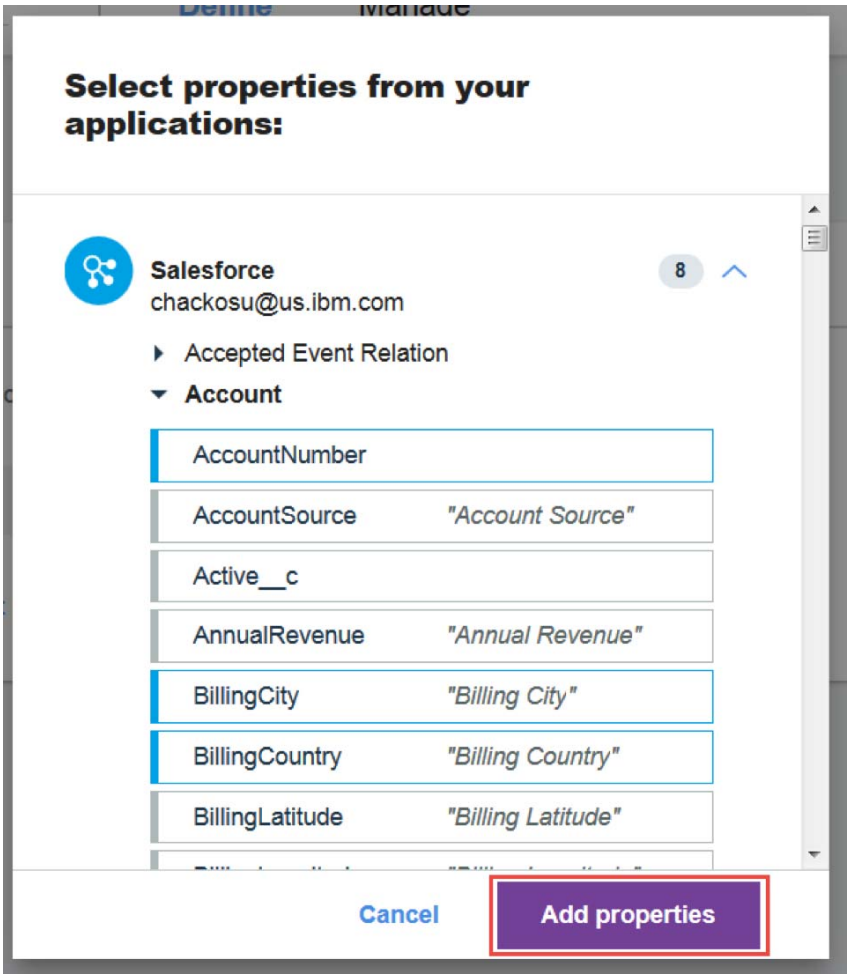
__ 2. Name your new flow **Account API** and your model **Account**. Click **Create model**.



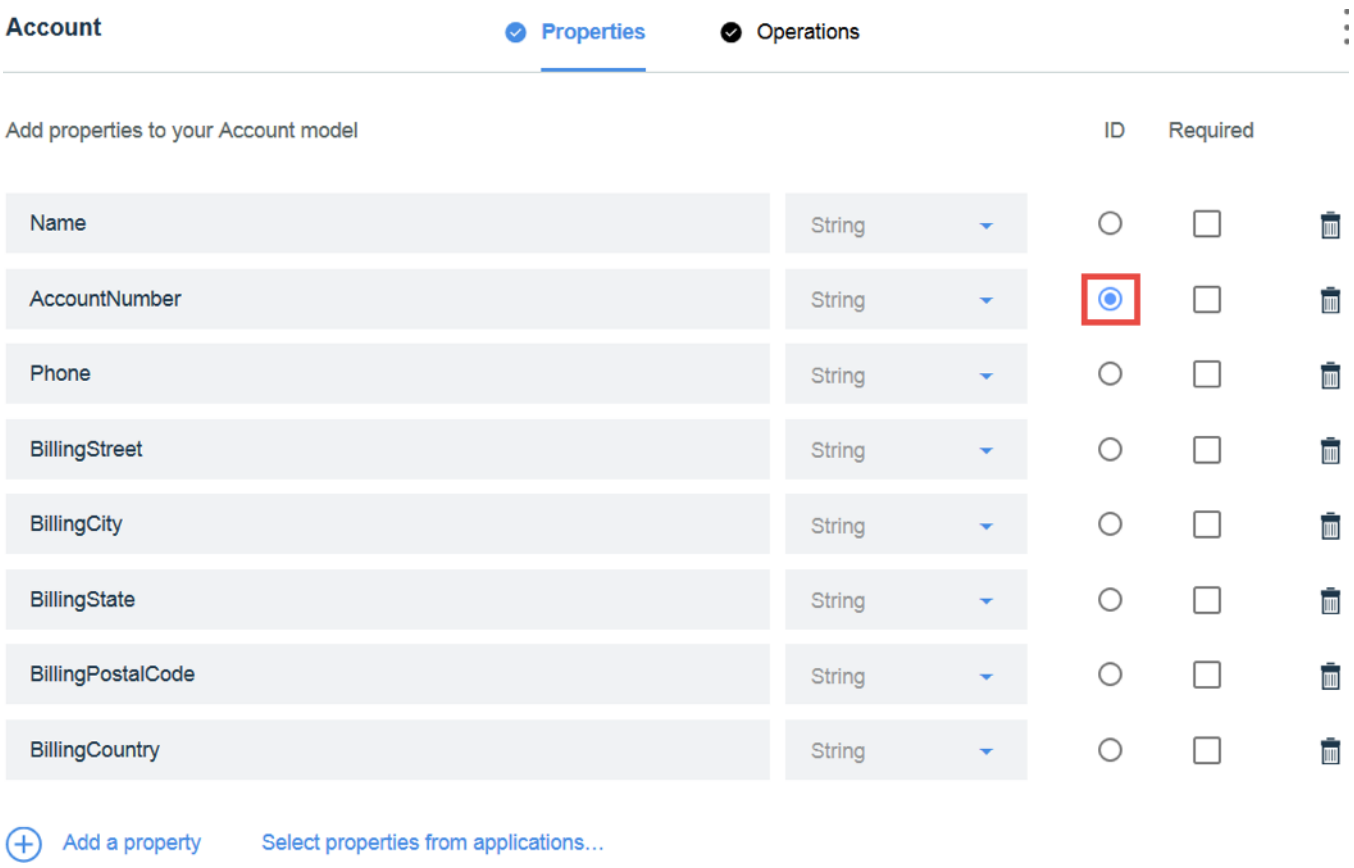
__ 3. Create some properties on the Account model. Select properties from the Salesforce application's **Account** object. You can select multiple properties at once. Properties are displayed in the order that you select them.

- AccountNumber
- BillingCity
- BillingCountry
- BillingPostalCode
- BillingState
- BillingStreet
- Name
- Phone





__ 4. Mark AccountNumber as the ID by selecting the option. You are going to query accounts by using this field.

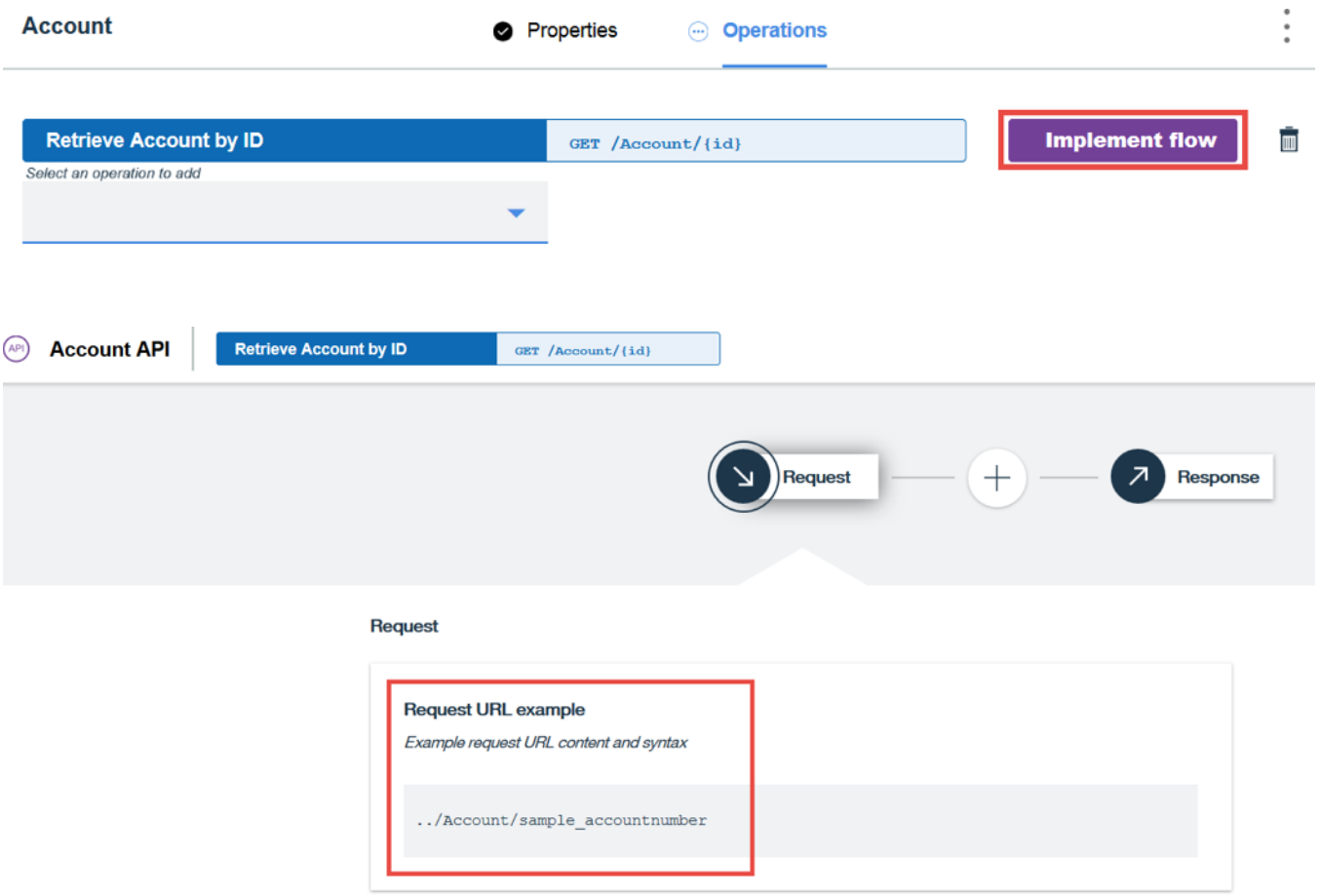


Implement the Retrieve Account operation

__ 1. Click the **Operations** tab, and select the **Retrieve Account by ID** operation to add it.

__ 2. You should see the new API operation defined as GET / Account/{id}.

__ 3. Click the **Implement flow** button to begin working on the operation. This opens the flow editor and shows you a sample request URL that an app might make.

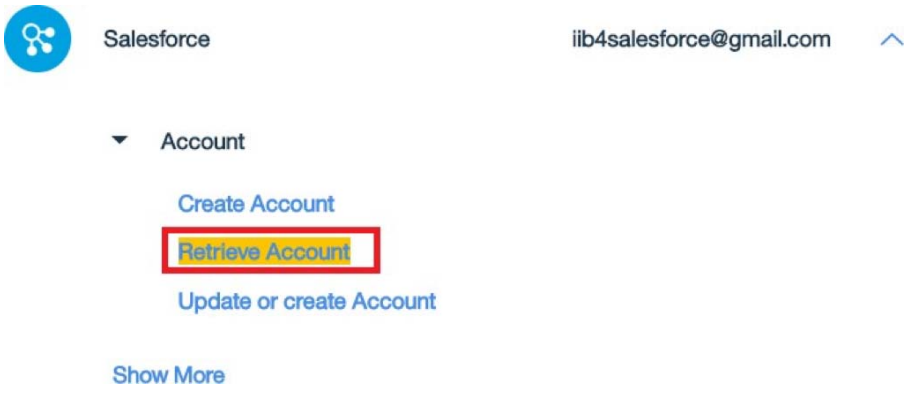


Retrieve the Account information from Salesforce

The mobile app is going to supply you with an Account number, and you use that to query Salesforce for the Account object.

__ 1. Click the (+) to add an action to the flow.

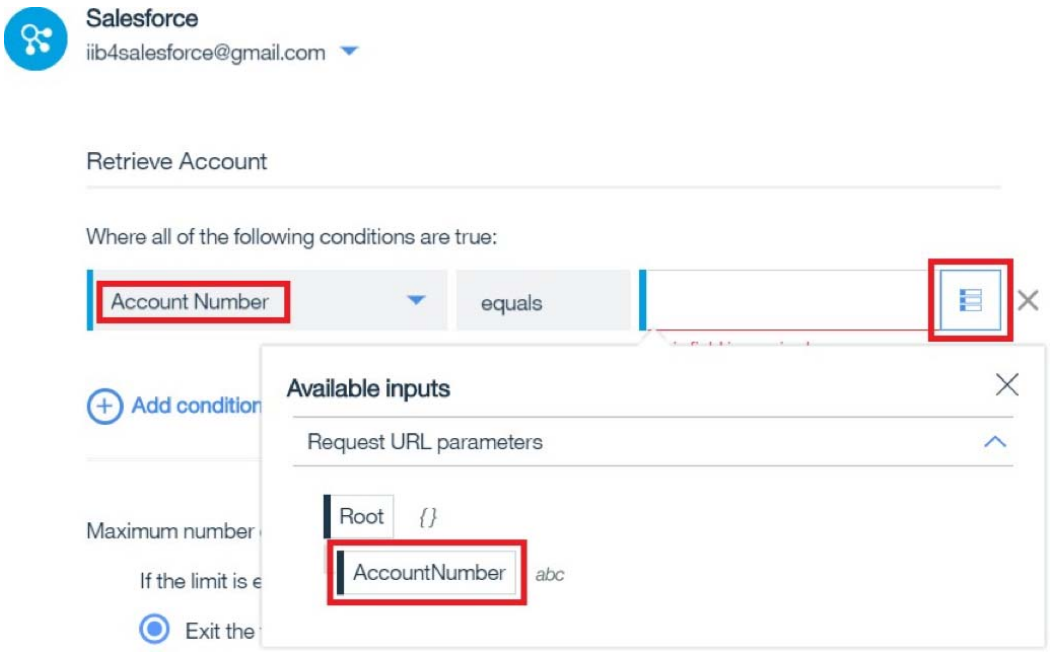
__ 2. Type **Retrieve Account** in the search field, and select the action from Salesforce. IBM App Connect connects to Salesforce to obtain configuration options for this account.



__ 3. Choose **Account Number** as the left side of the lookup condition.

__ 4. Click the icon on the right to display a list of references that are available in the context at this point in the flow.

__ 5. Choose **Account Number** from the list of request parameters.



You now have a fully configured retrieve operation that looks for exactly one Account in Salesforce that matches the AccountNumber included in the API Request URL. If zero or more than one are found, then the flow reports an error when it runs.

Return the Order information

The last step is to associate the retrieved data with the API response.

__ 1. Click **Response action** in the flow, and complete the reply mapping for each of the target fields in the response. For the **Name** field, type `Account Name` and select the field under **Salesforce / Retrieve Account**.

__ 2. Repeat for each of the response fields:

__ a. AccountNumber: **Parameters > AccountNumber**

__ b. Phone: **Retrieve Account > Account Phone**

__ c. BillingStreet: **Salesforce / Retrieve Account > Billing Street**

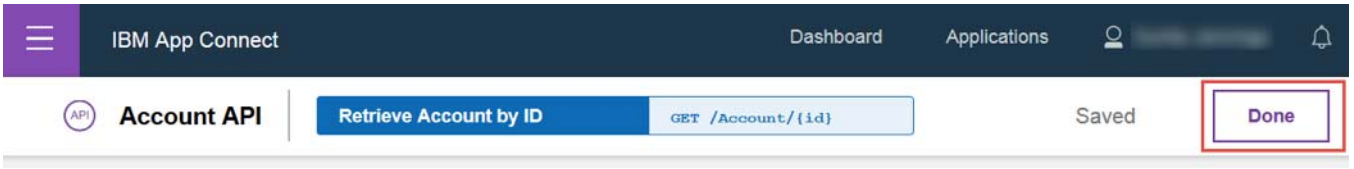
__ d. BillingCity: **Salesforce / Retrieve Account > Billing City**

__ e. BillingState: **Salesforce / Retrieve Account > Billing State/Province**

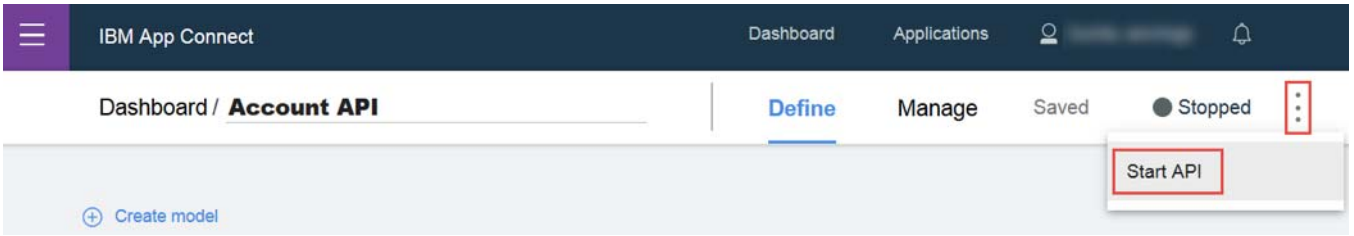
__ f. BillingPostalCode: **Salesforce / Retrieve Account > Billing Zip/Postal Code**

__ g. BillingCountry: **Salesforce / Retrieve Account > Billing Country**

__ 3. Click **Done** to exit the flow editor.

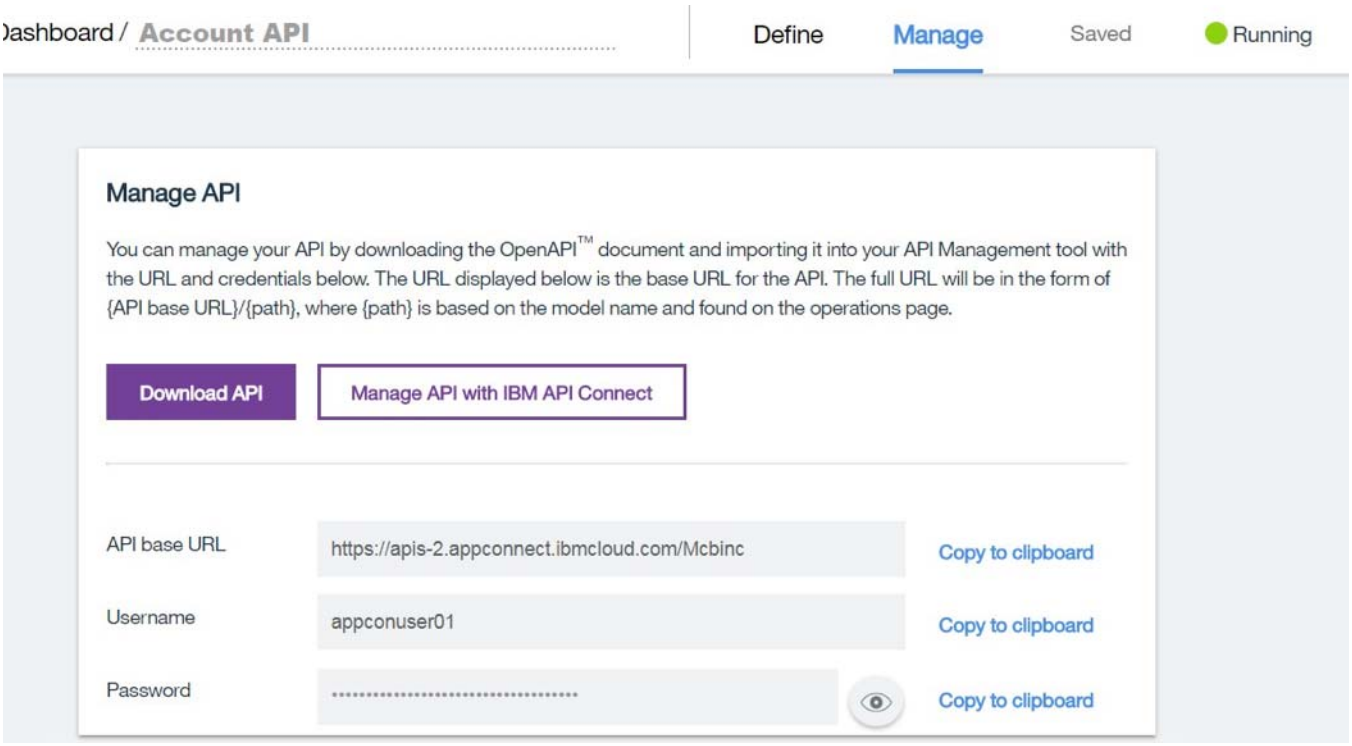


__ 4. Click the menu in the upper-right and select **Start API**.



Test your new API operation

Now you have the first operation on your Account API running. Click the **Manage** tab and copy the base URL, user name, and password for your API to a notepad.



Use one or more of the following options to call your API. In each case, the API URL you use should be: `<baseURL>/Account/Account/1000`
You can also use the other sample account numbers that are preloaded as part of the prerequisite steps: `1001` and `1002`

Option 1: cURL (Command-line tool)

__ 1. At the command-line, type:

```
curl -X GET -- basic -u <username>:<password> <baseURL>/Account/1000
```

You should see the following response:

```
{"Name":"Callum Jackson","AccountNumber":"1000","Phone":"null",
"BillingStreet":"null","BillingCity":"null","BillingState":"null",
"BillingPostalCode":"null","BillingCountry":"null"}
```

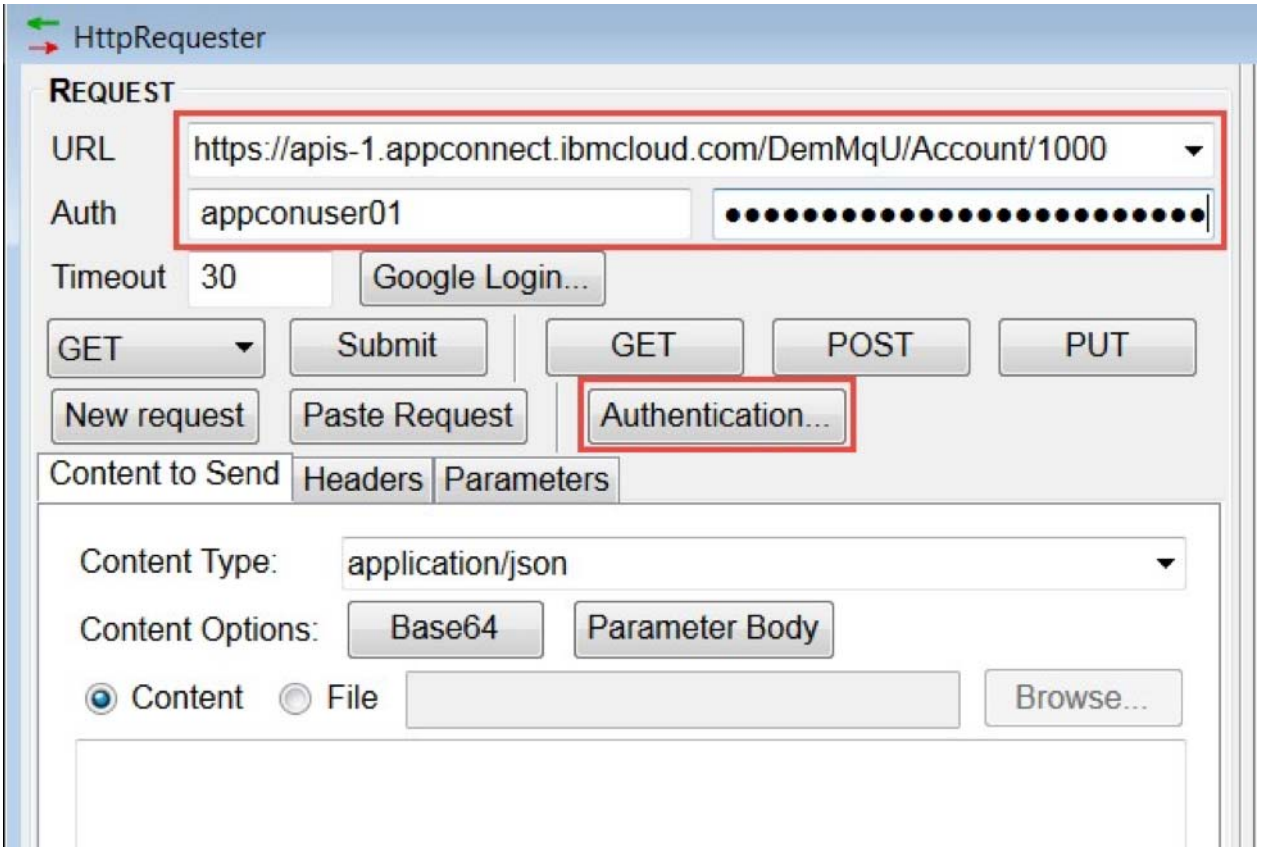
Option 2: HttpRequester (Firefox app)

__ 1. Launch **HttpRequester**.

__ 2. In the URL field, type: `<baseURL>/Account/1000`

__ 3. Click **Authentication**.

__ 4. In the **Auth** fields that appear, type the user name and password that you copied earlier.



__ 5. Click **Submit**.

__ 6. In the response field, you should see the following text:

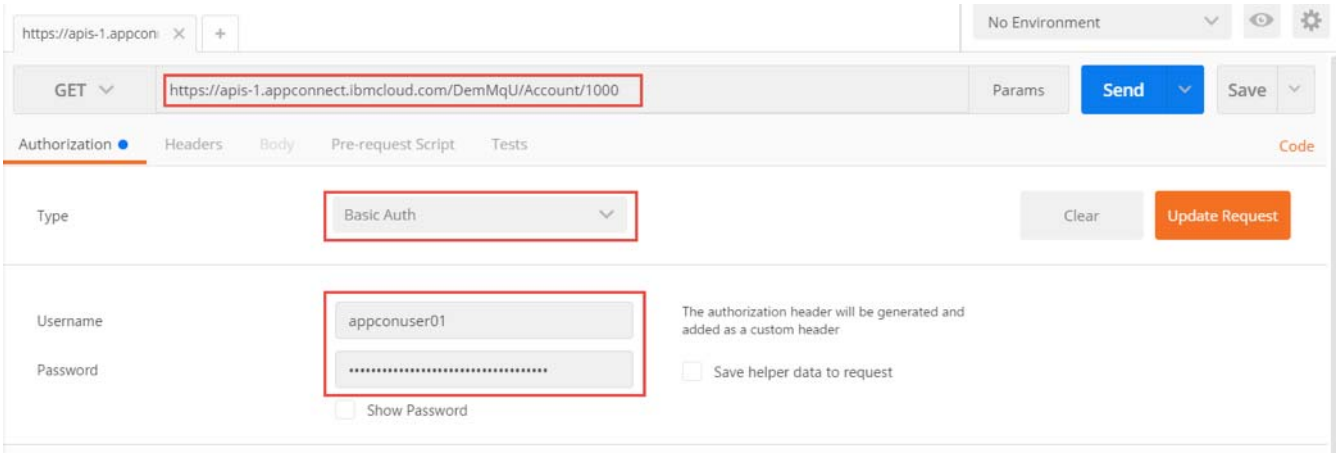
```
{"Name":"Callum Jackson","AccountNumber":"1000","Phone":"null",
"BillingStreet":"null","Billi"BillingPostalCode":"null",
"BillingCountry":"null"}
```

Option 3: Postman (Chrome app)

__ 1. Launch the **Postman** app.

__ 2. Next to the GET, enter the API URL: **baseURL/Account/1000**

__ 3. On the **Authorization** tab, select **Basic Auth** and provide the user name and password you copied earlier.



__ 4. Click **Send**. You should see the following response:

```
{

"AccountNumber": "1000",
"Phone": "null",
"BillingStreet": "null",
"BillingCity": "null",
"BillingState": "null",
"BillingPostalCode": "null",
}
```

End of exercise

Exercise review and wrap-up

In this exercise, you updated the Salesforce accounts and contracts to be created.

Appendix A. Creating a Salesforce.com developer account

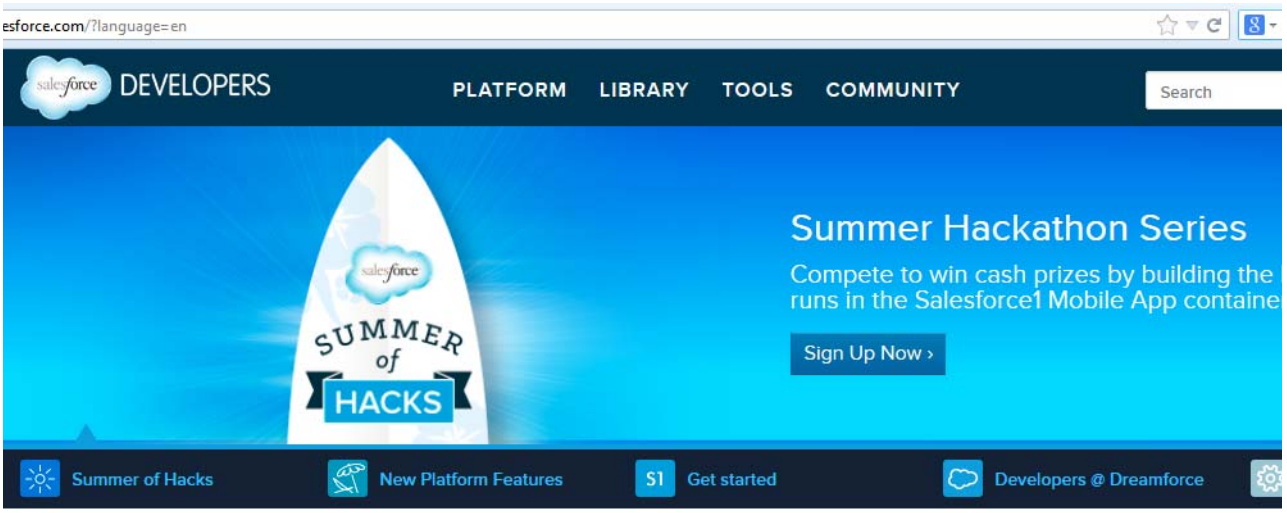
To complete the labs, you need a valid Salesforce.com user name and password to access a Salesforce.com account. As connectivity is provided through the Salesforce.com API, you also need to know the Salesforce.com security token. Complete this appendix before starting the labs.

The following section takes you through setting up an account, activating the access to your Salesforce account, and obtaining the security token. Do this series of steps from within the proof of technology virtual machine and not from the native PC.

If you already have a developer account and know this information, then you still need to run through the section to activate the notebook to access your Salesforce account.

Setting up an account

__ 1. Using a web browser go to the URL: http://developer.force.com/.
Click **Sign Up Now**.



__ 2. Complete the form details and submit. You need a valid email address, and you must be able to access the email.

__ 3. Sign in to your email account.

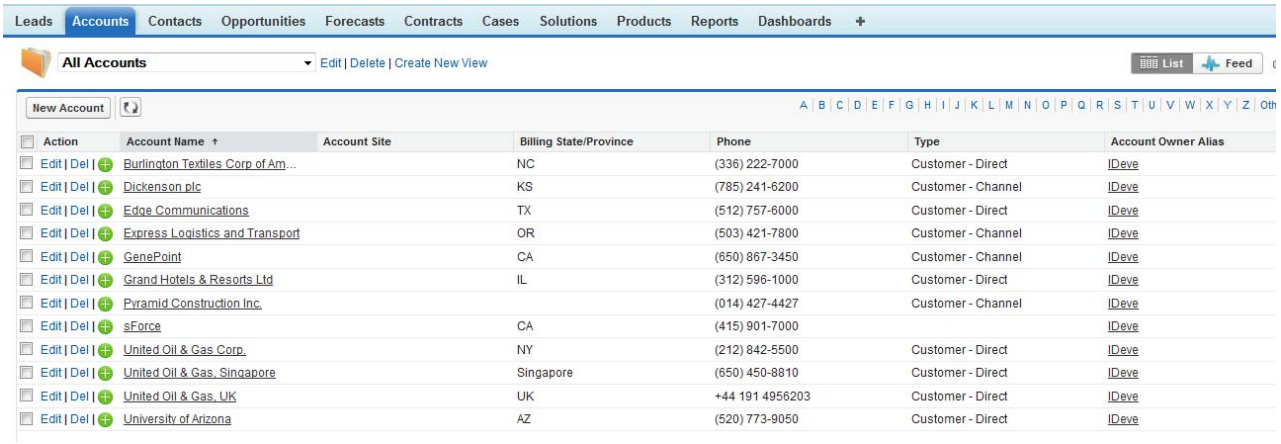
__ 4. You receive an email from Salesforce.com. Open the one with the subject **Salesforce.com login confirmation** and follow the link to confirm the account setup.

__ 5. Enter a password of your choice.
You now have access to the developer account. Verify the sample data with the following steps.

__ a. Click the **Accounts** tab.

__ b. From the **View** menu, select **All Accounts**.

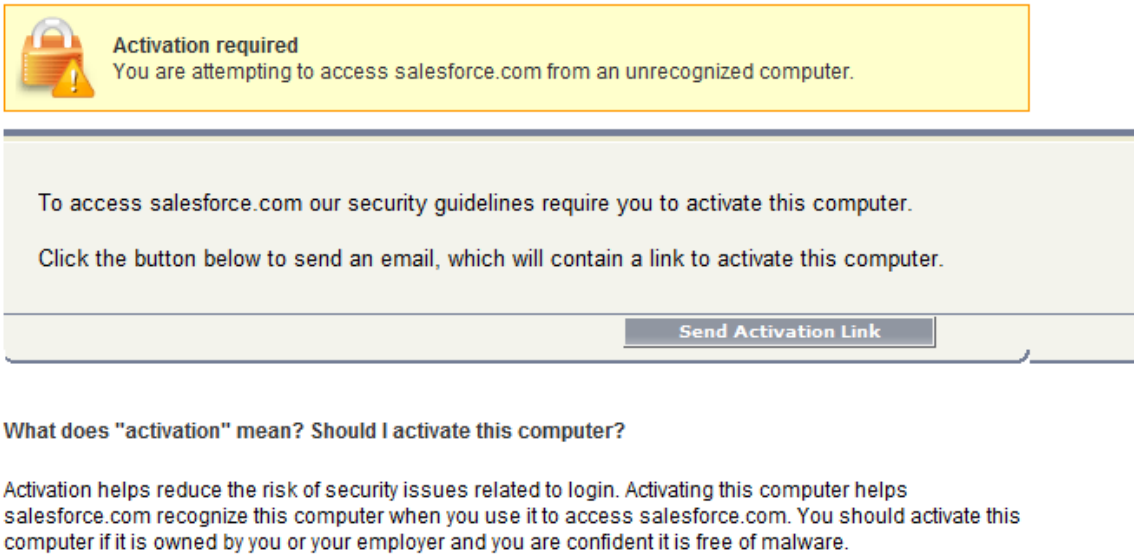
__ c. Click the **Go** icon. You are now shown the list of sample accounts. You can look at the detail for these accounts by clicking an account name.



You now need to retrieve your security token.

Activating Salesforce.com for use on the notebook

__ 1. Log in to Salesforce.com. You see the message that requires activation. If you do not see this message, you do not need to activate the notebook.



__ 2. Click **Send Activation Link** to send an email to the email address that is used for your Salesforce.com account.

__ 3. Access your email and follow the link that is provided in that email. You can now access your Salesforce.com account from this machine.

__ 4. Set and Retrieve the Security Token for Existing Salesforce.com Accounts

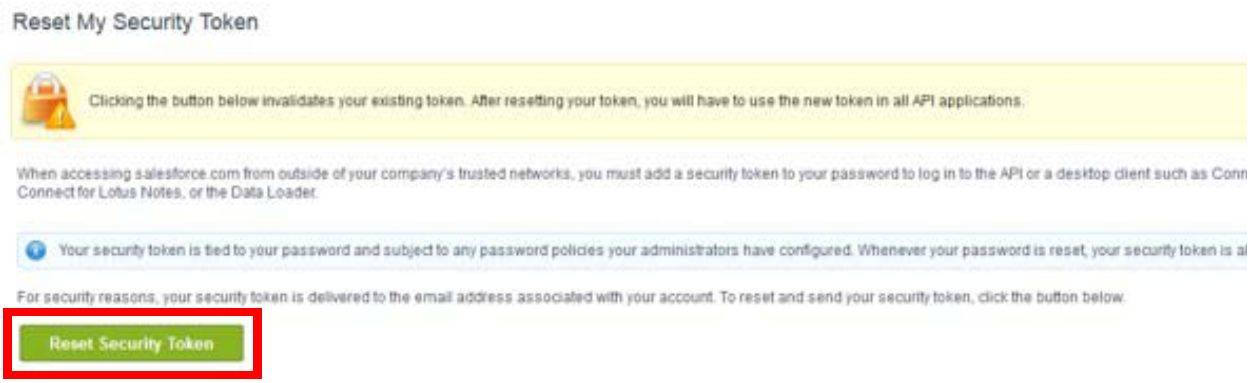
__ 5. If you have an existing Salesforce.com account, you must activate the laptop to access your Salesforce.com account before completing this section.

__ 6. Log in to Salesforce.com.

__ 7. Click your name at the top right of the screen and select **My Settings**.

__ 8. Expand the **Personal** menu on the left side of the screen and select **Reset My Security Token**.

__ 9. Click the button **Reset Security Token**.



An email will be sent to your email address. Access this email, and make a note of the security token. You need this is token to set up the connection from App Connect Professional to your Salesforce.com account. Note also: every time you change your password, the security token is

changed as well. When you copy and paste the token into the endpoint configuration, it is important that you trim out any leading or trailing spaces as they might cause authentication errors.

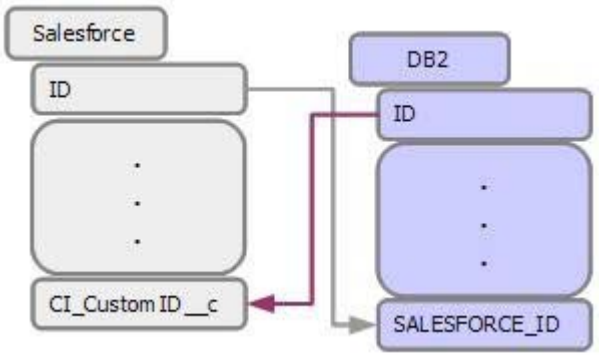
End of exercise

Exercise review and wrap-up

In this exercise, you set up an account, activated the access to your Salesforce account, and obtained the security token from within the proof of technology virtual machine.

Appendix B. Updating the Salesforce account object

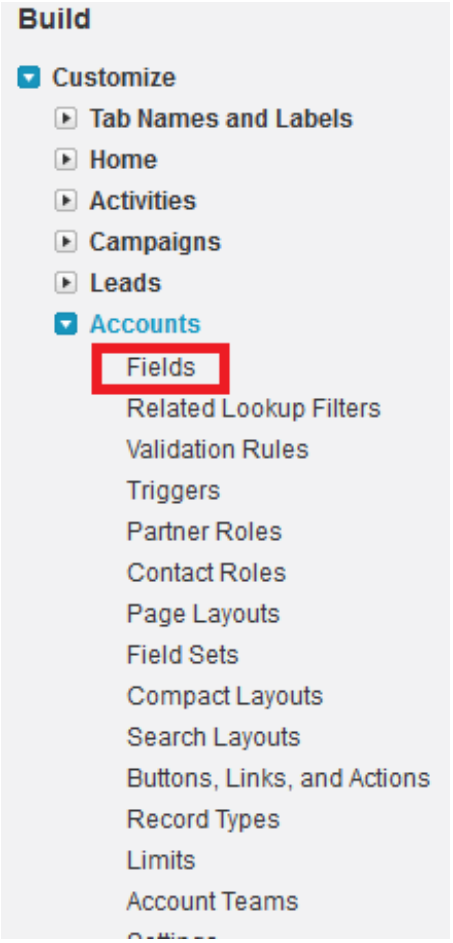
When coordinating the Salesforce Account with a legacy Account object in a database, it is customary for each object to define a field that contains the id value for the opposite environment.



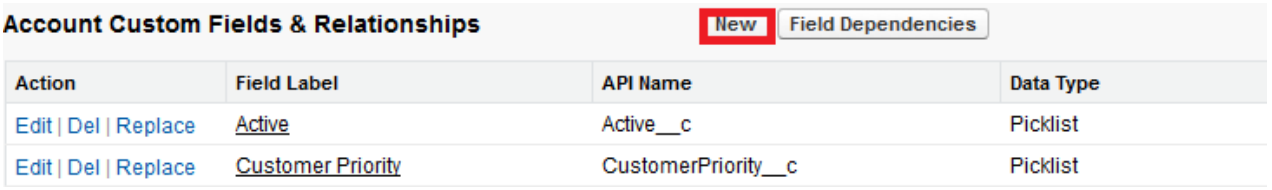
Use these steps to add an External ID field to your Salesforce Account.

__ 1. Log on to your **Salesforce Developer** account from http://developer.force.com.

__ 2. Go to **Setup > Customize** (under Build) **> Accounts > Fields**.



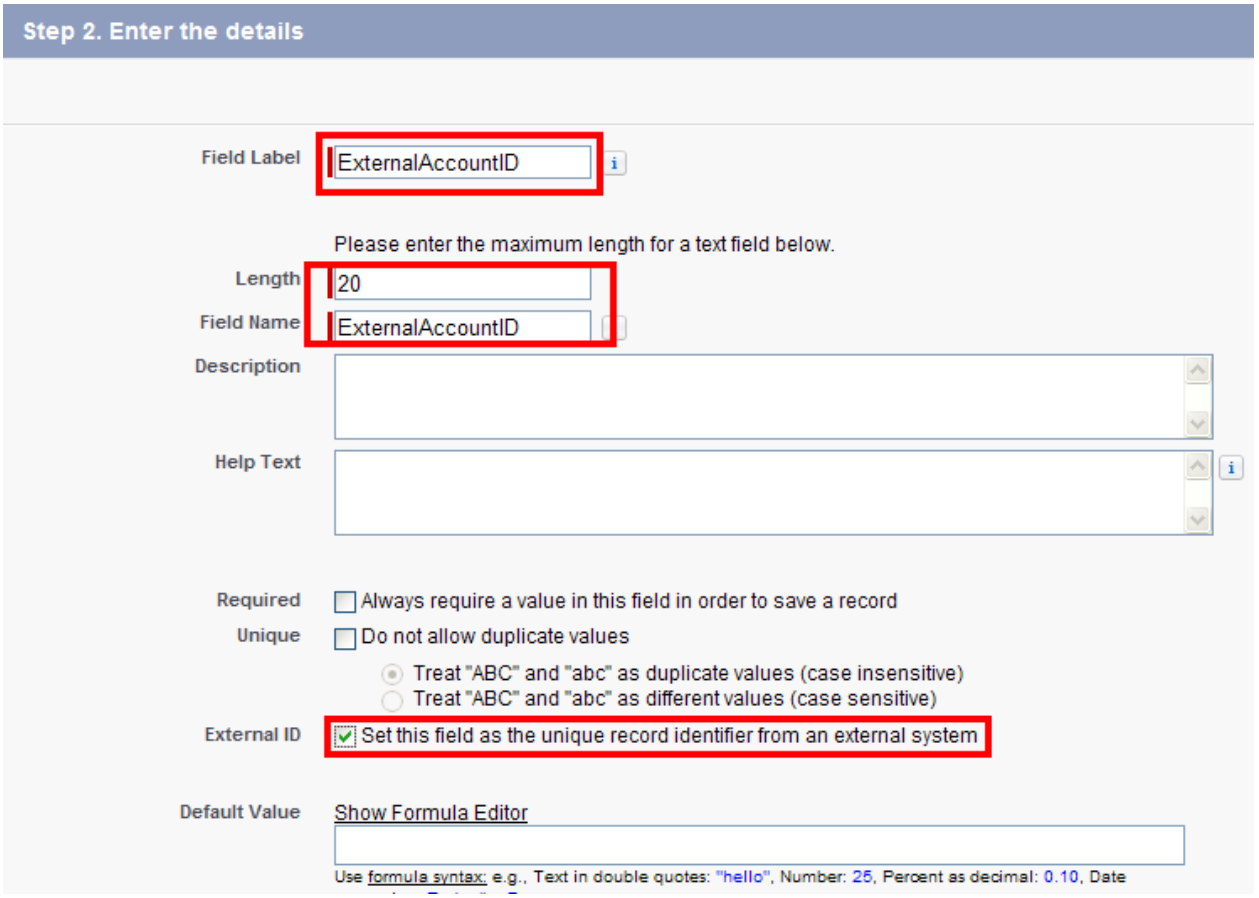
__ 3. Click the **New** button in the “Account Custom Fields & Relationships” section at the bottom.



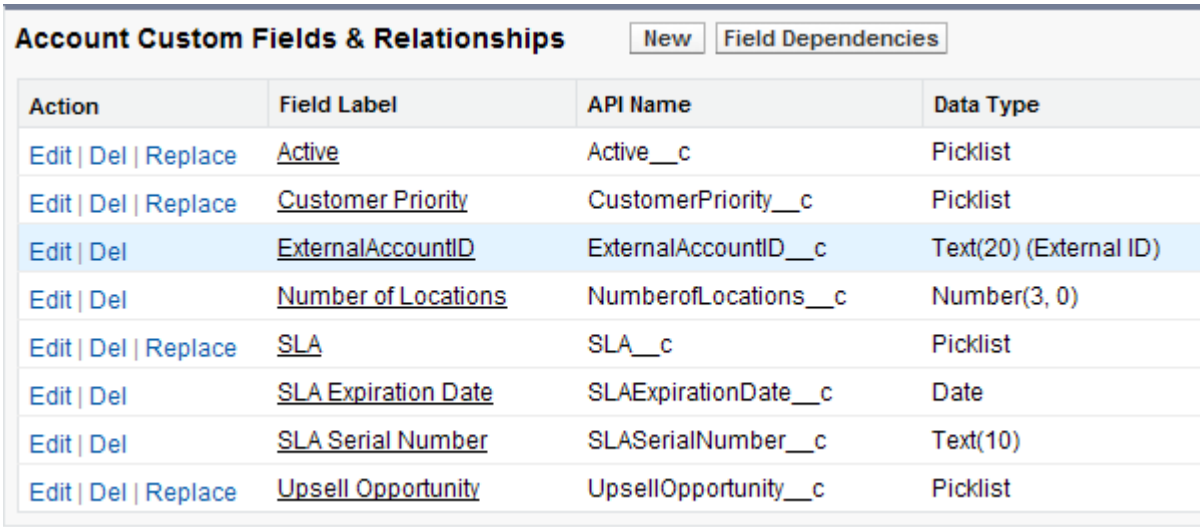
__ 4. On the first page, select **Text** as the data type and click **Next**.

__ 5. Enter `ExternalAccountID` for the **Field Label** and **Field Name**. The name that is shown in Studio is the API name, which is generated by appending `__c` to the Field Name: `ExternalAccountID__c`. Appending of `__c` is done automatically.

__ 6. Enter `20` for the field length, and select the **External ID** check box.



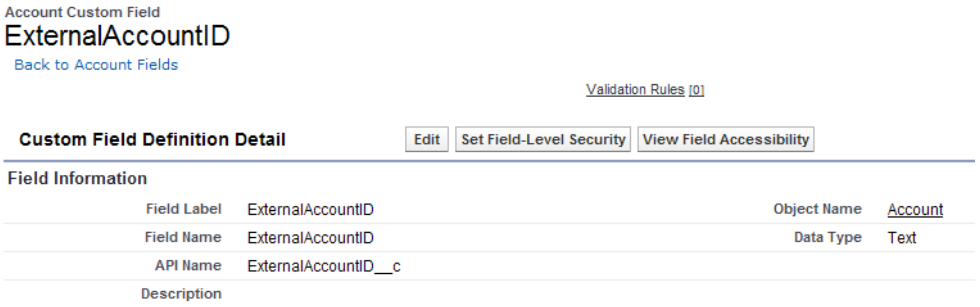
__ 7. Click **Next** and take the defaults to the end. You should see the new field listed as shown in the following figure.



Account Custom Fields & Relationships New | Field Dependencies

Action	Field Label	API Name	Data Type
Edit | Del | Replace	Active	Active__c	Picklist
Edit | Del | Replace	Customer Priority	CustomerPriority__c	Picklist
Edit | Del	ExternalAccountID	ExternalAccountID__c	Text(20) (External ID)
Edit | Del	Number of Locations	NumberofLocations__c	Number(3, 0)
Edit | Del | Replace	SLA	SLA__c	Picklist
Edit | Del	SLA Expiration Date	SLAExpirationDate__c	Date
Edit | Del	SLA Serial Number	SLASerialNumber__c	Text(10)
Edit | Del | Replace	Upsell Opportunity	UpsellOpportunity__c	Picklist

__ 8. Opening the field object shows the API field name.



Now, the ExternalAccountID__c field is listed as an External ID field in the **Salesforce Upsert Objects** activity.

End of exercise

Exercise review and wrap-up

In this exercise, you added an External ID field to your Salesforce Account.

IBM Training