Developing a plug-in connector for web service endpoints using WebSphere Cast Iron: Part 3: Implementing the ListObjects orchestration

Raghvendra Neelekani (raneelek@in.ibm.com)
Software Engineer
IBM

04 March 2015

Introduction
Part 2 of this tutorial series showed you how to implement the TestConnection orchestration for a Salesforce.com application, including using a Salesforce.com login operation to connect to a Salesforce.com application using a WebService endpoint. Part 3 shows you how to implement the ListObjects orchestration for your plug-in connector. The ListObjects orchestration is generated by the CDK wizard when any of the configured activities require dynamic discovery for their input or output schemas.

Implementing the ListObjects orchestration
The ListObjects orchestration is called by the connector when you click Browse as part of the activity configuration. The main purpose of the ListObjects orchestration is to return a list of objects back to the connector. Each object should have a name and a label, which are displayed in the Browse dialog. Therefore you need to implement the browsing of objects logic in this orchestration. Depending on your connector design, you can get the list of objects from the target application by making an API call, or you can hard code the objects list within the orchestration.

In Part 1, you configured two activities -- Create Objects and Get Server Timestamp -- in the CDK wizard. For the Create Objects activity, you selected the Input/Output requires dynamic discovery option, which means that this activity requires the browsing of objects feature and dynamic discovery of the schema for the objects. Since you selected this option in the CDK wizard, the CDK framework generated this ListObjects orchestration in which you need to implement the logic of getting the objects list.
When you open the generated ListObjects orchestration in Cast Iron studio, you will see two preconfigured activities:

**Figure 1. Generated ListObjects template orchestration**

- **Provide Service (ListObjects) activity** – Entry point for the list objects operation. From this activity, you will get access to all of the Connection fields in the Map Outputs section, as shown in Figure 1. These Connection fields are the ones you configured in the CDK wizard during orchestration generation.
- **Send Reply (Send Reply) activity** – Used to send the list of objects back to the connector.

To get the list of objects from the Salesforce.com application, Salesforce.com provides a SOAP API named describeGlobal in the partner.wsdl. The describeGlobal SOAP API fetches all standard and custom objects from the Salesforce.com application. Therefore, the ListObjects orchestration implementation for Salesforce.com involves using Cast Iron WebService Invoke Service activities to log in to Salesforce.com, invoke the describeGlobal operation, and send the result back to the connector. Here is the procedure to implement the ListObjects orchestration for Salesforce.com:

**Configure Provide Service activity**

1. Click the **Provide Service** activity in the orchestration and click **Map Outputs**.
2. Click **Copy=>**, select the variable **header** in the list of output parameters, and click **Create**.
   - The Connection fields are copied from a header parameter to an orchestration variable, which can be used by other activities.
3. Double-click the **header** variable on the To Orchestration side and rename it to **SalesforceConnectionFields**.

**Configure Salesforce Login activity**

1. Next you need a WebService endpoint to make webservice calls to Salesforce.com. If you already created the WebService endpoint for Salesforce.com in Part 2, then go to Step 4.
Otherwise, create a new endpoint; In the Projects tab, right-click **Endpoints => Create Endpoint** and select **WebService**.

2. In the WebService endpoint panel, click **Browse** on the WSDL Name field and select **partner.wsdl**. The Salesforce.com partner.wsdl will be loaded to this WebService endpoint, as shown in Figure 2. Therefore this WebService endpoint represents a Salesforce.com endpoint, which you can use to make calls to Salesforce.com using webservices defined in the partner.wsdl.

   **Figure 2. WebService Endpoint panel configured with Salesforce.com partner wsdl**

3. Rename this newly created WebService endpoint to **SalesforceEndpoint**: Right-click the newly created WebService endpoint and select **Rename**.

4. Drag and drop the **SalesforceEndpoint** endpoint to the orchestration pane between the Provide Service and Send Reply activities, and then and select the **Invoke Service** activity. This activity can be used to invoke web services to a Salesforce.com application.

5. Click the newly added **Invoke Service** activity in the orchestration. An activity checklist is displayed at the bottom.

6. Go to the Summary section, rename the activity to **Salesforce Login**, and press **Enter**.

7. Go to the Configure section and select **login** operation from the list of WSDL operations to configure this activity to make a Salesforce.com login webservice call.

8. Go to the Map Inputs section and you will see two fields displayed -- username and password. These fields are required for the Salesforce.com login operation.

9. The URL field is an optional parameter for this activity, so you need to enable it by adding optional parameters as shown in Figure 3. Right-click in the blank space in the To Activity side and select **Show Optional Parameters**. All optional parameters for this activity are displayed. You will see a field location to which you need to map the URL from the Connection fields.
10. Map the Salesforce Connection fields to this activity input: Click **Select Inputs**, select the **SalesforceConnectionFields** variable, and click **OK**. Map the Connection fields to the Salesforce Login activity, as shown in Figure 4:

**Figure 4. Mapping from Salesforce connection fields to WebService Invoke Service activity input**

11. Go to the Map Outputs section and copy the **body** to a new variable on the right side, and rename the variable as **loginResponse**.

**Configure Salesforce list objects activity**

Next you will configure one more Webservice activity to connect to the Salesforce.com application and fetch all of the standard and custom objects, using the Salesforce.com describeGlobal wsdl operation:

1. Drag and drop the **SalesforceEndpoint** endpoint to the orchestration between the Salesforce Login and Send Reply activities, and select the **Invoke Service** activity.
2. Click that new activity to get the checklist at the bottom.
3. Go to the Summary section, rename the activity to **Get Salesforce object list**, and then press **Enter**.
4. Go to the Configure section and select the **describeGlobal** operation from the list of WSDL operations to configure this activity to make a Salesforce.com describeGlobal webservice call.
5. Go to Map Inputs section and you will see one top-level element displayed -- describeGlobal. This operation does not require any data to be passed.

6. You need to provide login session id and server URL to this activity. To do that, enable all of the optional parameters. You will see location and header parameters.

7. Map the Salesforce login session id and server URL to this activity input; Click Select Inputs, select the loginResponse variable, and click OK. Map the login response fields to this activity as shown in Figure 5:

   **Figure 5. Mapping from Salesforce login response to WebService Invoke Service activity input**

   ![Map Inputs Diagram](image)

   8. Go to the Map Outputs section, copy the body to a new variable on the right side, and rename the variable to describeGlobalResponse.

**Configure Send Reply activity**

This section shows you how to configure the Send Reply activity to send the Salesforce.com list of objects back to the connector. The list of objects is in the describeGlobalResponse variable, and you use this variable in the Send Reply activity and map it to its input, as shown in Figure 6:

   1. Click on the Send Reply activity. You will see the Checklist for this activity at the bottom.
   2. Click Map Inputs.
   3. Click Select Inputs.
   4. Select the describeGlobalResponse variable from the list of variables and click OK.
   5. Map the name field to the objectType element and the label field to the label element on the right side.
You have now configured a valid ListObjects orchestration. A best practice is to add the Try-Catch block to the entire orchestration to handle error conditions.

**Configure Try-Catch block**

1. Right-click the initial orchestration image and select Add Catch Branch to add a catch block for the entire orchestration.
2. Drag and drop another Send Reply activity from SalesforceEndpoint to the CatchAll block. Rename this activity as Send Fault Reply.
3. Go to the Map Inputs section of this Send Fault Reply activity.
4. Provide a default value empty list to the label element. The complete orchestration is shown in Figure 7:

**Figure 7. Complete ListObjects orchestration**

You can verify the ListObjects orchestration using the Cast Iron Invoke Service utility.
1. Go to the Orchestration menu and click **Invoke Service**, as shown in Figure 8. The orchestration starts and opens the Invoke Service window.

**Figure 8. Selecting the Invoke Service utility**

2. Provide your Salesforce.com credentials in the UserName, Password, and URL fields and click **Execute**. The ListObjects orchestration is called and it fetches the result.

3. Click **Show Response** on the right side.

4. You will see the response from this orchestration on the right side window:
To verify the negative cases, provide invalid credentials and note the response from this orchestration.

**Conclusion**

Part 3 showed you how to implement a plug-in connector List Objects orchestration as part of plug-in connector development. In Part 4, you will learn how to implement a DescribeObject template orchestration in order to dynamically generate the schema feature for the connector.

**Acknowledgments**

The author would like to thank Sarath K. Ambadas and Srihari Na Kumar of IBM India for reviewing and providing valuable suggestions for this tutorial.
Resources

- **WebSphere Cast Iron resources**
  - **WebSphere Cast Iron Cloud Integration Knowledge Center**
    A single portal to all WebSphere Cast Iron Cloud Integration documentation, with conceptual, task, and reference information on installing, configuring, and using WebSphere Cast Iron.
  - **WebSphere Cast Iron Cloud Integration product library**
    Product announcements, case studies, white papers, and more.
  - **WebSphere Cast Iron Cloud Integration product page**
    Product descriptions, product news, training information, support information, and more.
  - **WebSphere Cast Iron Cloud Integration support**
    A portal for support problems and their solutions, plus downloads, fixes, problem tracking, and more.
  - **WebSphere Cast Iron Cloud Integration Community forums**
    Get answers to your technical questions and share your expertise with other WebSphere Cast Iron users.
  - **IBM Redbook: Getting started with WebSphere Cast Iron Cloud Integration**
    Detailed introduction to the development and administrative interfaces for WebSphere Cast Iron.
  - **Salesforce.com: SOAP API Developer's Guide**
    Salesforce provides programmatic access to your organization’s information using simple, powerful, and secure APIs.
  - **WebSphere Cast Iron Connector Development Kit (CDK) Developer’s Guide**
    For developers who want to use the WebSphere Cast Iron Studio development platform to develop and deploy plug-in connectors that extract data from application systems or other data sources.
  - **Developing a starter activity for plug-in connectors using the WebSphere Cast Iron Studio CDK**
    This developerWorks article shows you how to use the WebSphere Cast Iron CDK to develop a plug-in connector that has a starter activity, deploy the plug-in, and test it locally on WebSphere Cast Iron Studio.
  - **Hints and pointers for connector development in WebSphere Cast Iron Studio**
    This developerWorks article shows you how to easily configure the base orchestrations for connectors developed with the WebSphere Cast Iron Studio CDK wizard. Includes tips on password and fault handling.

- **WebSphere resources**
  - **developerWorks WebSphere**
    Technical resources for developers who use WebSphere and related middleware products. Includes downloads, how-to information, support resources, and a free technical library of tutorials, best practices, IBM Redbooks, and product manuals.
  - **Most popular WebSphere trial downloads**
    No-charge trial downloads for key WebSphere products.
  - **WebSphere forums and community**
Product-specific forums where you can get answers to your technical questions and share your expertise with other WebSphere users.

- **WebSphere demos**
  Download and watch these self-running demos, and learn how WebSphere products can provide business advantage for your company.

- **WebSphere-related tutorials on developerWorks**
  Nearly 3000 tutorials on WebSphere and related technologies by top practitioners and consultants inside and outside IBM. Search for what you need.

- **WebSphere-related books from IBM Press**
  Convenient online ordering through Barnes & Noble.

- **WebSphere-related events**
  Conferences, trade shows, Webcasts, and other events around the world.

- **developerWorks resources**
  - **Trial downloads for IBM software products**
    No-charge trial downloads for selected IBM® DB2®, Lotus®, Rational®, Tivoli®, and WebSphere® products.
  - **developerWorks blogs**
    Join a conversation with developerWorks users and authors, and IBM developers.
  - **Technical events for developers**
    Conferences, seminars, and technical events to deepen your technical expertise and help you succeed in your software projects.
  - **developerWorks podcasts**
    Listen to interesting and offbeat interviews and discussions with software innovators.
  - **developerWorks on Twitter**
    Check out recent Twitter messages and URLs.
  - **IBM Education Assistant**
    A collection of multimedia educational modules that will help you better understand IBM software products and use them more effectively to meet your business requirements.
About the author

Raghvendra Neelekani

Raghavendra Neelekani is a Staff Software Engineer for IBM in India. He has been working for IBM since 2009, and has extensive experience with Java and Web 2.0 technologies. He works on the WebSphere Cast Iron Development Team and has developed many Cast Iron connectors. You can contact Raghavendra at raneelek@in.ibm.com.