Developing a plug-in connector for web service endpoints using WebSphere Cast Iron: Part 1: Generating artifacts using the CDK wizard

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

04 March 2015

Introduction

IBM® WebSphere® Cast Iron Cloud Integration is used for data integration and migration across different on-premise and cloud-based applications. The Cast Iron Connector Development Kit (CDK) enables you to develop plug-in connectors for web service and ReST based applications without custom coding, and then plug these connectors into the product while minimizing product dependencies. This five-part tutorial series shows you how to develop a plug-in connector for a Salesforce.com application. Salesforce.com provides multiple WSDLs, each for a different use.

Objectives

Part 1 shows you how to use the Cast Iron CDK and the SOAP API exposed in the Salesforce.com partner.wsdl to create and configure a new Salesforce.com plug-in connector. In Part 1 you will learn about:

- Design considerations when developing a plug-in connector.
- Using the Cast Iron CDK wizard to create a plug-in connector for a Salesforce.com application, configure the connector properties, and generate activity orchestrations.

Prerequisites

You should be familiar with WebSphere Cast Iron Studio and its concepts of activities and orchestrations. You should also be familiar with using web services provided by SaaS-based applications such as Salesforce.com.
System requirements
You should have Cast Iron Studio installed. This tutorial series uses Cast Iron studio V7.0.0.2.

Designing your plug-in connector
The Cast Iron CDK is used to develop plug-in connectors for SaaS-based applications such as Salesforce.com, RightNow, and Dropbox, which expose web services or ReST APIs.

You do not always have to develop a plug-in connector to connect to a SaaS-based application. You can still use existing Cast Iron connectors, such as web service or HTTP connectors, to connect to a SaaS application. Here are some situations in which it is advantageous to develop a plug-in connector:

1. If the APIs and object schema structure of the SaaS-based application are complex and require significant time to understand and use them, then developing a plug-in connector will save subsequent users the time and effort of dealing with the APIs and schema structure.
2. If multiple teams in an organization are working on the same SaaS application, then developing a single plug-in that can be used by everyone will increase productivity and prevent repeated work.
3. If the APIs and object schema structure of a SaaS application change frequently, then developing a plug-in connector will let you make changes in once place, and seamlessly migrate the changes out to all connector users.

Before starting the development of plug-in connector, you should explore and finalize the following factors:

1. Understand the functionalities, object structures, and usage of the SaaS application.
2. Explore different methods to communicate with the SaaS application, such as web services or ReST APIs.
3. Explore different APIs and operations of the SaaS application using external tools such as SoapUI. You can also use Cast Iron Web Service and HTTP connectors to use and understand those APIs.
4. Finalize the connection properties required to communicate with the SaaS application.
5. Finalize the APIs to be used when developing the connector, based on your business use cases.
6. List the activities that need to be developed as part of the plug-in connector, based on your business needs.

If you take the Salesforce.com application as an example, it exposes a number of different APIs that you can use to integrate with Salesforce. For more information, see Salesforce.com APIs. In this tutorial you will use the Salesforce.com SOAP API (web services) to develop a plug-in connector.

For client applications, Salesforce.com exposes SOAP APIs in Partner Web Services WSDL. Download the latest partner WSDL from the Salesforce.com application and explore the different services exposed in it.
Cast Iron CDK Wizard

The Cast Iron CDK is completely wizard-based and enables the building of orchestrations. While using the CDK wizard, you need to provide the connector properties with which it generates template orchestrations and other artifacts required for the connector. Each template orchestration corresponds to an activity or operation in the connector. So connector development consists of implementing these template orchestrations with your custom implementation using the existing Cast Iron Web Service and HTTP connector activities to invoke web service and ReST APIs, respectively.

Here are the different template orchestrations for each activity and operation that the Cast Iron CDK wizard generates:

1. **TestConnection orchestration** – Corresponds to the Test Connection functionality of the connector with which you can test the connection to a target application. Therefore you need to implement the test connection functionality in this orchestration, and it appears in the connector’s endpoint panel.

2. **ListObjects orchestration** – Corresponds to the Browsing of objects functionality in the connector activities. This functionality lets you list all objects for user selection in that activity. If you do not want Browsing of objects functionality for any connector activities, you can disable it in the CDK wizard, and the ListObjects orchestration will not be generated.

3. **DescribeObject orchestration** – Used to generate input and output schema structure of the selected object.

4. **Activity orchestration** – The CDK generates a separate orchestration for each activity configured in the CDK wizard. Each orchestration is named with the activity name, and you need to implement these orchestrations based on what the activity should do.

Create a new plug-in connector using the Cast Iron CDK wizard

1. Open Cast Iron Studio V7.0 or later.
2. Select File => New => Create New Connector Project, as shown in Figure 1:
Figure 1. Opening the New Connector Project Wizard

3. Specify the location to save the connector project by clicking Browse. Type `SalesForceDemoConnector` as the name of the connector project and click **OK** to launch the Cast Iron CDK Wizard. The Connector Wizard tab opens.

4. Specify the following connector details in the Connector Information page:
Developing a plug-in connector for web service endpoints using
WebSphere Cast Iron: Part 1: Generating artifacts using the CDK
wizard

Figure 2. Connector wizard – connector information

<table>
<thead>
<tr>
<th>Name</th>
<th>SalesforceDemoConnector</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.0.0</td>
<td>This is a demo plug-in connector for Salesforce.com application.</td>
</tr>
</tbody>
</table>

Add WSDL or Schema Files

Add WSDL/Schemas: partner.wsdl

5. Click Next. The Add Connection Fields page is displayed.

Configuring the connection fields

In the Add Connection Fields page, you need to specify the connection properties that are used to communicate with the target application. These connection fields are displayed on the connector endpoint panel and available in all template orchestrations. The order of the fields in this section is maintained in the Endpoint panel. By default, the user name and password fields are preconfigured.

1. The Salesforce.com login requires the Username, Password, and URL fields. Hence you just need to add one more field with name URL of type STRING and format STRING, as specified below:
   a. Click the + button. A new entry field0 is added in the connection fields.
   b. Double click on field0, change the name to URL, and press the Enter key.
   c. You can also provide default values for any of the connection fields. Specify the default value https://login.salesforce.com/services/Soap/u/32.0 in the Connection Field
Default Value column for the URL field and press **Enter**. This default value is taken from partner.wsdl.

**Figure 3. Add connection fields**

2. If the connector has to communicate with the target application through a proxy server, then select the checkbox **Add Proxy Connection Details to the Endpoint** to add additional proxy properties in the Connector Endpoint panel. In this tutorial, the proxy configuration will not be used.

3. Click **Next**. The Add Activities page is displayed.

**Adding activities**

In the Add Activities page, you need to configure the activities required for the connector. This tutorial illustrates connector development with these two activities:

- **Create Objects activity** – Used to create a new record of any object type in the Salesforce.com application. You must select the object type to configure this activity, and you must configure this activity to have Browse functionality. You must also configure input and output schema structure for this activity.
- **Get Server Timestamp activity** – Used to get the Salesforce.com application server timestamp information.

**Configure Create Objects activity**

Developing a plug-in connector for web service endpoints using WebSphere Cast Iron: Part 1: Generating artifacts using the CDK wizard
To create objects in a Salesforce.com application, you need to use the Create operation from partner.wsdl. Here is the input and output schema structure for the Salesforce.com Create operation:

**Listing 1. Salesforce.com create operation -- Input schema**

```xml
<element name = "create">
    <complexType>
        <sequence>
            <element name = "sObjects" type = "ens:sObject" minOccurs = "0" maxOccurs = "unbounded"/>
        </sequence>
    </complexType>
</element>
<!--Dynamic sObject -->
<complexType name = "sObject">
    <sequence/>
</complexType>
```

**Listing 2. Salesforce.com create operation -- Output schema**

```xml
<element name = "createResponse">
    <complexType>
        <sequence>
            <element name = "result" type = "tns:SaveResult" minOccurs = "0" maxOccurs = "unbounded"/>
        </sequence>
    </complexType>
</element>
<complexType name = "SaveResult">
    <sequence>
        <element name = "errors" type = "tns:Error" minOccurs = "0" maxOccurs = "unbounded"/>
        <element name = "id" type = "tns:ID" nillable = "true"/>
        <element name = "success" type = "xsd:boolean"/>
    </sequence>
</complexType>
<complexType name = "Error">
    <sequence>
        <element name = "fields" type = "xsd:string" nillable = "true" minOccurs = "0" maxOccurs = "unbounded"/>
        <element name = "message" type = "xsd:string"/>
        <element name = "statusCode" type = "tns:StatusCode"/>
    </sequence>
</complexType>
```

From the schemas displayed in Listings 1 and 2, the input structure for the Create operation is a Salesforce.com object schema that extends sObject. Therefore, based on the object selected during discovery, the schema will be created dynamically and displayed in the map inputs. If you find any elements under the sObject sequence, comment them out and browse partner.wsdl again in the CDK wizard Connector information page to make it consumable by Cast Iron Studio. You can browse the output schema `createResponse` element directly from partner.wsdl.

Configure the Create Objects activity as shown in Figure 4. By default, the Add Activities section displays a single activity entry. You can rename or delete this entry.
1. Double-click activity0, change the name to Create Objects, and press Enter.
2. Select Input/Output Parameter Requires Dynamic Discovery, which enables the Cast Iron CDK framework to generate ListObjects and DescribeObject template orchestrations.
3. Select Input is Discovered Dynamically, which means you are dynamically discovering the schema for Create Objects activity input.
4. Click Select Input, select the create element under partner as shown in Figure 5, and then click OK:
5. Click **Select Output**, select the **createResponse** element under partner, and then click **OK**.

**Configure Get Server Timestamp activity**

To get the timestamp information, Salesforce.com provides a direct web service operation `getServerTimestamp`. Hence you do not need to discover the input/output structure for this activity dynamically. With this information, configure the Get Server Timestamp activity, as shown in Figure 6:
Figure 6. Configure Get Server Timestamp activity

1. Click on the + button.
2. Double-click activity1, change the name to Get Server Timestamp, and press Enter.
3. Click Select Input, select the getServerTimestamp element under partner, and then click OK.
4. Click Select Output, select the getServerTimestampResponse element under partner, and then click OK.
5. Click Next. The Create Orchestrations and Artifacts window is displayed. Click Yes.
6. Then Final Summary page is displayed with a summary of generated artifacts. Click Finish.
7. The Final Summary page closes and you see the template orchestrations and other artifacts generated for the connector development, as shown in Figure 7:
Figure 7. Final summary of connector configuration and generated artifacts

Conclusion

Part 1 of this tutorial series showed you how to create a new plug-in connector for a Salesforce.com application and configure the connection fields and activities using the Cast Iron CDK wizard. In Part 2, you will learn how to build a TestConnection template orchestration to implement the Test Connection 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

Raghvendra 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 Raghvendra at raneelek@in.ibm.com.

© Copyright IBM Corporation 2015

Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)