**Intended Audience:**

This document would help user/customer/Informatica Cloud vendor who would like to use Informatica Cloud Platform for integrating data from ReST endpoint to other data source or vice-versa.

**Brief Description about ReST Connector:**

Informatica Cloud ReST Connector allows you to integrate data with REST based web service applications that are either internal to an organization or hosted externally. The connector supports Web Services that conform to REST grammar. This connector supports almost frequently used ReST use-cases and also provides flexibility to user to implement ReST based data integration to their custom service endpoint, which does not conform to commonly known definitions/Authentications/designs.

**Why Use ReST Custom Authentication?**

ReST connector supports following type of authentications:

Basic

Digest

OAuth  
 No Authentication (*Atom* *type of request/response*)

In some cases you might have your own authentication mechanism to connect to the REST Web Service or, the mechanism you have may not fit into the options mentioned above. This document would help you to access REST web Services and build integration using Informatica ReST connector.

We have attached sample ReST CustomAuth project which would help you start with development. Please find the steps you need to follow to setup a ReST CustomAuth solution.

**How to Implement CustomAuth ReST Integration?**

**Things you need to setup ReST-Integration Development Env**

To start with ReST integration development environment, you might need the following:

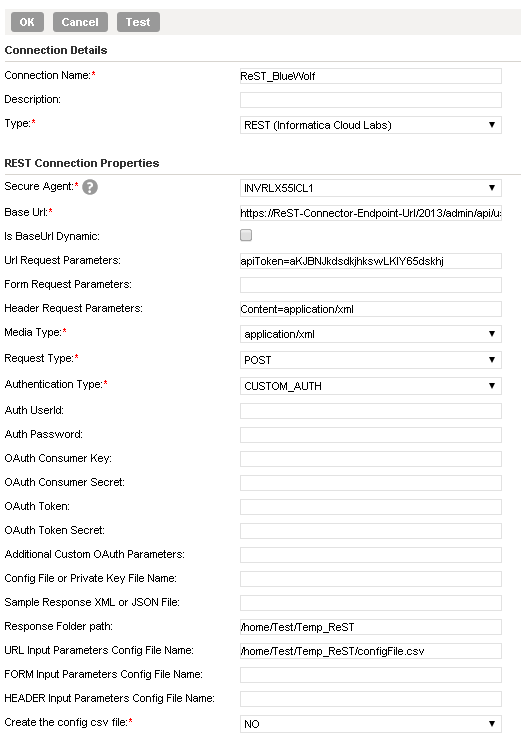
1. Informatica Cloud Subscription (*Trial account/org should be fine*)
2. Download & Install Informatica Cloud secure agent
3. Eclipse IDE (Kepler or anything latest)
4. Download the ReST Custom Project from gitHub
5. Fundamental (or) Basic Java programming knowledge
6. ReST endpoint, to which you need to connect and integrate to

**How to import Eclipse Project?**

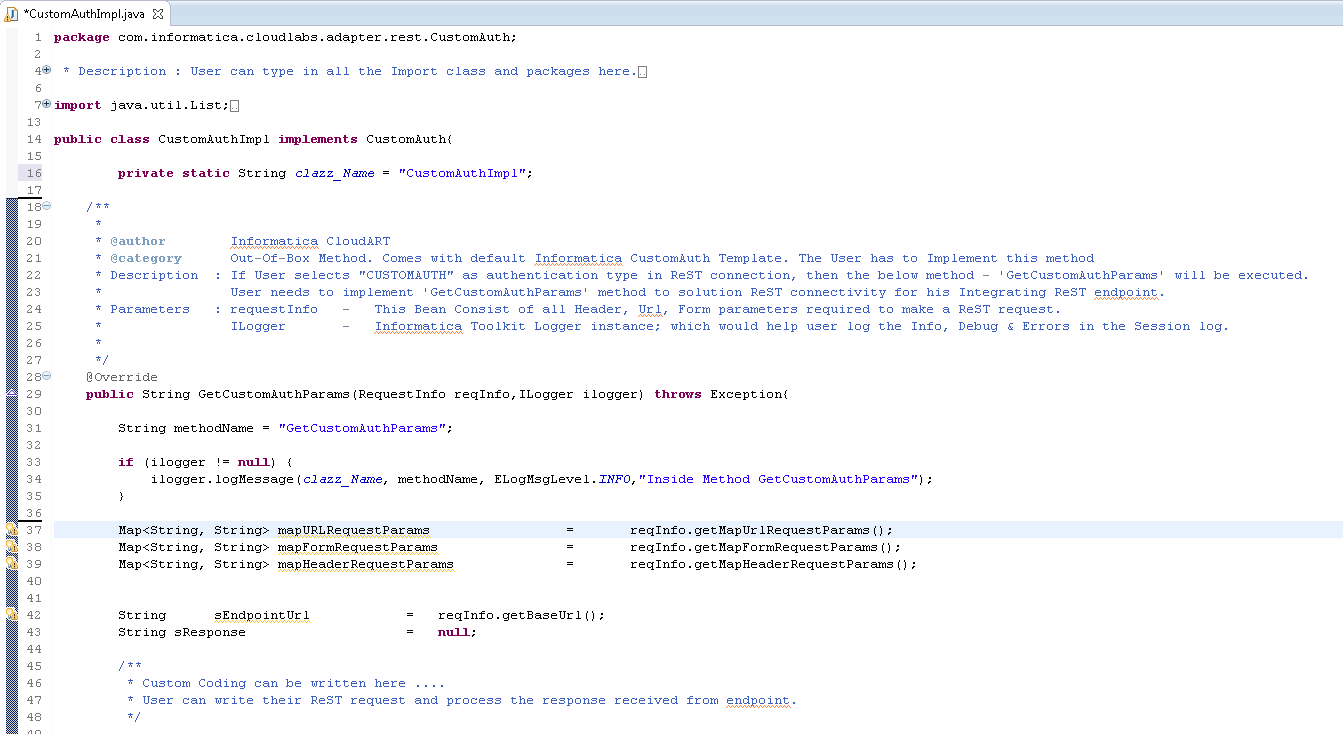
1. Download ReST-CustomAuth project from the url <http://example>
2. Unzip the downloaded file.
3. Open the Eclipse IDE.
4. Go to File🡪Import🡪’Existing Projects into Workspace’ and browse the ReST-CustomAuth Project folder.
5. You would be customizing the ReST connector in the Java file ‘CustomAuthImpl’ (package - *com.informatica.cloudlabs.adapter.rest.CustomAuth*)

**How to Implement Custom Coding using Informatica ReST Connector template?**

ReST Connector – Connection configuration Page

****

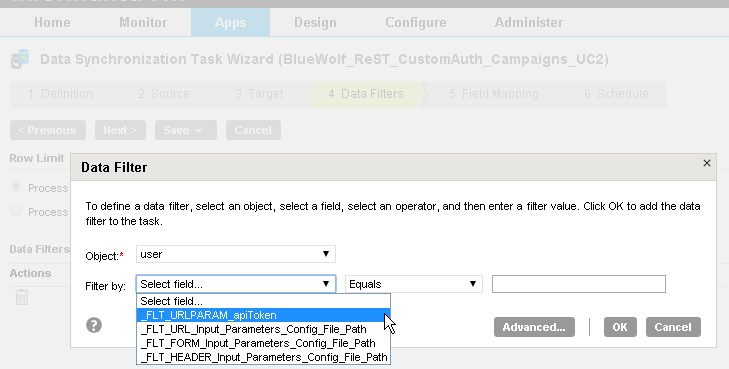
Compare & Contrast ReST Connection Attribute with Java Code Template



Compare and contrast Java bean Methods with Connection Attributes

|  |  |  |
| --- | --- | --- |
| Connection Attribute  (*as seen in UI*) | Java Bean Attribute  (RequestInfo) | Description |
| **Base Url** | getBaseUrl() | User can get the baseUrl specified in the connection page using this method. |
| **Is BaseUrl Dynamic** | getIsDynamicBaseUrl() | If the ‘*is baseUrl Dynamic*’ check box checked, then this method would return ‘**true**’ else ‘**false**’ will be returned. |
| **Url Request Parameters** | NA | Not available in RequestInfo Bean. This method facilitates metadata for DS task creation. |
| **Form Request Parameters** | NA | Not available in RequestInfo Bean. This method facilitates metadata for DS task creation. |
| **Header Request Parameters** | NA | Not available in RequestInfo Bean. This method facilitates metadata for DS task creation. |
| **Media Type** | getMediaType() | Provides the “media-type” defined in the connection page. |
| **Request Type** | getReqType() | HTTP request type defined by the user in connection page. |
| **Authentication Type** | getsAuthType() | Defines the type of Authentication. Code flow comes to CustomAuthImpl class, only if the user selects ‘CUSTOM\_AUTH’ as the Authentication Type. |
| **Auth UserId** | getsAuthUserId() | Auth Userid value can be retrieved using this method |
| **Auth Password** | getsAuthPassword() | Auth Password value can be retrieved using this method |
| **OAuth Consumer key** | getsOAuthConsumerKey() | OAuth Consumer Key can be fetched using this method from the bean. Valid only if user selects OAuth as Auth type. |
| **OAuth Consumer Secret** | getsOAuthConsumerSecret() | OAuth Consumer Secret can be fetched using this method from the bean. Valid only if user selects OAuth as Auth type. |
| **OAuth Token** | getsOAuthToken() | OAuth Token can be fetched using this method from the bean. Valid only if user selects OAuth as Auth type. |
| **OAuth Token Secret** | getsOAuthTokenSecret() | OAuth Token Secret can be fetched using this method from the bean. Valid only if user selects OAuth as Auth type. |
| **Additional Custom OAuth Parameters** | getsAdditionalCustomAuthParam() | Custom parameter, if any which users wants specified. |
| **Config File or Private Key File Name** | getConfigKeyOrPrivateKeyFileName() | FileName (*with Path*) for Customization. Optional field. |
| **Sample Response XML or JSON File** | getXmlPath() | FileName (*only path*) for writing the ReST response to a file. |
| **Response Folder Path** | getResponseFilePath() |  |

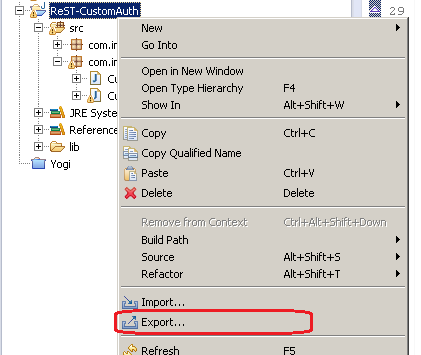
Compare and contrast Java bean Methods with Data Filter Fields

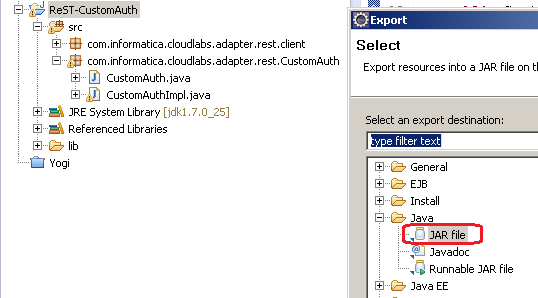
****

|  |  |  |
| --- | --- | --- |
| Data Filter Attribute | Java Bean Attribute | Description |
| \_FLT\_URL\_Input\_Parameter\_Config\_File\_Path | getMapUrlRequestParams() | The ReST Connector read the Url parameter file defined in the data filter and gives the collection of Url Parameter values |
| \_FLT\_FORM\_Input\_Parameter\_Config\_File\_Path | getMapFormRequestParams() | The ReST Connector read the FORM parameter file defined in the data filter and gives the collection of Url Parameter values |
| \_FLT\_HEADER\_Input\_Parameter\_Config\_File\_Path | getMapHeaderRequestParams() | The ReST Connector read the HEADER parameter file defined in the data filter and gives the collection of Url Parameter values |
|  |  |  |

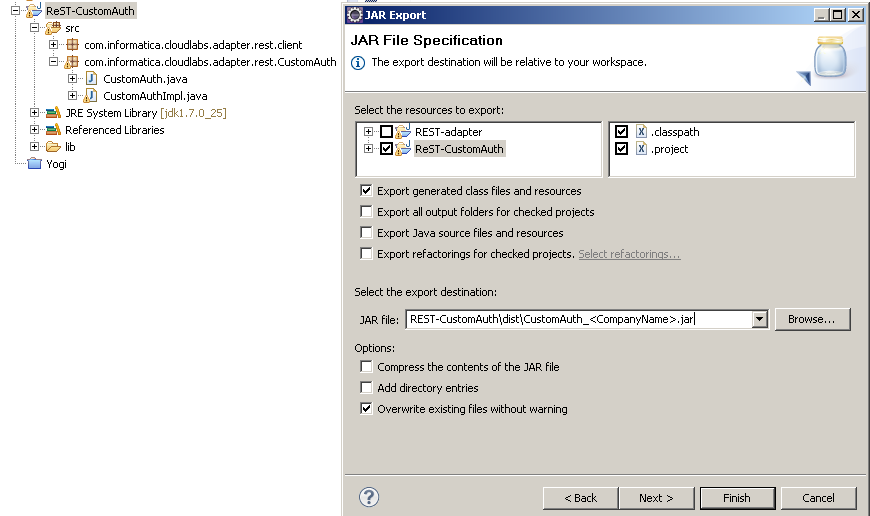
**How to Create ReST-CustomAuth Jar?**

Following are the steps to create a CustomAuth Jar using eclipse and CustomAuth Template Project.

1. Right click on the “ReST-CustomAuth” project (*in package explorer*) in eclipse tool. Select Export option
2. Choose “Jar File” (*under Java folder*). Click Next



1. Select option to export only *class files*. Select the jar file export path



1. Click ‘*Next*’ till ‘*finish*’ button is enabled.

**How to Deploy the CustomCode (*CustomAuth*)?**

* After exporting the ReST-CustomAuth jar file, user need to deploy the Jar file in Secure agent before creating or running a DS task.
* Stop the Secure agent.
* Copy the Custom Auth Jar to the following path of ReST connector in your Secure agent:
  + <*agent install path*>\main\bin\rdtm\javalib\<plugin\_id>
  + <*agent install path*>\main\tomcat\plugins\<plugin\_id>
* If you already have a customAUth Jar with same name, over write the existing jar.
* Never take a backup of the custom Auth or any other Jar in the secure agent directory or sub-directory.
* Start the secure agent

**FAQs?**