**Integration Framework**

**Implementation**

**Understanding Integration Flow**

**Version 1.2**

**Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Issue Date | Description of Version / Changes | Author |
| 1.0 | 2/2/2018 | Initial Draft | Amiya Das |
| 1.1 | 8/2/2018 | Reviewed & Corrected. Added execution flow and diagram. | Vikesh Gajula |
| 1.2 | 24/08/2018 | Added details of Connected app, Auth. Provider and Named credentials | Vikesh Gajula |

**Introduction:**

Salesforce allows you to integrate with external SOAP and REST Web services using callouts. You can use utilities for JSON, XML, data security, and encoding.

Salesforce callout enables you to tightly integrate your Apex with an external service by making a call to an external Web service or sending a HTTP request from Apex code and then receiving the response. Apex provides integration with Web services that utilize SOAP and WSDL, or HTTP services (Restful services).

This document explains the framework which is generic and can be used to integrate salesforce with external system using REST & SOAP API callout.

**Advantages of using this framework:**

* Fixed pattern followed by developers to implement a service
* Reduce code redundancy
* Minimum development effort
* Easy to understand and maintain code
* Configurable option to log details of every service call in a custom object for analyzing failed calls, callout time, reporting, etc.
* Input & Output processing is handled within framework
* Headers including certificate are set using custom setting within framework
* Basic authentication (Username & Password) and certificate usage is supported
* Configurable email alerts in case of failure of service callouts

**To Do in framework:**

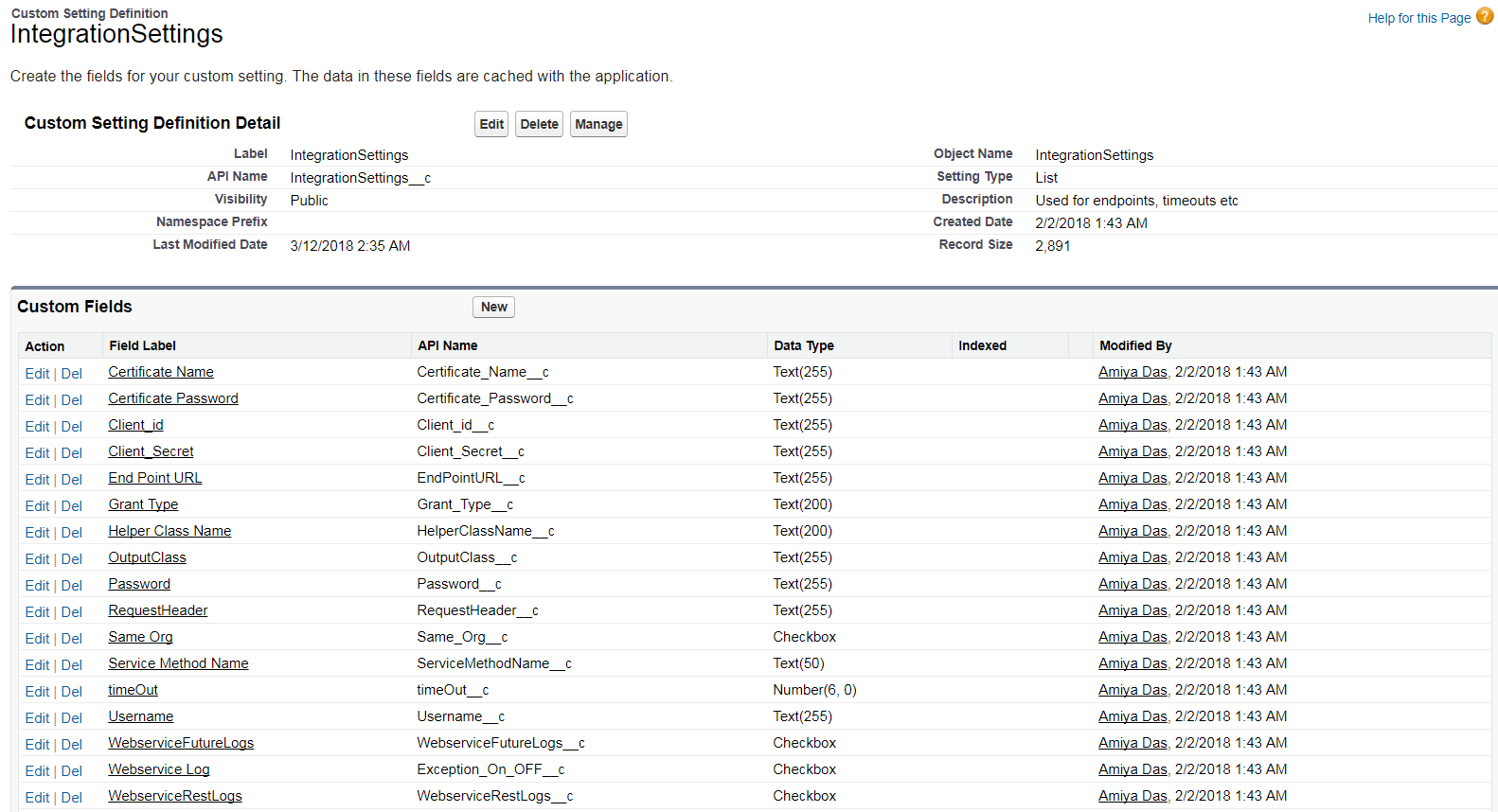
* Multiple authentications like Oauth, User-Agent, etc.
* Http callouts using XML format as request

**REST FRAMEWORK**

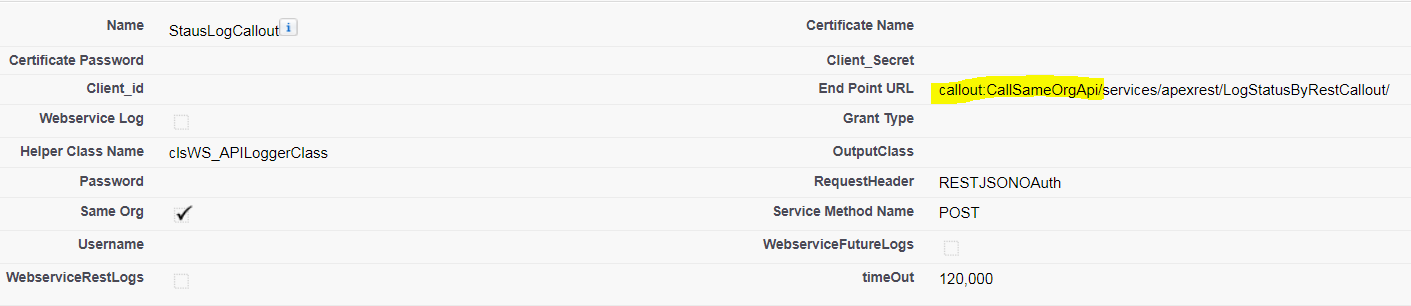
**Custom settings:**

1. [**IntegrationSettings**](https://jlrcustomerportal-dev-ed.my.salesforce.com/01I7F000001V2R2?setupid=CustomSettings)**:** This is a custom setting where we store all the details required to make a call out to the external system such as [End Point URL](https://jlrcustomerportal-dev-ed.my.salesforce.com/00N7F00000Dm8fe?setupid=CustomSettings), API Key, Client Id, [Client Secret](https://jlrcustomerportal-dev-ed.my.salesforce.com/00N7F00000BhGeT?setupid=CustomSettings), Agent, Timeout, User Name , Password, etc. Also the flag values such as [Webservice Log](https://jlrcustomerportal-dev-ed.my.salesforce.com/00N7F00000CjfU3?setupid=CustomSettings) & [WebserviceRestLogs](https://jlrcustomerportal-dev-ed.my.salesforce.com/00N7F00000Dmw0x?setupid=CustomSettings) used to enable the callout details logging are stored in this custom setting.

Create an entry for every service with corresponding details. If same endpoint is used for multiple services but different methods / operations then create separate entry for each method / operation.

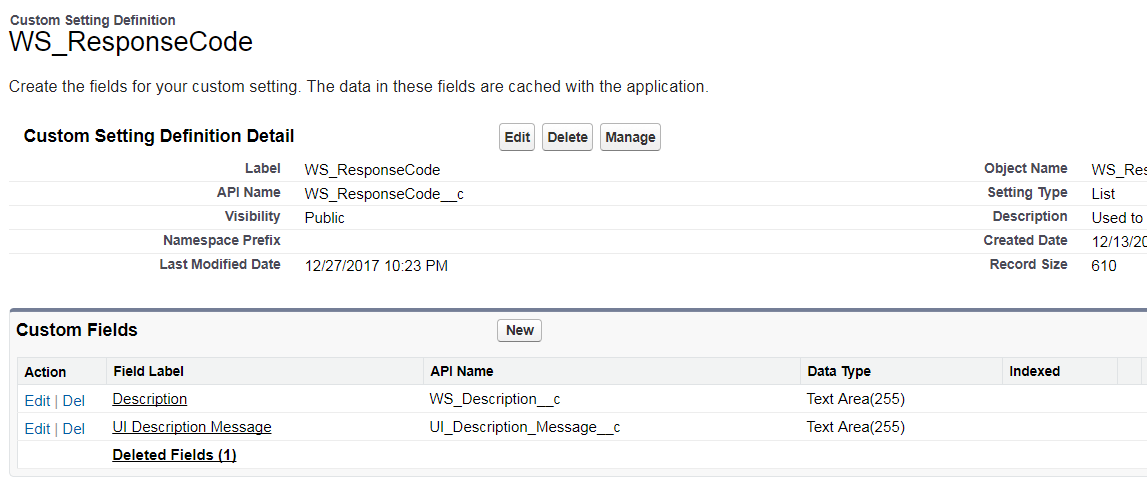


The record for services using named credentials should have URL as highlighted below without the domain name and using named credentials.

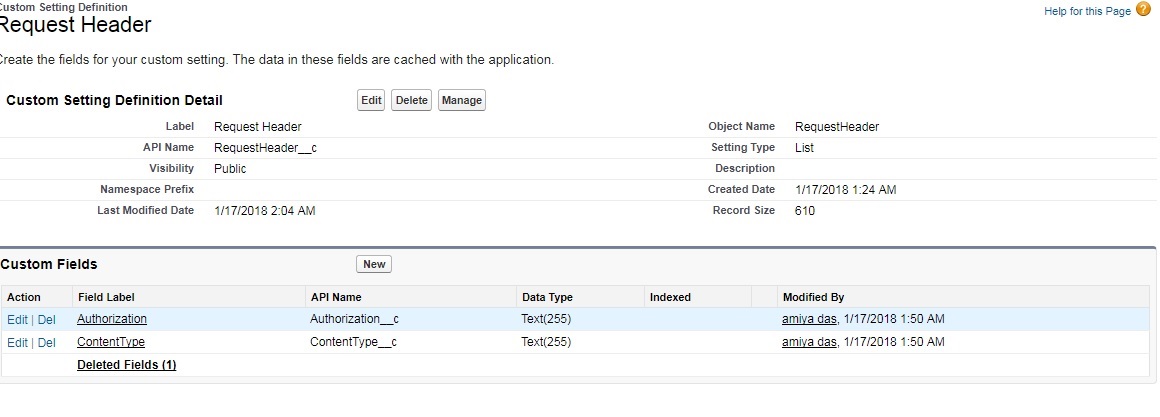


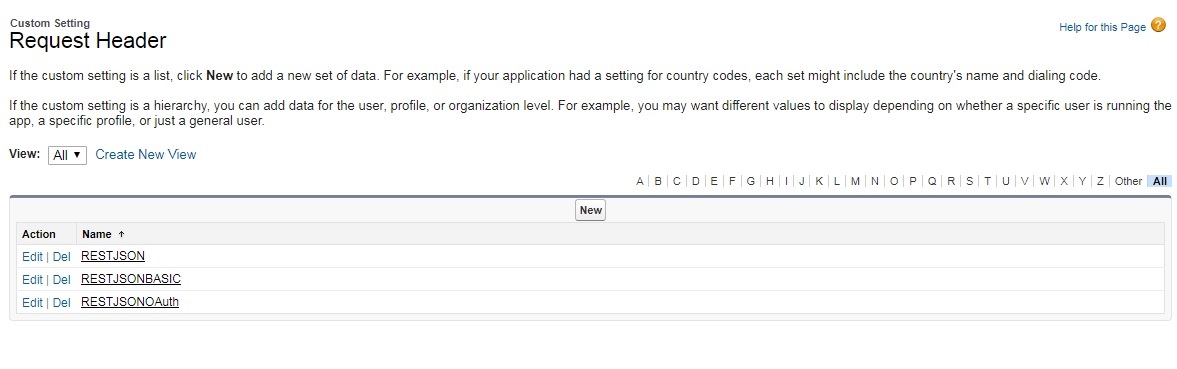
1. [**WS\_ResponseCode**](https://jlrcustomerportal-dev-ed.my.salesforce.com/01I7F000001V40I?setupid=CustomSettings): This custom setting is used to store the list of response code with user-friendly success/error messages.

If an error code is not present then create an entry for same with corresponding message.



1. **Request Header:** This custom setting is used to store values such as authorization and content types.

****



**Apex Classes:**

1. **BaseIntegration:** These is the generic abstract integration class which is used for both REST and SOAP integration. Below are the methods used in this class: -

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Description** | **Input Parameter** | **Output Parameter** |
| getCertificate | This method is used to get client certificate by calling *IntegrationHelper* class (discussed later) and *getClientCertificate* method via certificate name stored in [*IntegrationSettings*](https://jlrcustomerportal-dev-ed.my.salesforce.com/01I7F000001V2R2?setupid=CustomSettings) | null | null |
| populateEndPoint | As the name suggests this method is used to populate end point by using *EndPointURL\_\_c* stored in [*IntegrationSettings*](https://jlrcustomerportal-dev-ed.my.salesforce.com/01I7F000001V2R2?setupid=CustomSettings)*. It internally calls add parameters method to replace or add parameters in URL.* | Null | null |
| addParameters | This method is used for adding parameters to endpoint if available | Endpoint as a string, additional parameters as map | Endpoint as a string |
| getRequestHeaders | This method is used to get request headers by calling *IntegrationHelper* class and *getRequestHeaders* method by passing integrationSettings instance as a parameter. | null | null |
| performCallout | This method is used to prepare request, perform callout, process response and do logging. This method in turn calls the populateEndPoint, getRequestHeaders & getCertificate methods. | Input as instance of Object | Object instance as output. |
| logServiceStatus | This method is used for creating an entry in Webservice logs object for every service call. | null | null |

1. **RESTIntegration:** This class extends *BaseIntegration* class and hence inherits all properties and methods in parent class. The main purpose of this class is for preparing request, hitting the service and then processing the response. Methods used in this class are as below: -

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Description** | **Input Parameter** | **Output Parameter** |
| callout | This method is used to call *performCallout* method in *BaseIntegration* class | Object instance | Object |
| prepareRequest | This method is used for setting values of HTTP request instance for values such as timeout, service method name, certificate name & password, request headers, endpoint, etc. This method also serializes the request as JSON and set as request body. | null | null |
| execute | This method is used to call http.send method by passing the request and getting the response. | null | null |
| processResponse | This method is used for processing the response by de-serializing the response body. It reads the status code from response and display the status(success/error), error code and error message accordingly. If output class mentioned in Integrationsettings for the service then response would be de-serialized and type casted to instance of that output wrapper class. | Null | null |
| setMethodForCallout | This method is used to set the operation such as get, post, patch by using the ServiceMethodName defined in custom settings. | null | null |

1. **IntegrationHelper: -** This helper class is used to get request headers, get client certificate name and processing http response codes stored in custom settings *WS\_ResponseCode\_\_c.* Methods included are as below: -

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Description** | **Input Parameter** | **Output Parameter** |
| getRequestHeaders | This method is used to read authorization details from IntegrationSettings, content type from RequestHeader settings and prepare a map of key-value pair. This key-value pair map will be used to set the header values in request. | IntegrationSettings\_\_c (custom settings) instance | Map containing strings as key value pair for returning headers |
| getClientCertificate | This is used to get the client certificate by comparing the certificate name with string passed as a parameter. | Certificate name as a string | Certificate as string |
| getWebResponseCode | This is used to fetch the response codes record for status code passed. Custom setting name *WS\_ResponseCode\_\_c* | Status code as a string | Instance of *WS\_ResponseCode\_\_c* |

1. **WebserviceLogHelper**: This class is used for inserting Webservice logs. Based on value of webserivce rest logs flag defined in custom settings, the record would either be inserted via DML or via REST callout. Methods included are as below: -

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Description** | **Input Parameter** | **Output Parameter** |
| logStatus | Used for inserting WebService\_Log\_\_c record based on Boolean value for WebserviceRestLogs field defined in custom settings. If false then inserted via DML else by calling the *logStatusByRESTServiceCall* defined below. | Instance of *WebService\_Log\_\_c,* Boolean value for WebserviceRestLogs | null |
| logStatusByRESTServiceCall | Creating the instance of *RESTIntegration* class(constructor) and passing the integration settings named *StausLogCallout.* Calling callout method passing serialized instance of WebService\_Log\_\_c. | Null | null |

1. **LogStatusRestCallout: -** This is a Rest Webservice class with endpoint defined as ‘/LogStatusByRestCallout/\*’.

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Description** | **Input Parameter** | **Output Parameter** |
| createLogs | This method is used for inserting web service logs when the ‘webserviceRestLogs’ flag is true defined in custom settings and catching the exception if any. It also calls an instance of wrapper class *clsWS\_webLog4RestCallClass (*discussed below) and desterilizing the JSON and storing in webservice log object. | Null | null |

1. **WebserviceLog: -** This is a wrapper class for webservice\_logs\_\_c object. All the variables declared such as jsonString, Method\_Type, Response\_Description, etc. are used to create logs in Webservice log object. Method included is as below: -

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Description** | **Input Parameter** | **Output Parameter** |
| Parse (Not required) | This methodis used to deserialize the json passed as string | Json as a string | Json in deserialized format |

1. **CallOutResults: -** WrapperClass for holding response and error details of the callout.

**Note: You need to have a domain before you create the below connected app, auth. Provider and Named credentials for smooth authentication.**

**Connected App:**

1. **CallSameOrgApi\_app:-** This connected app will be used for authenticating the request when calling the REST api service of same Salesforce org. As part of framework this will be used to insert webservice log records via REST callout but it can be extended for making callouts to other REST services of same org.

Below is the screenshot for reference:

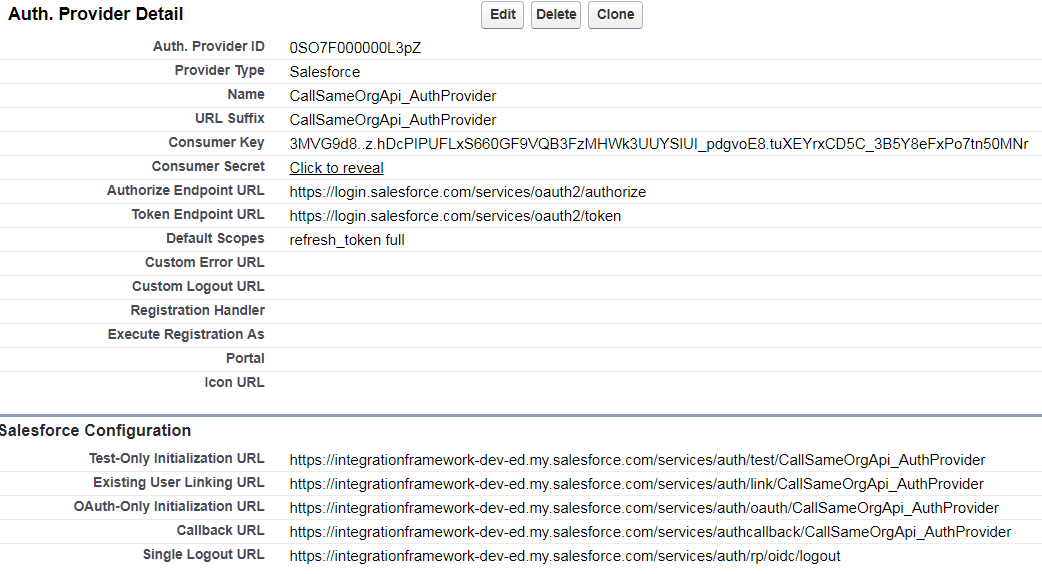


While creating the app keep the callback URL value as ‘TEMP:VALUE’. Once the Auth. Provider is created then replace the callback URL value with the callback URL from Auth. Provider.

**Auth. Provider:**

1. **CallSameOrgApi\_AuthProvider :-** This authentication provider will be used for authenticating the request when calling the REST api service of same Salesforce org. As part of framework this will be used to insert webservice log records via REST callout but it can be extended for making callouts to other REST services of same org.

Below is the screenshot for reference:

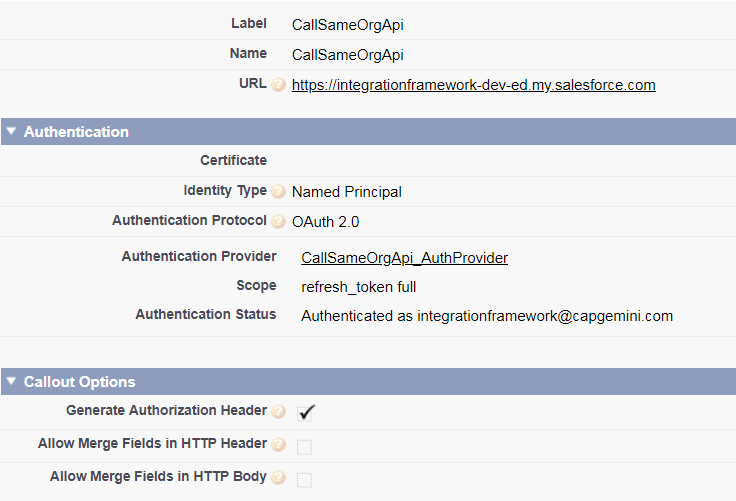


Copy the consumer key and secret from Connected App. Also copy the auto-generated callback URL and paste in Connected app. So, this would link the connected app and auth. Provider together.

**Named Credential:**

1. **CallSameOrgApi :-** This named credential will be used for authenticating the request when calling the REST api service of same Salesforce org. As part of framework this will be used to insert webservice log records via REST callout but it can be extended for making callouts to other REST services of same org.

Below is the screenshot for reference:



For the URL, enter the URL for your ORG but make sure it’s **\*\*\*. my.salesforce.com** i.e. domain name. Then clear the checkbox “Start Authentication Flow on Save” and click save.

Click Edit, make sure the checkbox is checked and save again.

The reason for not setting the checkbox the first time is that if you make some mistakes saving this you’ll have to restart the form.

**REST Framework Implementation Example**

A mock service has been used to test the working. Custom setting with name ‘GETTestAPI’ is created mentioning all details such as timeout value, endpoint URL, certificate name, request header etc. The JSON response obtained on hitting this service is de-serialized and stored as response body in ‘Webservice logs’ object for reference.

For testing error messages, an error API is being used in custom setting IntegrationSettings with name as ‘GETTestErrorAPI’ mentioning all details such as timeout value, endpoint url, certificate name, request header etc. The error codes and error messages will be stored in ‘Webservice logs’ object for reference when failure response is generated.

1. **MeetingStatusHelper: -** Helper class which holds methods for calling the GET and POST request of meeting status service. For every rest service, there should be an associated helper class. Methods included are as below: -

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Description** | **Input Parameter** | **Output Parameter** |
| getMeetingStatus | Returns the instance of CalloutResults with response for GET method on meeting service. Created the instance of *RESTIntegration* class and passed the integrating setting name ‘GETTestAPI’. Since the operation doesn’t require any input hence null is passed to callout method call. | null | Callout results for GET method |
| postMeetingStatus | Returns the instance of CalloutResults with response for post method for meeting service. Create instance of wrapper class *MeetingStatusInput* and set the values. Created the instance of *RESTIntegration* class and passed the integrating setting name ‘TestAPIPost’. Passed the instance of *MeetingStatusInput* to callout method call. | null | Callout results for POST method |

1. **MeetingStatusInput: -** This is a wrapper class used in *MeetingStatusHelper.* For every service if there are more than one input variable then create a wrapper class as per the request structure and pass it to callout method call.
2. **clsMeeting\_Struc: -** This is a wrapper class specific to service output format. For every service if there are more than one output variable then create a wrapper class as per the response structure and set the value of IntegrationSettings.OutputClass field with the wrapper class name so that framework will deserialize and typecast the response to instance of output wrapper class.

**Anonymous Code: -**

Http h = new Http();

// Instantiate a new HTTP request, specify the method (GET) as well as the endpoint

HttpRequest req = new HttpRequest();

req.setEndpoint('https://jlrcustomerportal-deved.my.salesforce.com/services/apexrest/LogStatusByRestCallout');

req.setMethod('POST');

req.setBody(jsonString);

// Send the request, and return a response

HttpResponse res = h.send(req);

-------------------------------------------------------------------------------------------

RESTIntegration rest = new RESTIntegration('GETTestAPI');

Object result = rest.callout(null);

System.debug('@@result : ' +result);

-----------------------------------------------------------------------------------------------

String userName = userInfo.getName();

String source = 'Salesforce';

MeetingStatusInput input = new MeetingStatusInput();

MeetingStatusInput.AuditDetails audit = new MeetingStatusInput.AuditDetails();

audit.name = 'Agent';

audit.value = userName+source;

List<MeetingStatusInput.AuditDetails> lstAudit = new List<MeetingStatusInput.AuditDetails>();

lstAudit.add(audit);

input.audit = lstAudit;

input.MeetingStatus = 'Completed';

RESTIntegration rest = new RESTIntegration('TestAPIPost');

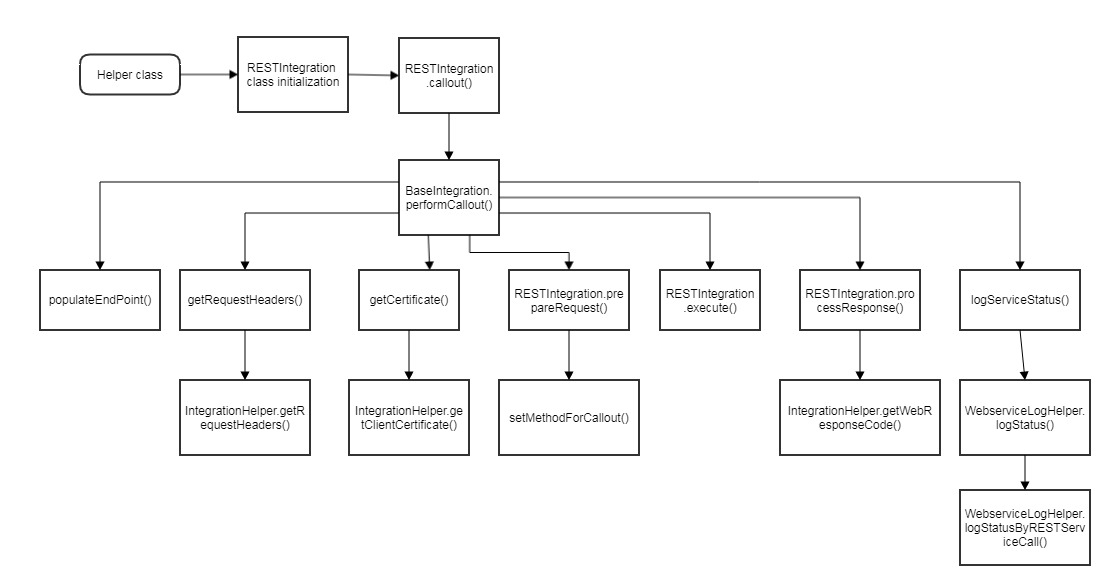
CallOutResults result = (CallOutResults)rest.callout(input);

System.debug('@@result : ' +result);

**Order of Execution: -**

|  |  |
| --- | --- |
| **Function** | **Classes** |
| getMeetingStatus | MeetingStatusHelper |
| Static Block of Service + Constructor | RESTIntegration |
| Callout | RESTIntegration |
| performCallout | BaseIntegration |
| populateEndPoint  getRequestHeaders -> (IntegrationHelper.getRequestHeaders)  getCertificate -> (IntegrationHelper.getClientCertificate) | BaseIntegration |
| prepareRequest - > setMethodForCallout | RESTIntegration |
| execute | RESTIntegration |
| processResponse -> (IntegrationHelper.getWebResponseCode) | RESTIntegration |
| logServiceStatus | BaseIntegration |
| logStatus | WebserviceLogHelper |
| logStatusByRESTServiceCall (If logging is by REST callout) | WebserviceLogHelper |
|  | |

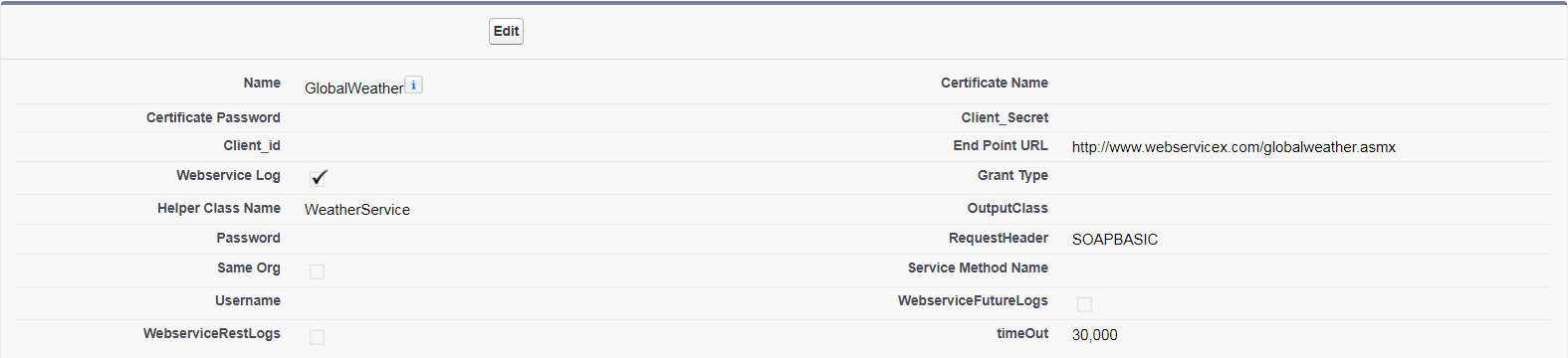
**Flow Diagram: -**

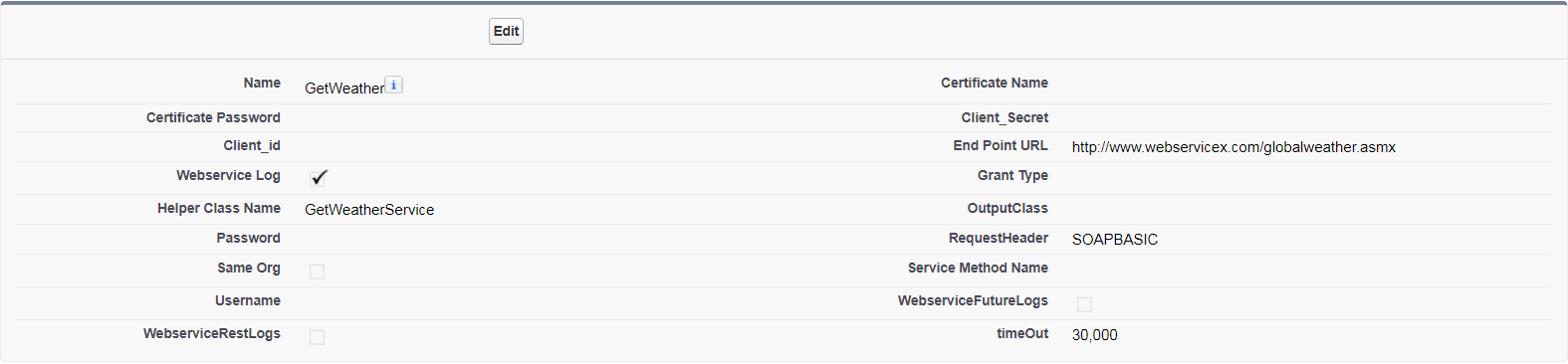
****

**SOAP FRAMEWORK**

**Custom settings:**

Similar to REST framework, same custom settings have been used. The screenshots are attached below for reference.





**Apex Classes (Framework Specific):**

1. **ServiceInterface: -** This is an interface class with three methods (prepareInputRequest, executeCallout, prepareOutputResponse). For every SOAP service implemented, there should be a service class implementing this interface.
2. **SOAPHeader: -** This is a wrapper class to pass endpoint, input/output http headers, client certificate name and call timeout values from framework to service class.
3. **SOAPIntegration: -** Like RestIntegration class, this class also extends *BaseIntegration* class and hence inherits all properties and methods in parent class. The main purpose of this class is for preparing request headers, calling service class prepareInputRequest, executeCallout & prepareOutputResponse methods. Methods included in this are as below: -

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Description** | **Input Parameter** | **Output Parameter** |
| callout | This method is used to perform callouts passing the request | Request as object instance. | Object |
| prepareRequest | Used for preparing request headers by setting the callout time, request headers, endpoint and client certificate name on SOAPHeader instance. It internally calls service class prepareInputRequest to prepare the request. | Null | null |
| execute | It internally calls service class *executeCallout* method passing instance of *SOAPHeader.* | null | null |
| processResponse | It internally calls service class prepareOutputResponse method for processing the response. Instance of CalloutResults is passed as input. | null | Null |
| setMethodForCallout | This method is blank. |  |  |

**SOAP Framework Implementation Example**

We have used a WSDL file of Global weather service which returns cities list and weather information based on country name and city name passed in input. The parsed WSDL class is *wwwWebservicexNet*.

**Apex Classes:**

1. **WeatherService: -** This class implements *ServiceInterface* class. It is used for preparing input request, making service callout and process output response.

For every SOAP service, we need to have a service class implementing the methods of interface.

Methods included here as below: -

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Description** | **Input Parameter** | **Output Parameter** |
| prepareInputRequest | Create instance of *wwwWebservicexNet. GetCitiesByCountry\_element* (wrapper class generated by parsing WSDL) and returns this instance. Map the values from service input to instance of the wrapper class. | Service input as object instance | Request as object instance |
| executeCallout | Create instance of *wwwWebservicexNet.GlobalWeatherSoap* class. Set the header values and call the execution method from WSDL generated class passing the input prepared in above method.  Return the response returned by service without any processing. | Instance of SOAPHeader class | Response as object instance |
| prepareOutputResponse | Check if status is success then Parse the response object and set it to a variable or instance of wrapper class based on response format.  We can add all the business logic for processing the response and map the response data to wrapper class instance.  Returns instance of variable or wrapper class with only required values. | Instance of CallOutResults class | Variable or instance of wrapper class as object instance |

1. **WeatherHelper: -** Helper class which holds methods for each service operation. For every SOAP service, there should be an associated helper class. Methods included are as below: -

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Description** | **Input Parameter** | **Output Parameter** |
| getCitiesByCountry | Returns the instance of CalloutResults with response of Weather service getCitiesByCountry method. Created the instance of *SOAPIntegration* class and passed the integrating setting name ‘GlobalWeather’. CountryName in a string variable is passed to callout method. | Country name as string | null |

1. **wwwWebservicexNet: -** WSDL generated class by parsing the WSDL file. Give the name of class specific to your service. Even if the service has multiple operations, the file has to be imported only once and used for implementing all operations.

**Anonymous Code: -**

string countryName = 'US';

SOAPIntegration soap = new SOAPIntegration('GlobalWeather');

CallOutResults result = (CallOutResults)soap.callout(countryName);

System.debug('@@result : ' +result);

------------------------------------------------------------------------

WeatherHelper.WeatherWrapper getWeatherWrap=new WeatherHelper.WeatherWrapper();

getWeatherWrap.countryName = 'US';

getWeatherWrap.CityName='Boston';

SOAPIntegration soap = new SOAPIntegration('GetWeather');

CallOutResults result = (CallOutResults)soap.callout(getWeatherWrap);

System.debug('@@result : ' +result);

**Order of Execution: -**

|  |  |
| --- | --- |
| **Function** | **Classes** |
| getCitiesByCountry | WeatherHelper |
| Static Block of Service + Constructor | SOAPIntegration |
| Callout | SOAPIntegration |
| performCallout | BaseIntegration |
| populateEndPoint  getRequestHeaders -> (IntegrationHelper.getRequestHeaders)  getCertificate -> (IntegrationHelper.getClientCertificate) | BaseIntegration |
| prepareRequest | SOAPIntegration |
| prepareInputRequest | ServiceClass (Service specific class implementing ServiceInterface) |
| execute | SOAPIntegration |
| executeCallout | ServiceClass (Service specific class implementing ServiceInterface) |
| GetCitiesByCountry | wwwWebservicexNet.GlobalWeatherSoap  (Service method in WSDL generated class) |
| processResponse | SOAPIntegration |
| prepareOutputResponse | ServiceClass (Service specific class implementing ServiceInterface) |
| logServiceStatus | BaseIntegration |
| logStatus | WebserviceLogHelper |
| logStatusByRESTServiceCall (If logging is by REST callout) | WebserviceLogHelper |
|  | |

**Flow Diagram: -**

