**Integration of salesforce org to another salesforce org**

**Steps :**

**1:** First of all we have to create a connected app in source salesforce org from where we have to export the data according to query.

**2:** Second Step is that we have to create a rest callout class and ase url mapping concept by which when we hit the end point in another org to get the data so this call would be called and send the systematic json formate data .

**We can initialize HTTP Methods :**  
POST         ——————>  **C**reate  
GET            ——————>  **R**ead  
PUT            ——————>  **U**psert  
DELETE     ——————>  **D**elete &  
PATCH       ——————>  Update

In the class this class must be global so that it can all from everywhere in whole application.

**3;** last step is that we will have to a simple class form generating the access token with the help of Client secret and Client id . Don’t be worried about that how will it because we will discuss and explain toward , here it is simple rough over view , after generating the access token we will hit the end point to access the (get,post,delete,put) one of them http method that is exist in source org and retrieve the data as response .

**Explation of all Steps :**

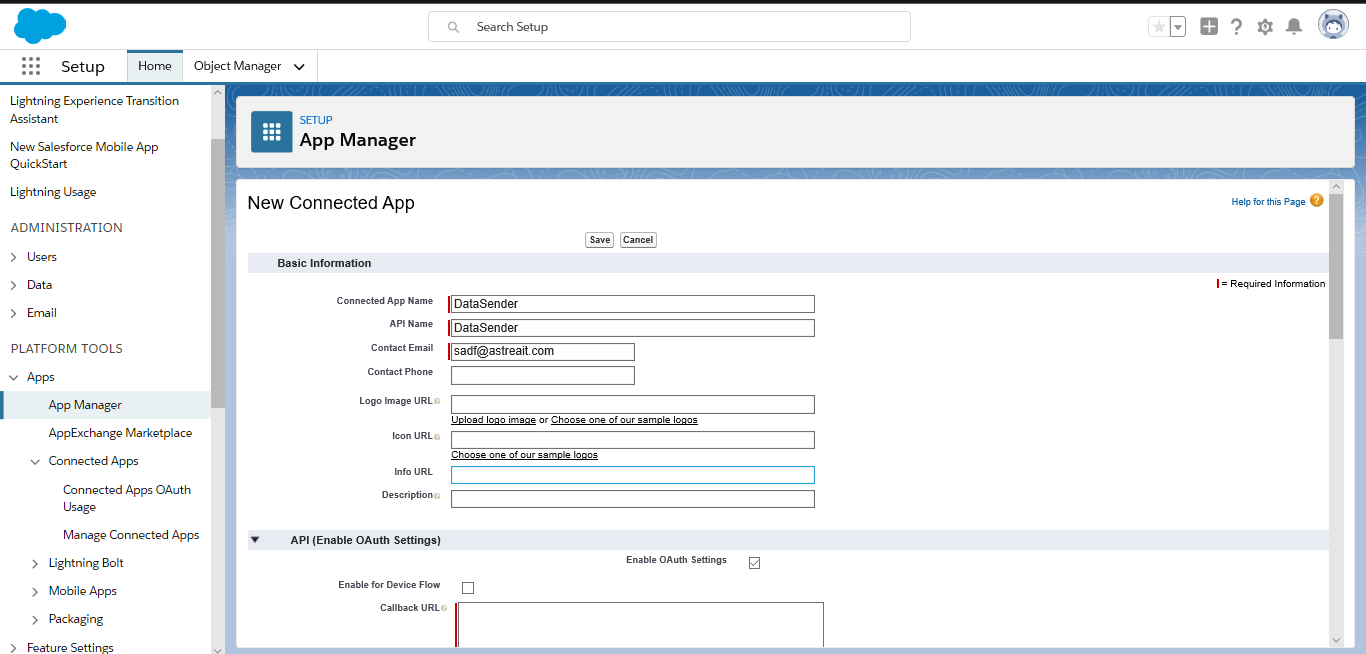
**Step 1: Create Connected App:**

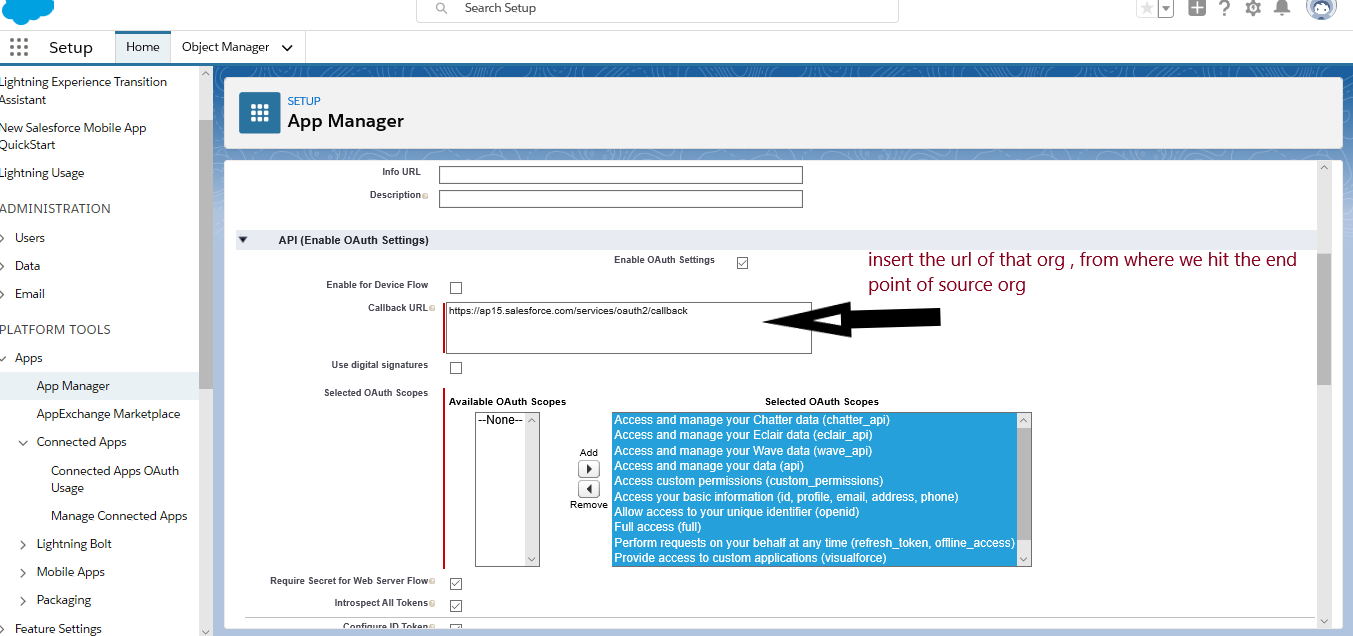
In Source Org Create one Connected App:

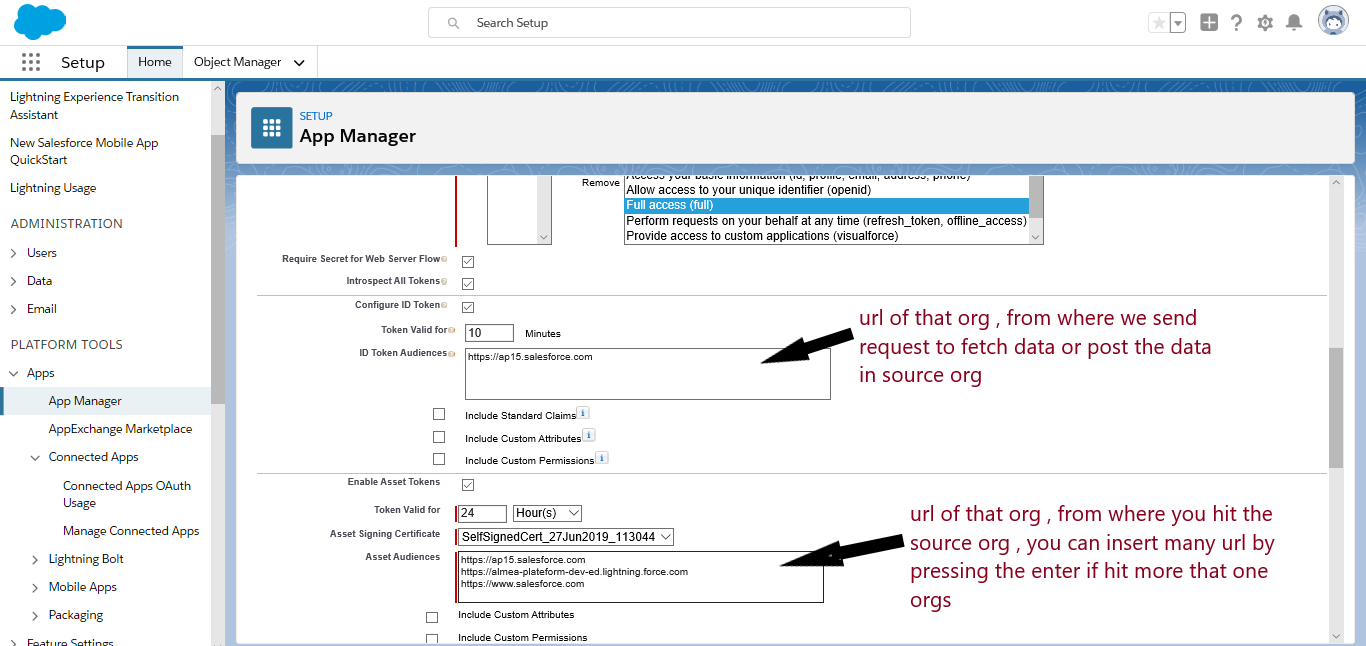
Login into Salesforce Sandbox or Salesforce Developer Org

then goto Setup-> App->Connected App

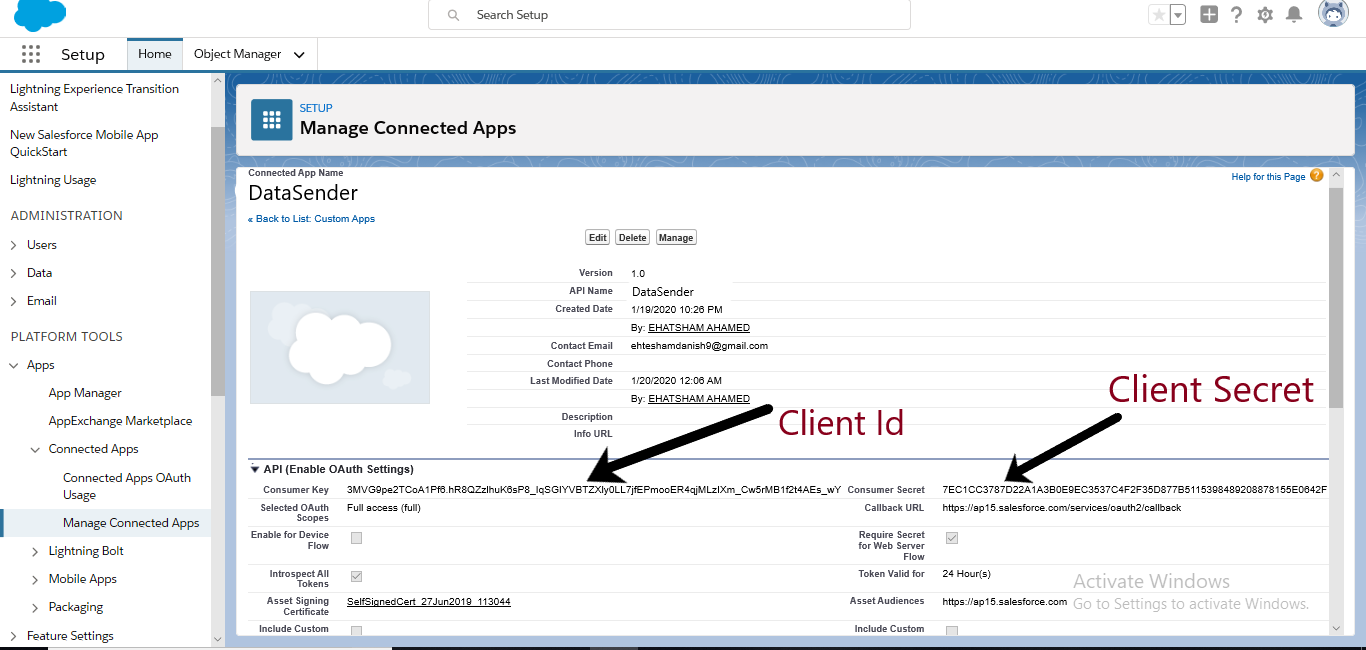
Click New







After pressing the save button you will see such type of screen from where you will get client secret and client Id:



Step2: Rest callout class Generation in source salesforce org :

Now we will create only two http method post and get in rest callout class you can . more :

@RestResource(urlMapping='/api/filteredDataSender/\*')

global with sharing class MyFirstRestApiClass

{

@HttpGet

global static map<string,list<sObject>> filteredDataSender(){

RestRequest req = RestContext.request;

RestResponse res = RestContext.response;

string NumberOfRecord = req.requestURI.substring(req.requestURI.lastIndexOf('/')+1);

system.debug('numberOfRecords is -->'+NumberOfRecord);

list<Account> accRec=new list<Account>();

String querys='Select Name,Id,AccountNumber,Industry,CreatedDate From Account Order By CreatedDate Desc Limit '+NumberOfRecord;

accRec=Database.query(querys);

system.debug('acount data is --->'+accRec);

list<Contact> ConRec =new list<Contact>();

string queryCon='Select Name,AccountId,Email,LeadSource,Phone,CreatedById,Department,OwnerId from Contact Order By CreatedDate Desc Limit '+NumberOfRecord;

ConRec=Database.query(queryCon);

system.debug('contact data '+conRec);

list<Opportunity> OppRec =new list<Opportunity>();

string queryOpp='Select Name,Amount,CloseDate,ExpectedRevenue,LeadSource,StageName,Type,AccountId,CreatedById,OwnerId from Opportunity Order By CreatedDate Desc Limit '+NumberOfRecord;

OppRec=Database.query(queryOpp);

system.debug('contact data '+OppRec);

if(accRec != Null && ConRec != Null && OppRec != Null)

{

Map<string,list<object>> mbData=new Map<string,list<object>>();

mbData.put(‘Account’,accRec);

mbData.put(‘Account’,conRec);

mbData.put(‘Account’,oppRec);

}

return mbData;

}

}

@HttpPost

global static string filteredDataSender(){

@HttpPost

global static MyUserDefinedClass echoMyType(string name)

{

//RestRequest req = RestContext.request;

//String Id = req.params.get('name');// for fetch the data this is second method ..

Contact c=new Contact();

c.LastName=name;

insert c;

return 'contact sucessfully created';

}

}

Step 3: Create a class to hit source org and generating the acess token :

public class AlmeaIntegrationData {

private final String clientId = '3MVG9pe2TCoA1Pf6.hR8QZzlhuK6sP8\_IqSGIYVBTZXly0LL7jfEPmooER4qjMLzIXm\_Cw5rMB1f2t4AEs\_wY';

private final String clientSecret = '7EC1CC3787D22A1A3B0E9EC3537C4F2F35D877B5115398489208878155E0642F';

private final String username = 'ehatsham.astrea@trailhead.com';

private final String password = '@htc19ALMEA';

public class deserializeResponse

{

public String id;

public String access\_token;

public String integrated\_Url;

public String token\_type;

}

public deserializeResponse token() {

string body='grant\_type=password&client\_id='+clientId+'&client\_secret='+clientSecret+'&username='+username+'&password='+password+'9wQNyQXWQYLiP4cwEG0jdvDqQ';//the high lighted content is security token for removing the ip trusted error .

Http httpRequest = new Http();

HttpRequest req = new HttpRequest();

String authorizationHeader = 'Bearer ' + '00D0o000001BjDr!AQsAQEAnOnssx5sifUSJ\_AUTRGKvs37aNFNmF8.Y7WjMgFnsonQw8JyCe.amY\_pyEeiWk9KnXsO7ExbCsqELlzvWKtEOyp1Q';

-🡪 (Bearer ) space must be after Bearer’\_’ at underscore position) in AuthorizationHeader:

system.debug(body);

req.setBody(body);

req.setMethod('POST');

req.setHeader('Authorization-Type','OAuth2.0');

// req.setHeader('redirect\_uri', 'https://almea-plateform-dev-ed.my.salesforce.com/services/oAuth2.0/token');

req.setHeader('Content-Type', 'application/x-www-form-urlencoded');

//for getting the access token you have to put application/x-www-form-urlencoded in Content-Type not 'application/json'

req.setEndpoint('https://login.salesforce.com/services/oauth2/token');

req.setHeader('accept','application/json');

HttpResponse rsp=httpRequest.send(req);

Map<String,Object> results = (Map<String, Object>) JSON.deserializeUntyped(rsp.getBody());

deserializeResponse newResData=new deserializeResponse();

newResData.access\_token=string.valueOf(results.get('access\_token'));

newResData.integrated\_Url=string.valueOf(results.get('instance\_url'));

newResData.id=string.valueOf(results.get('id'));

newResData.token\_type=string.valueOf(results.get('token\_type'));

system.debug('restonse data -->'+results);

system.debug('restonse data -->'+results.get('access\_token'));

return newResData;

}

public static void getDataSalesforce(){

AlmeaIntegrationData aid=new AlmeaIntegrationData();

deserializeResponse returnDataRest=new deserializeResponse();

returnDataRest =aid.token();

Http httpRequest=new Http();

HttpRequest rqt=new HttpRequest();

string strReq=string.valueOf(returnDataRest.access\_token).SubString(15);

rqt.setMethod('GET');

rqt.setHeader('Authorization', 'Bearer ' +returnDataRest.access\_token );

// rqt.setHeader('Signature', 'UU6hMsboO5qprstl1ZjjnId0IcuriG2jK9FBw4v62JQ=');

rqt.setHeader('Content-Type', 'application/json');

//rqt.setHeader('Content-Type','application/x-www-form-urlencoded');

rqt.setHeader('accept','application/json');

system.debug('access token '+string.valueOf(returnDataRest.access\_token).SubString(15));

// returnDataRest.integrated\_Url+

rqt.setEndpoint(returnDataRest.integrated\_Url+'/services/apexrest/api/filteredDataSender/200');

system.debug('url is -->'+returnDataRest.integrated\_Url+'/services/apexrest/api/filteredDataSender/200');

HttpResponse jkd = httpRequest.send(rqt);

System.debug(UserInfo.getOrganizationId()+' '+UserInfo.getSessionId().SubString(15));

// for showing the Token because salesforce does not show access token it always print session id is removed .

system.debug('example result-->'+jkd.getBody());

string er=jkd.getBody();

system.debug('body data '+er);

map<string,object> valueData=(map<string,object>)JSON.deserializeUntyped(jkd.getBody());

system.debug('real data --->'+valueData);

system.debug('Account --->'+ valueData.get('account'));

system.debug('Contact --->'+valueData.get('contact'));

system.debug('Opportunity --->'+valueData.get('opportunity'));

list<object> acc = new list<object>();

/\* for(object pd:(list<object>)valueData.get('account')){

acc.add(pd);

}\*/

for(object pd:(list<object>)valueData.get('contact')){

map<string,object> uber=(map<string,object>)pd;

acc.add(uber.get('Name'));

}

system.debug('account are that -->'+acc);

}

Public static void createAccountPost(){

AlmeaIntegrationData aid = new AlmeaIntegrationData();

deserializeResponse returnDataRes t= new deserializeResponse();

returnDataRest = aid.token();

Http httpRequest = new Http();

HttpRequest rqt = new HttpRequest();

String accessToken= returnDataRest.access\_token;

String endPoint = 'https:// returnDataRest. integrated\_Url /services/apexrest/api/filteredDataSender/';

Http h2 = new Http();

HttpRequest req1 = new HttpRequest();

req1.setHeader('Authorization','Bearer ' + accessToken);

req1.setHeader('Content-Type','application/json');

req1.setHeader('accept','application/json');

req1.setMethod('POST');

string name=’”name”:”Ehatsham”’;

req1.body(name);

req1.setEndpoint(endPoint);

HttpResponse res1 = h2.send(req1);

String trimmedResponse = res1.getBody().unescapeCsv().remove('\\');

system.debug('@@@RESPONSE@@'+trimmedResponse);

string parser=( string)JSON.deserializeUntyped(res1.getBody());

system.debug(parser);

}

}

**Exposing salesforce data through Apex REST API:-**

By using APEX REST API we can build a light weight API to access salesforce data through either XML or JSON communication medium from third party application.

APEX REST API supports CRUD (create, read, update and delete) operation through its GET,POST,DELETE,PUT,PATCH HTTP methods.

***HTTP GET METHOD :- Requests data from salesforce.***

***HTTP POST METHOD :- Submits data to be processed(creates a new record) into salesforce.***

***HTTP DELETE METHOD :- Performs deletion(Deleting a record) operation into salesforce.***

***HTTP PUT AND PATCH METHOD :-PUT and PATCH can be utilized to modify and update existing records in salesforce.***

**What is Apex REST endpoint ?**

You want your Java application should create a case record in salesforce through REST API, For that you need to build an REST API Endpoint URL in salesforce where java application can access that URL and can do CRUD(create, read, update and delete) operation.

**How to create Apex REST endpoint ?**

Before start writing Apex class, you need to annotate “@RestResource” and define the urlMapping by giving a name like below,

***@RestResource(urlMapping='/RESTCaseAccess/\*')***

Once you done with defining urlMapping, then your custom REST API url is ready for use and it looks like below

[**“https://YourserverInstance.salesforce.com/services/apexrest/*RESTCaseAccess/(your parameter)* “**](https://na8.salesforce.com/services/apexrest/FieldCase%E2%80%9D)

If you don’t have any parameter for hitting the url last one ‘/’ must be exist. Like

[**“https://YourserverInstance.salesforce.com/services/apexrest/*RESTCaseAccess/* “**](https://na8.salesforce.com/services/apexrest/FieldCase%E2%80%9D)

then the global Apex class can declare different methods depending on if the incoming request is an HTTP GET, POST, PUT or DELETE request.

**HTTP POST REST API Example :-**

***@RestResource(urlMapping='/RESTCaseAccess/\*')***

***global with sharing class RESTCreateCase {***

***@HttpPost***

***global static String createNewCase(String companyName, String caseType) {***

***List<Account> company = [Select ID, Name, BillingState from Account where Name =:companyName];***

***Case c = new Case();***

***c.OwnerId = '00541000000dxZM';***

***c.AccountId = company[0].Id;***

***c.Subject = caseType + ' for '+companyName;***

***c.Status = 'Open';***

***c.Type = caseType;***

***c.priority = 'High';***

***c.origin = 'Phone';***

***insert c;***

***return 'Case created ';***

***}***

***}***

Another example of http method as rest api integration for source org restResource class :

@HttpDelete

global static string deleteContact()

{

RestRequest req=RestContext.request;

String ids=req.params.get('Id');

List<Contact> c=[select id from contact where id=:ids];

delete c;

return 'contact deleted successfully';

}

@Httpput

global static string putContact(String ids,String Name)

{

Contact c=[select id from contact where id=:ids];

c.LastName=Name;

update c;

return 'contact updated successfully';

}

**Note :** Remember one thing that @httpget does not have ability to fetch the value from url or body from url

like ->

@HttpGet

global static List<Contact> getContact(string Name)

{

List<Contact> c=[select id from contact where name=:Name];

return c;

}

**Above mentioned example is wrong from the aspect of get the data from that one org which hit the source while the correct is below one:**

@HttpGet

global static string getContact()

{

RestRequest req = RestContext.request;

RestResponse res = RestContext.response;

string name = req.requestURI.substring(req.requestURI.lastIndexOf('/')+1);

// RestRequest req=RestContext.request;

// String nameStr=req.params.get('Id'); // for fetch the data from body

List<Contact> c=[select id from contact where name=: nameStr];

delete c;

return 'contact deleted successfully';

}

**Note :**  you can [create an after insert/update trigger](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_qs_trigger.htm) on the Lead object, and have the trigger call a [@future annotated method](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_classes_annotation_future.htm) which, in turn, calls a web service at the external system via an [HTTP callout](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_classes_restful_http.htm). Note that you cannot perform a callout directly from a trigger, only asynchronously via a future method.

**Note :** In session setting , we will have to some change to operate this ,

### Require secure connections (HTTPS)

|  |
| --- |
| Require secure connections (HTTPS)  https://almea-plateform-dev-ed.my.salesforce.com/img/s.gif |
| Require secure connections (HTTPS) for all third-party domains |
| Force relogin after Login-As-User |
| Require HttpOnly attribute |
| Use POST requests for cross-domain sessions |
| Enforce login IP ranges on every request  https://almea-plateform-dev-ed.my.salesforce.com/img/s.gif |

**Identity Verification**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | Information | These settings are also located in [Identity Verification](https://almea-plateform-dev-ed.my.salesforce.com/one/one.app#/setup/IdentityVerification/home). You can change these settings in either location. | |
| Enable the SMS method of identity verification  https://almea-plateform-dev-ed.my.salesforce.com/img/s.gif |
| Require security tokens for API logins from callouts (API version 31.0 and earlier) |
| Let users authenticate with a physical security key (U2F) |
| Let users authenticate with a certificate |
| Require identity verification during two-factor authentication (2FA) registration |
| Require identity verification for email address changes  https://almea-plateform-dev-ed.my.salesforce.com/img/s.gif |
| Require email confirmations for email address changes (applies to external users in Lightning Communities)  https://almea-plateform-dev-ed.my.salesforce.com/img/s.gif |
| Allow automated location-based verifications with Salesforce Authenticator |
| Allow only from trusted IP addresses |
|  |

**Note:** if you got 505 error in response so you will have to check first rest api class in source org please sure first after poassing the only one string as simple manner that it is working or not sometime rest callout class (of source org) create some conflict or error that is related to mapping and wrapper format in global systematic structure so please maintained it ..

**Note:** The only thing you need to do is retrieving body of the post request

@HttpPost

global static void processPost() {

Map<String, Object> requestBody = (Map<String, Object>) JSON.deserializeUntyped(RestContext.request.requestBody.toString());

//do something with the resulting map - this is the body of the POST request

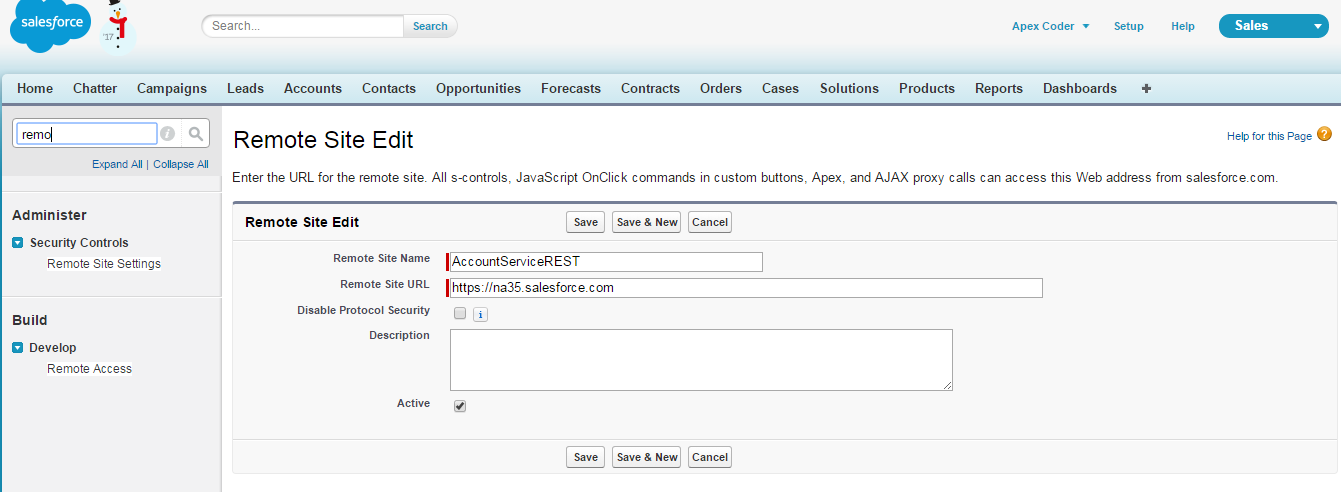
}

**Note: Invoke REST SERVICE from APEX :**

**Set source org’s url in Remote site setting in request sending salesforce org ;**

First you need to add the REST endpoint under Salesforce “Remote Site Settings”. To add the remote site settings, please follow the below steps:

1. From Setup, enter Remote Site Settings in the Quick Find box, then select Remote Site Settings.
2. Click New Remote Site.
3. Enter Remote Site Name.
4. Enter the URL for REST Service endpoint.
5. Click Save.



**Note :** if you are facing session time out or request time out error then set (Time out) in http request instance in your code like

HttpRequest req = new HttpRequest();

req.setEndpoint('https://na35.salesforce.com/services/apexrest/AccountService');

req.setMethod('GET');

req.setHeader('Authorization', 'OAuth ' + sessionId);

req.setHeader('Content-Type','**application/json**');

**req.setTimeout(120000);**

HttpResponse res = new HttpResponse();

res = http.send(req);

}