

# Integration HandBook



**WiseQuarter Education**



## Integration

Integration means to create a connection between a specific Salesforce instance and another database or system. The connection can be inbound, outbound, or bi-directional, and you may be connecting to another database, another Salesforce instance, or another cloud-based data source. Integrations can be focused on sharing data between systems, or continuing a business process automation from one system to another.

In other words; It is the integration of the Salesforce CRM with other clouds, apps on the Salesforce AppExchange and/or with 3rd party software.

### Api:

API stands for Application Programming Interface. It is the means that allows two applications to talk to each other. For example, when you use an app on your phone, the app connects to the internet, grabs data from a server, and presents it back to you in a readable format. With the right API, this whole process should happen without a hitch.

### Salesforce Integration Capabilities

You need a **high-functioning API** that allows any two applications to communicate with each other. When we are talking about integrations, it's important that we understand their timing and direction. Timing falls into two categories:

1. **Synchronous:** You make a call to another system, but you have to wait for a response in return. The processing won't continue until the response has been received.
2. **Asynchronous:** You make a call to another integrated system, but you don't have to wait for a response – further processing can continue. For example, if you have a background job that will take a long time to process, you don't want to wait for the response from that to be created (as you risk a timeout).

The other consideration is the direction of the integration, which can be outbound or inbound. Ask the question of which system is the initiator; if the answer is Salesforce then it's an outbound call, and if it's another system, it's an inbound call.

Integration capabilities are what Salesforce gives us in order to build these integrations. Here are two that you should know about:

## 1. REST API

This focuses on data-based operations:

- GET to query a database
- POST to create a record
- PUT to update
- PATCH
- DELETE

REST API is best for web or mobile applications. Regarding data formats, it uses **XML** or **JSON**. **JSON works better with data and is best for web or mobile applications.** This means that REST doesn't use much bandwidth and is easily consumed by web browsers.

In terms of timing, REST is synchronous. For example, you send a POST request to create a record in Salesforce and you will receive a response as to whether it was successful or not.

[Salesforce Workbench](#) is a suite of tools that allows you to interact with Salesforce using APIs. I recommend you do some exploring with Workbench to get familiar with the REST API.

## 2. SOAP API

SOAP API was more commonly used by older systems, but you may still come across it. It's best for system-to-system integrations, back-end system communication, and for applications that require formal hand-off (contracts) between the API and the consumer (thanks to [WSDL](#)).

While it's reliable and well-established, it tends to be **slower and use more bandwidth than REST**. It uses **XML** for very structured payloads and is asynchronous, meaning that it doesn't need to wait for a response and can continue with other processing without causing blocks.

### **XML:**

**XML is a software- and hardware-independent tool for storing and transporting data.**

- XML stands for extensible Markup Language
- XML is a markup language much like HTML



- XML was designed to store and transport data
- XML was designed to be self-descriptive

**Maybe it is a little hard to understand, but XML does not DO anything.**

- It has sender information
- It has receiver information
- It has a heading
- It has a message body

```
Note
To: Tove
From: Jani
Reminder
Don't forget me this
weekend!
```

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

**XML and HTML were designed with different goals:**

- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags are

**The XML language has no predefined tags:**

- The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.
  - HTML works with predefined tags like <p>, <h1>, <table>, etc.
  - With XML, the author must define both the tags and the document structure.
- XML simplifies data sharing
- XML simplifies data transport
- XML simplifies platform changes
- XML simplifies data availability

**Example:**

```
<?xml version="1.0" encoding="UTF-8"?>

<LightningComponentBundle
  xmlns="http://soap.sforce.com/2006/04/metadata">

  <apiVersion>55.0</apiVersion>

  <isExposed>true</isExposed>

  <targets>

    <target>lightning__AppPage</target>

    <target>lightning__HomePage</target>
```

**JSON:**

- JSON stands for **JavaScript Object Notation**
- JSON is a lightweight format for storing and transporting data
- JSON is often used when data is sent from a server to a web page
- JSON is "self-describing" and easy to understand



## Example:

it defines an employees object: an array of 3 employee records (objects):

```
{
  "employees": [
    {"firstName": "John", "lastName": "Doe"},
    {"firstName": "Anna", "lastName": "Smith"},
    {"firstName": "Peter", "lastName": "Jones"}
  ]
}
```

## Json Syntax:

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

The JSON syntax is derived from JavaScript object notation syntax, but the JSON format is text only. Code for reading and generating JSON data can be written in any programming language.

- **JSON data is written as name/value pairs, just like JavaScript object properties.**

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

```
"firstName": "John"
```

JSON names require double quotes. JavaScript names do not.

- **JSON objects are written inside curly braces.**

Just like in JavaScript, objects can contain multiple name/value pairs:

```
{"firstName": "John", "lastName": "Doe"}
```

- **JSON arrays are written inside square brackets.**

Just like in JavaScript, an array can contain objects:

```
"employees": [
  {"firstName": "John", "lastName": "Doe"},
  {"firstName": "Anna", "lastName": "Smith"},
  {"firstName": "Peter", "lastName": "Jones"}
]
```

In the example above, the object "employees" is an array. It contains three objects.

Each object is a record of a person (with a first name and a last name).

- **A common use of JSON is to read data from a web server, and display the data in a web page.**

For simplicity, this can be demonstrated using a string as input.

First, create a JavaScript string containing JSON syntax:

```
var text = '{ "employees" : [' +
'{"firstName":"John" , "lastName":"Doe" },' +
'{"firstName":"Anna" , "lastName":"Smith" },' +
'{"firstName":"Peter" , "lastName":"Jones" } ]}';
```

**console.log(text);**

```
// { "employees" : [{ "firstName":"John" , "lastName":"Doe" }, {"firstName":"Anna" , "lastName":"Smith" }, {"firstName":"Peter" , "lastName":"Jones" } ]}
```

Then, use the JavaScript built-in function `JSON.parse()` to **convert the string into a JavaScript object**:

```
var obj = JSON.parse(text);
console.log(obj);
```

```
/* {
  employees: [
```



## Apex Rest Method:

- Apex REST supports two formats for representations of resources: **JSON** and **XML**.
- JSON representations are passed by default in the body of a request or response, and the format is indicated by the Content-Type property in the HTTP header.
- You can retrieve the body as a Blob from the HttpRequest object if there are no parameters to the Apex method.
- If parameters are defined in the Apex method, an attempt is made to deserialize the request body into those parameters.
- If the Apex method has a non-void return type, the resource representation is serialized into the response body.

### These return and parameter types are allowed:

- Apex primitives (excluding sObject and Blob).
- sObjects
- Lists or maps of Apex primitives or sObjects (only maps with String keys are supported).

### Note:

- Apex REST doesn't support XML serialization and deserialization of Connect in Apex objects.
- Apex REST does support JSON serialization and deserialization of Connect in Apex objects.
- Also, some collection types, such as maps and lists, aren't supported with XML

### Use these annotations to expose an Apex class as a RESTful Web service:

- [@ReadOnly](#)
- [@RestResource](#)(urlMapping=' /yourUrl ')
- [@HttpDelete](#)
- [@HttpGet](#)
- [@HttpPatch](#)
- [@HttpPost](#)
- [@HttpPut](#)

- ❖ Methods annotated with @HttpGet or @HttpDelete should have no parameters. This is because GET and DELETE requests have no request body, so there's nothing to deserialize.
- ❖ The @ReadOnly annotation supports the Apex REST annotations for all the http requests: @HttpDelete, @HttpGet, @HttpPatch, @HttpPost, and @HttpPut.
- ❖ A single Apex class annotated with @RestResource can't have multiple methods annotated with the same HTTP request method. For example, the same class can't have two methods annotated with @HttpGet.

## Salesforce Apex REST API Key Methods

Apex code has several built-in methods that you can leverage when creating standard HTTP request-response protocols between a client and the Salesforce platform. These include standard methods such as **GET**, **POST**, **PUT**, and **DELETE**.

The REST callouts in Apex are associated with HTTP methods and endpoints. The HTTP method request that you callout will dictate the type of action that is desired for a given resource. You can call out these API methods to receive data from an external service or send data from Apex code to an external service.

REST guidelines recommend using a specific HTTP method on a particular type of call made to the server. These HTTP methods are as follows:

- GET
- POST



- PUT
- PATCH
- DELETE

## 1) GET

You can use the **HTTP GET method** to **read** (or retrieve) a resource representation. GET returns a representation in JSON or XML format in the safe path and an HTTP response code of **200 (OK)**. It most often returns a **400 (BAD REQUEST)** or **404 (NOT FOUND)** in an error case.

## 2) POST

The **POST verb** is most often used for **creating** new resources. In particular, developers use it to create subordinate resources. On successful creation, POST returns HTTP status **201**, returning a Location header with a link to the newly created resource.

## 3) PUT

PUT method can be used for **updating** the capabilities. However, you can also use PUT to **create** a resource where a client chooses the resource ID instead of the server.

## 4) PATCH

You can use the PATCH method to **modify** capabilities. The PATCH request only needs to contain the changes to the resource, not the complete resource. This resembles PUT, but the body has a set of instructions describing how you should modify a resource currently residing on the server to produce a new version.

## 5) DELETE

You can use the DELETE method to **delete** a resource identified by a URL.

## Further information:

From the above classification, you can see that the least difficult and most commonly used Apex REST API method that you can call out is a GET request. A GET request signifies that a client (sender) wants to retrieve data about a specific resource from the server. Once the server receives a GET request, it will process the request and return the requested information to the client.

You often use a lot of GET requests when browsing the internet. For example, let's say you want to access your Facebook account on your phone. You fire up your browser and click on the Facebook bookmark. Your browser then performs a GET request soliciting for Facebook's home page. The HTML page displayed on your screen is the response object.



Methods for HttpRequest	Methods for HttpResponse
<b>getBody()</b> Retrieves the body of this request.	<b>getBody()</b> Retrieves the body returned in the response.
<b>getBodyAsBlob()</b> Retrieves the body of this request as a Blob.	<b>getBodyAsBlob()</b> Retrieves the body returned in the response as a Blob.
<b>getBodyDocument()</b> Retrieves the body of this request as a DOM document.	<b>getBodyDocument()</b> Retrieves the body returned in the response as a DOM document.
<b>getCompressed()</b> If true, the request body is compressed, false otherwise.	<b>getHeader(key)</b> Retrieves the contents of the response header.
<b>getEndpoint()</b> Retrieves the URL for the endpoint of the external server for this request.	<b>getHeaderKeys()</b> Retrieves an array of header keys returned in the response.
<b>getHeader(key)</b> Retrieves the contents of the request header.	<b>getStatus()</b> Retrieves the status message returned for the response.
<b>getMethod()</b> Returns the type of method used by HttpRequest.	<b>getStatusCode()</b> Retrieves the value of the status code returned in the response.
<b>setBody(body)</b> Sets the contents of the body for this request.	<b>getXmlStreamReader()</b> Returns an XmlStreamReader that parses the body of the callout response.
<b>setBodyAsBlob(body)</b> Sets the contents of the body for this request using a Blob.	<b>setBody(body)</b> Specifies the body returned in the response.
<b>setBodyDocument(document)</b> Sets the contents of the body for this request. The contents represent a DOM document.	<b>setBodyAsBlob(body)</b> Specifies the body returned in the response using a Blob.
<b>setClientCertificate(clientCert, password)</b> This method is deprecated. Use setClientCertificateName instead.	<b>setHeader(key, value)</b> Specifies the contents of the response header.
<b>setClientCertificateName(certDevName)</b> If the external service requires a client certificate for authentication, set the certificate name.	<b>setStatus(status)</b> Specifies the status message returned in the response.
<b>setCompressed(flag)</b> If true, the data in the body is delivered to the endpoint in the gzip compressed format. If false, no compression format is used.	
<b>setEndpoint(endpoint)</b> Specifies the endpoint for this request.	
<b>setHeader(key, value)</b> Sets the contents of the request header.	
<b>setMethod(method)</b> Sets the type of method to be used for the HTTP request.	
<b>setTimeout(timeout)</b> Sets a timeout for the request between 1 and 120,000 milliseconds. The timeout is the maximum time to wait for establishing the HTTP connection. The same timeout is used for waiting for the request to start. When the request is executing, such as retrieving or posting data, the connection is kept alive until the request finishes.	
<b>toString()</b> Returns a string containing the URL for the endpoint of the external server for this request and the method used, for example, Endpoint=http://YourServer, Method=POST	



## A) Get Data from an External Service Using the Apex REST API

Salesforce developer console ile data çekme. (@HttpRequest)

- Let's write an Apex Class that sends a GET request
- The external web service will send back the response in JSON format. Since JSON is essentially a string, you will be using the built-in Apex **JSONParser** class to convert it to an object.

### EX:

Fetching All Customers. (GET) (From goRest to Salesforce Org) (By Name)

customerIntegration Class

```
public class customerIntegration {

    public static void fetchAllCustomers() {
        // todo http olustur
        Http http = new Http(); // postman acildi
        // todo http request olustur
        HttpRequest request = new HttpRequest();
        request.setMethod('GET');
        request.setEndpoint('https://gorest.co.in/public/v2/users');

        // gelen response isle
        HttpResponse response = http.send(request); // http.send(request); send butonuna basmak

        if(response.getStatusCode() == 200) {
            system.debug('Response getHeaderKeys() ' + response.getHeaderKeys());

            /*
            Response getHeaderKeys() (
            x-pagination-limit,
            x-pagination-page,
            Server,
            x-links-previous, vary, x-frame-options, x-download-options, x-links-next, Report-To,
            referrer-policy, ...)
            */
            system.debug('Response getHeader(Server) ::: ' + response.getHeader('Server'));
            // Response getHeader(Server) ::: cloudflare

            system.debug('Response getBody() ' + response.getBody());
            /*
            Response getBody() [
            { "id":5739,
            "name":"Ms. Darshwana Somayaji",
            "email":"ms_darshwana_somayaji@jakubowski.com",
            "gender":"female",
            "status":"inactive"},
            {"id":5737,"name":"Datta Embranthiri","email":"datta_embranthiri@wuckert-
            weissnat.io","gender":"female","status":"active"},
            {"id":5734,"name":"Gov...
            */
        } else {
            system.debug('Error :: ' + response.getStatusCode());
        }
    }
}
```

Anonymous Windows

```
customerIntegration.fetchAllCustomers();
```

ExecuteAnonymous Error

System.CalloutException: Unauthorized endpoint, please check Setup->Security->Remote site settings. endpoint = https://gorest.co.in/public/v2/users





After this error be solved, we may execute.

From SetUp --> Home --> Remote Site Settings --> New Remote Site

**EX:**

Search Customer By userId (**GET**) (From goRest to Salesforce Org)

customerIntegration Class

```
public class customerIntegration {

    public static void searchCustomerById(String userId){
        // todo http olustur
        Http http = new Http(); // postman acildi
        // todo http request olustur
        HttpRequest request = new HttpRequest();
        request.setMethod('GET');
        request.setEndpoint('https://gorest.co.in/public/v2/users/' + userId);

        // gelen response isle
        HttpResponse response = http.send(request); // http.send(request); send butonuna basmak

        if(response.getStatusCode() == 200) {
            system.debug('Response getHeaderKeys() ' + response.getHeaderKeys());

            /*
            Response getHeaderKeys() (
            x-pagination-limit,
            x-pagination-page,
            Server,
            x-links-previous, vary, x-frame-options, x-download-options, x-links-next, Report-To,
            referrer-policy, ...)
            */
            system.debug('Response getHeader(Server) :::: ' + response.getHeader('Server'));

            // Response getHeader(Server) :::: cloudflare

            system.debug('Response getBody() ' + response.getBody());
            /*
            Response getBody() {
            "id":5614,"name":"Navin Pillai",
            "email":"pillai_navin@reichel-labadie.biz",
            "gender":"female",
            "status":"inactive"}
            */
        }else {
            system.debug('Error :: ' + response.getStatusCode());
        }
    }
}
```

Anonymous Windows

```
customerIntegration.searchCustomerById ('5732');
```



EX:

Search Customer By Name (GET) (From goRest to Salesforce Org) (By Name)

customerIntegration Class

```
public class customerIntegration {

    public static void searchCustomerByName(String Name){
        // todo http olustur
        Http http = new Http(); // postman acildi
        // todo http request olustur
        HttpRequest request = new HttpRequest();
        request.setMethod('GET');
        request.setEndpoint('https://goREST.co.in/public/v2/users?name=' + Name);

        // gelen response isle
        HttpResponse response = http.send(request); // http.send(request); send butonuna basmak

        if(response.getStatusCode() == 200) {
            system.debug('Response getHeaderKeys() ' + response.getHeaderKeys());

            /*
            Response getHeaderKeys() (
            x-pagination-limit,
            x-pagination-page,
            Server,
            x-links-previous, vary, x-frame-options, x-download-options, x-links-next, Report-To,
            referrer-policy, ...)
            */
            system.debug('Response getHeader(Server) ::: ' + response.getHeader('Server'));

            // Response getHeader(Server) ::: cloudflare

            system.debug('Response getBody() ' + response.getBody());
            /*
            Response getBody() [
            {
            "id":5468,
            "name":"Akroor Bhattathiri",
            "email":"akroor_bhattathiri@pagac-kautzer.name",
            "gender":"female",
            "status":"inactive"
            },
            {
            "id":5406,
            "name":"Akroor Khanna",
            "email":"khanna_akroor@ryan.net",
            "gender":"female",
            "status":"active"
            },
            {
            "id":4634,
            "name":"Akroor Kaul", "e
            */
        }else {
            system.debug('Error :: ' + response.getStatusCode());
        }
    }
}
```



```
customerIntegration.searchCustomerByName ('Akroor');
```

## customerIntegration Class

```
public class customerIntegration {  
  
    public static final String API_ENDPOINT = 'https://gorest.co.in/public/v2/users'  
  
    public static void searchCustomerByName(String Name){  
        // todo http olustur  
        Http http = new Http(); // postman acildi  
        // todo http request olustur  
        HttpRequest request = new HttpRequest();  
        request.setMethod('GET');  
        request.setEndpoint(API_ENDPOINT+'?name=' + Name);  
  
        // gelen response isle  
        HttpResponse response = http.send(request); // http.send(request); send butonuna basmak  
  
        if(response.getStatusCode() == 200) {  
            system.debug('Response getHeaderKeys() ' + response.getHeaderKeys());  
  
            /*  
            Response getHeaderKeys() (  
            x-pagination-limit,  
            x-pagination-page,  
            Server,  
            x-links-previous, vary, x-frame-options, x-download-options, x-links-next, Report-To,  
            referrer-policy, ...)  
            */  
            system.debug('Response getHeader(Server) :::: ' + response.getHeader('Server'));  
  
            // Response getHeader(Server) :::: cloudflare  
  
            system.debug('Response getBody() ' + response.getBody());  
            /*  
            Response getBody() [  
            {  
            "id":5468,  
            "name":"Akroor Bhattathiri",  
            "email":"akroor_bhattathiri@pagac-kautzer.name",  
            "gender":"female",  
            "status":"inactive"  
            },  
            {  
            "id":5406,  
            "name":"Akroor Khanna",  
            "email":"khanna_akroor@ryan.net",  
            "gender":"female",  
            "status":"active"  
            },  
            {  
            "id":4634,  
            "name":"Akroor Kaul", "e  
            */  
        }else {  
            system.debug('Error :: ' + response.getStatusCode());  
        }  
    }  
}
```



## Anonymous Windows

```
customerIntegration.searchCustomerByName ('Akroor');
```

## Serialization and Deserialization

Use the JSON class methods to perform roundtrip serialization and deserialization of your JSON content. These methods enable you to serialize objects into JSON-formatted strings and to deserialize JSON strings back into objects.

**Serialization** is a process of converting an apex object into stream of bytes so that it can be transferred over a network or stored in a salesforce record. (From sObj to JSON). (Objecti JSON formatina çevirme)

## Anonymous Windows

```
Account acc = [SELECT Id, Name, industry FROM Account WHERE Industry != null Limit 1];
System.debug('acc: ' + acc);
// acc: Account:{Id=0018d00000GTLh1AAH, Name=WR Email Deneme, Industry=Banking}

String jsonstr = JSON.serialize(acc); // From sObject To JSON String
System.debug('JSON.serialize(acc): ' + jsonstr);
/*
JSON.serialize(acc):
{"attributes":{"type":"Account","url":"/services/data/v56.0/subjects/Account/0018d00000GTLh1AAH"},
"Id":"0018d00000GTLh1AAH",
"Name":"WR Email Deneme",
"Industry":"Banking"}
*/

List<Account> accList = [SELECT Id, Name, industry FROM Account WHERE Industry != null Limit 3];
System.debug('accList: ' + accList);
/*
accList: (Account:{Id=0018d00000GTLh1AAH, Name=WR Email Deneme, Industry=Banking},
Account:{Id=0018d00000GTLq4AAH, Name=Time Based Deneme, Industry=Banking},
Account:{Id=0018d00000Bks37AAB, Name=GenePoint, Industry=Biotechnology})
*/
String jsonstr = JSON.serialize(accList);
System.debug('JSON.serialize(accList): ' + jsonstr);
/*
JSON.serialize(accList): [
{"attributes":{"type":"Account","url":"/services/data/v56.0/subjects/Account/0018d00000GTLh1AAH"},
"Id":"0018d00000GTLh1AAH","Name":"WR Email Deneme","Industry":"Banking"},
{"attributes":{"type":"Account","url":"/services/data/v56.0/subjects/Account/0018d00000GTLq4AAH"},
"Id":"0018d00000GTLq4AAH",&...
*/
```



**Deserialization** is a process of converting a JSON into an apex object. Deserialization is the exact opposite of Serialization – which convert bytes of stream into object. (JSON formatını sObject e çevirme)

## Anonymous Windows

```
Account acc = [SELECT Id, Name, industry FROM Account WHERE Industry != null Limit 1];
```

```
String jsonstr = JSON.serialize(acc); // From sObject To JSON String
```

```
Account jsonAcc = (Account)JSON.deserialize(jsonStr, Account.class); // From JSON to sObj.
```

```
System.debug('JSON.deserialize(jsonStr, Account.class): ' + jsonAcc);
```

```
/*  
JSON.deserialize(jsonStr, Account.class): Account:{Industry=Banking, Id=0018d00000GTLh1AAH,  
Name=WR Email Deneme}  
*/
```

```
List<Account> accList = [SELECT Id, Name, industry FROM Account WHERE Industry != null  
Limit 3];
```

```
String jsonstr = JSON.serialize(accList); // From sObject List To JSON Array
```

```
List<Account> jsonAccList = (List<Account>)JSON.deserialize(jsonStr, List<Account>.class);  
// From JSON Array to sObject List.
```

```
System.debug('List<Account>)JSON.deserialize(jsonStr, List<Account>.class): ' + jsonAccList);
```

```
/*  
JSON.deserialize(jsonStr, Account.class):  
(  
Account:{Industry=Banking, Id=0018d00000GTLh1AAH, Name=WR Email Deneme},  
Account:{Industry=Banking, Id=0018d00000GTLq4AAH, Name=Time Based Deneme},  
Account:{Industry=Biotechnology, Id=0018d00000Bks37AAB, Name=GenePoint}  
)  
*/
```



## EX: (GET) (From goRest to Salesforce Org)

Create a **Customer** Object in Salesforce Org. and create fields: **name, email, gender, status**.  
Gelen JSON **Deserialize** edilecek ve yeni objecti Database e insert edeceğiz. (For All)  
(geçici bir object-class oluşturarak-TempObj class) çünkü gelen JSON da da Id olduğu için.

### customerIntegration Class

```
public class customerIntegration {
    public static final String API_ENDPOINT = 'https://gorest.co.in/public/v2/users'

    public static void fetchAndInsertCustomers( ){

        Http http = new Http();

        HttpRequest request = new HttpRequest();
        request.setMethod('GET');
        request.setEndpoint(API_ENDPOINT);

        HttpResponse response = http.send(request);

        if(response.getStatusCode() == 200) {
            system.debug('Response  getBody() ' + response.getBody() );

            List<TempObj> jsonTmpList = (List<TempObj>)JSON.deserialize(response.getBody(), List<TempObj>.class);
            // From JSON Array to sObject List.

            System.debug('List<TempObj>)JSON.deserialize(jsonStr, List<TempObj>.class): ' + jsonTmpList);

            List<Customer__c> jsonCstList = new List<Customer__c>();

            for(TempObj tmp: jsonTmpList){
                Customer__c newCustomer = new Customer__c();
                newCustomer.name = tmp.name;
                newCustomer.email__c = tmp.email;
                newCustomer.gender__c = tmp.gender;
                newCustomer.status__c = tmp.name;
                jsonCstList.add(newCustomer);
            }
            Database.insert(jsonCstList);
        }else {
            system.debug('Error :: ' + response.getStatusCode() );
        }
    }
}
```

### Anonymous Windows

```
customerIntegration.fetchAndInsertCustomers();
```

### TempObj Class

```
public class TempObj {
    public integer id;
    public string name;
    public string email;
    public string gender;
    public string status;
}
```



## EX: (GET) (From goRest to Salesforce Org)

(For Single sObject Records)

### customerIntegration Class

```
public class customerIntegration {
    public static final String API_ENDPOINT = 'https://gorest.co.in/public/v2/users'

    public static void fetchAndInsertSingleCustomers( ){

        Http http = new Http();

        HttpRequest request = new HttpRequest();
        request.setMethod('GET');
        request.setEndpoint(API_ENDPOINT+'/'+userId);

        HttpResponse response = http.send(request);

        if(response.getStatusCode() == 200) {
            system.debug('Response  getBody() ' + response.getBody() );

            TempObj jsonTmp = (TempObj)JSON.deserialize(response.getBody(), TempObj.class);
            // From JSON Array to sObject List.

            System.debug('List<TempObj>)JSON.deserialize(jsonStr, List<TempObj>.class): ' + jsonTmp);

            List<Customer__c> jsonCstList = new List<Customer__c>();

            Customer__c newCusttomer = new Customer__c();
            newCusttomer.name = jsonTmp.name;
            newCusttomer.email__c = jsonTmp.email;
            newCusttomer.gender__c = jsonTmp.gender;
            newCusttomer.status__c = jsonTmp.status;
            jsonCstList.add(newCusttomer);

            Database.insert(jsonCstList);

        }
    }
}
```

### Anonymous Windows

```
customerIntegration.fetchAndInsertSingleCustomers('5634');
```

### TempObj Class

```
public class TempObj {
    public integer id;
    public string name;
    public string email;
    public string gender;
    public string status;
}
```







## B) Send Data to an External Service Using the Apex REST API

HTTP POST request to send data to an external web service. The data packet again will be in JSON format.

**EX:** Sending data as a hard code (**POST**) (From Developer Console to goRest)

customerIntegration Class

```
public class customerIntegration {
    public static final String API_ENDPOINT = 'https://gorest.co.in/public/v2/users'

    public static void sendUserToGoRest(){

        Http http = new Http();
        String accessToken = '4d5904f8cd60978f0f1e5a297260b37614a35f3da1e397a55dbe8ea448c47779';

        HttpRequest request = new HttpRequest();
        request.setMethod('POST');

        // 1. yöntem
        request.setEndpoint(API_ENDPOINT);
        request.setHeader('Authorization', 'Bearer ' + accessToken);

        // 2. yöntem (url de görünür-tavsiye değil)
        // request.setEndpoint(API_ENDPOINT+'?access-token='+accessToken);

        request.setHeader('Content-Type', 'application/json;charset=UTF-8');

        String userForGoRest = '{"name": "Micheal", "email": "micheal46@micheal.com", "gender": "male",
                                "status": "active"}'; //JSON format

        request.setBody(userForGoRest);

        HttpResponse response = http.send(request);

        if(response.getStatusCode() == 201) {

            system.debug('Response getHeader(Content-Type) :::: ' + response.getHeader('Content-Type') );
            // Response getHeader(Content-Type) :::: application/json; charset=utf-8

            system.debug('Response getBody() ' + response.getBody() );
            // Response getBody() {"id":8040,"name":"Micheal", "email":"micheal46@micheal.com", "gender":"male", "status":"active"}

        }else {
            system.debug('Error :: ' + response.getStatusCode() );
        }
    }
}
```

Anonymous Windows

```
customerIntegration.sendUserToGoRest();
```



**EX:** Sending data of a customer obj. by id as a half dynamic (**POST**) (From Salesforce Org to goRest)

customerIntegration Class

```
public class customerIntegration {
    public static final String API_ENDPOINT = 'https://gorest.co.in/public/v2/users'

    public static void sendUserDynToGoRest(){

        Http http = new Http();
        String accessToken = '4d5904f8cd60978f0f1e5a297260b37614a35f3da1e397a55dbe8ea448c47779';

        HttpRequest request = new HttpRequest();
        request.setMethod('POST');

        request.setEndpoint(API_ENDPOINT);
        request.setHeader('Authorization', 'Bearer ' + accessToken);
        request.setHeader('Content-Type', 'application/json;charset=UTF-8');

        Customer__c sendingCustomer = [SELECT Id, Name, email__c, gender__c, status__c FROM Customer__c
                                       WHERE id='a0G8d000007X1lVEAS'];

        String jsonCst = JSON.serialize(sendingCustomer); // convert into JSON format to send

        request.setBody(jsonCst);

        HttpResponse response = http.send(request);

        if(response.getStatusCode() == 201) {

            system.debug('Response getHeader(Content-Type) :::: ' + response.getHeader('Content-Type') );
            // Response getHeader(Content-Type) :::: application/json; charset=utf-8

            system.debug('Response getBody() ' + response.getBody() );
            // Response getBody() {"id":8040,"name":"Micheal", "email":"micheal46@micheal.com", "gender":"male","status":"active"}

        }else {
            system.debug('Error :: ' + response.getStatusCode() );
        }
    }
}
```

Anonymous Windows

```
customerIntegration.sendUserDynToGoRest ();
```

Log executeAnonymous

## Error :: 422

goRest defines **own id**. And since we sent it with id, it conflicted and gave an error.

We need to send data **without ID**.

### Solution:

We create a temporary object class. We temporarily assign the records from the Salesforce database there.

Then we serialize it from the temporary object(class) and send it without ID



## Correct Answer:

customerIntegration Class

```
public class customerIntegration {
    public static final String API_ENDPOINT = 'https://gorest.co.in/public/v2/users'

    public static void sendUserDynToGoRest(){

        Http http = new Http();
        string accessToken = '4d5904f8cd60978f0f1e5a297260b37614a35f3da1e397a55dbe8ea448c47779';

        HttpRequest request = new HttpRequest();
        request.setMethod('POST');

        request.setEndpoint(API_ENDPOINT);
        request.setHeader('Authorization', 'Bearer ' + accessToken);
        request.setHeader('Content-Type', 'application/json;charset=UTF-8');

        Customer__c sendingCust = [SELECT Id, Name, email__c, gender__c, status__c FROM Customer__c
                                   WHERE id='a0G8d000007X1laEAC'];

        {
            SendingUser SendObj = new SendingUser();

            {
                SendObj.Name = sendingCust.Name;
                SendObj.email = sendingCust.email__c;
                SendObj.gender = sendingCust.gender__c;
                SendObj.status = sendingCust.status__c;
            }
        }

        String jsonCst = JSON.serialize(SendObj);    // convert into JSON format to send

        request.setBody(jsonCst);

        HttpResponse response = http.send(request);

        if(response.getStatusCode() == 201) {

            system.debug('Response getHeader(Content-Type) :::: ' + response.getHeader('Content-Type') );
            // Response getHeader(Content-Type) :::: application/json; charset=utf-8

            system.debug('Response getBody() ' + response.getBody() );
            // Response getBody(){ "id":8963,"name":"ilker","email":"trivedi_harinarayan@koss.biz","gender":"male","status":"active"}

        }else {
            system.debug('Error :: ' + response.getStatusCode() );
        }
    }
}
```

Anonymous Windows

```
customerIntegration.sendUserDynToGoRest ();
```

SendingUser Class

```
public class SendingUser{
    public string name;
    public string email;
    public string gender;
    public string status;
}
```



**EX:** Sending object From Anonymous Windows to goRest by getting data from user. (POST)  
(From Salesforce Org to goRest)

## customerIntegration Class

```
public class customerIntegration {
    public static final String API_ENDPOINT = 'https://gorest.co.in/public/v2/users'

    public static void sendToGoRest(String userName, String userEmail, String userGender,
String userStatus ){

        Http http = new Http();
        string accessToken = '4d5904f8cd60978f0f1e5a297260b37614a35f3da1e397a55dbe8ea448c47779';

        HttpRequest request = new HttpRequest();
        request.setMethod('POST');
        request.setEndpoint(API_ENDPOINT);
        request.setHeader('Authorization', 'Bearer ' + accessToken);
        request.setHeader('Content-Type', 'application/json;charset=UTF-8');

        SendingUser SendObj = new SendingUser();

        SendObj.Name = userName;
        SendObj.email = userEmail;
        SendObj.gender = userGender;
        SendObj.status = userStatus;

        String jsonObj = JSON.serialize(SendObj); // convert into JSON format to send

        request.setBody(jsonObj);

        HttpResponse response = http.send(request);

        if(response.getStatusCode() == 201) {

            system.debug('Response getHeader(Content-Type) :::: ' + response.getHeader('Content-Type') );
            // Response getHeader(Content-Type) :::: application/json; charset=utf-8

            system.debug('Response getBody() ' + response.getBody() );
            // Response getBody() {"id":9033,"name":"ilker","email":"ilker46@ilker.com","gender":"male","status":"active"}

        }else {
            system.debug('Error :: ' + response.getStatusCode() );
        }
    }
}
```

## Anonymous Windows

```
customerIntegration.sendUserToGoRest ();
```

## SendingUser Class

```
public class SendingUser{
    public string name;
    public string email;
    public string gender;
    public string status;
}
```



## EX:

When **insertToGoRest\_\_c** field checkbox is true, Send data of a customer object **By using trigger** (first create a insertToGoRest checkbox field for Customer\_\_c object) **(POST)** **(From Salesforce Org to goRest)**

### customerIntegration Class

```
public class customerIntegration {
    public static final String API_ENDPOINT = 'https://goREST.co.in/public/v2/users'

    public static void sendUserToGoRestByTrigger(Id mustId){

        Http http = new Http();
        string accessToken = '4d5904f8cd60978f0f1e5a297260b37614a35f3da1e397a55dbe8ea448c47779';

        HttpRequest request = new HttpRequest();

        request.setMethod('POST');
        request.setEndpoint(API_ENDPOINT);
        request.setHeader('Authorization', 'Bearer ' + accessToken);
        request.setHeader('Content-Type', 'application/json;charset=UTF-8');

        Customer__c sendingCust = [SELECT Id, Name, email__c, gender__c, status__c FROM Customer__c
                                   WHERE Id=:mustId];

        SendingUser SendObj = new SendingUser();

        SendObj.Name = sendingCust.Name;
        SendObj.email = sendingCust.email__c;
        SendObj.gender = sendingCust.gender__c;
        SendObj.status = sendingCust.status__c;

        String jsonCst = JSON.serialize(SendObj);    // convert into JSON format to send

        request.setBody(jsonCst);

        HttpResponse response = http.send(request);

        if(response.getStatusCode() == 201) {

            system.debug('Response getHeader(Content-Type) :::: ' + response.getHeader('Content-Type') );
            // Response getHeader(Content-Type) :::: application/json; charset=utf-8

            system.debug('Response getBody() ' + response.getBody() );
            // Response getBody() {"id":9282,"name":"ilkerT","email":"ilker@trigger.com","gender":"male","status":"active"}

        }else {
            system.debug('Error :: ' + response.getStatusCode() );
        }
    }
}
```

### insertCustomerToGoRestTrigger

```
trigger insertCustomerToGoRestTrigger on Customer__c
(after insert, after update) {

    for(Customer__c cst: Trigger.new){
```

### SendingUser Class

```
public class SendingUser{
    public string name;
    public string email;
    public string gender;
    public string status;
```



## Http Request From Client(Postman) To Salesforce

### Two ways of learning my domain url:

1. From salesforce set up, go home and write 'My Domain' in the Quick Find Box and click on it.
2. In developer console; open Anonymous Windows and execute the following code:

```
System.debug(System.url.getSalesforceBaseUrl());
```

To reach our object CLIENT should add **/services/apexrest/...** after domain  
**xxxxsalesforce.com/services/apexrest/cases** (for Case Obj.)

**SessionID:** A session Id is used to identify a user using salesforce UI or API tools, it has a time limit and can be manually expired by the user logging out or by an admin removing that session in setup.

To learn sessionID, Execute the following code in Anonymous Windows:

```
System.debug('SessionId:: ' + System.UserInfo.getSessionId().substring(15));
```

Or Go To **Setup** | Type **Session Management** in the Quick Find box | Click **Session Management**

### @HttpGet :

The **@HttpGet** annotation is used at the method level and enables you to expose an Apex method as a REST resource. This method is called when an HTTP GET request is sent, and returns the specified resource.

These are some considerations when using this annotation:

- To use this annotation, your Apex method must be defined as **global static**.
- Methods annotated with @HttpGet are also called if the HTTP request uses the HEAD request method.

### @HttpDelete :

The **@HttpDelete** annotation is used at the method level and enables you to expose an Apex method as a REST resource. This method is called when an HTTP DELETE request is sent, and deletes the specified resource.

To use this annotation, your Apex method must be defined as global static.

### @HttpPost :

The **@HttpPost** annotation is used at the method level and enables you to expose an Apex method as a REST resource. This method is called when an HTTP POST request is sent, and creates a new resource.

To use this annotation, your Apex method must be defined as global static.

### @HttpPut :

The **@HttpPut** annotation is used at the method level and enables you to expose an Apex method as a REST resource. This method is called when an HTTP PUT request is sent, and creates or updates the specified resource.

To use this annotation, your Apex method must be defined as global static.

### @HttpPatch :

The **@HttpPatch** annotation is used at the method level and enables you to expose an Apex method as a REST resource. This method is called when an HTTP PATCH request is sent, and updates the specified resource.

To use this annotation, your Apex method must be defined as global static.

\*\*\* **@HttpGet, @HttpDelete, @HttpPost, @HttpPut ve @HttpPatch** methodlarını içeren class oluşturduktan sonra **Postman** da Authorization kısmına **sessionId** ekleyerek **Get, Delete, Post, Put ve Patch** yapabilirsiniz.  
(Client = Postman)



```
Map<String, Object> params = (Map<String, Object>)JSON.deserializeUntyped(req.requestBody.toString());
```

**deserializeUntyped** Gelen request in Body sini string e çevirir.

Gelen Request in hangi formatta geldiğini bilmediğimiz için **deserializeUntyped()** ve Map kullandık. Map içerisinde string ve object çifti olarak veriler tutuluyor. Bu verileri for ile dönerek istediğimiz bilgileri alırız.

```
@RestResource(urlMapping='/yourURLNameHere/')
```

is used to allow a third party system to hit this URL. And used at the beginning of apex class.

**caseProviderManager Class**

```
@RestResource(urlMapping='/cases/*')
```

```
global class caseProviderManager {
```

```
    @HttpGet
```

```
    global static List<case> caseAllCases(){
```

```
        RestRequest req = RestContext.request;
```

```
        List<case> emptyList = new List<case>();
```

```
        String caseId = req.requestURI.substring(req.requestURI.lastIndexOf('/')+1);
```

```
        if( caseId.length()==18 ){
```

```
            List<case> caseList = [SELECT id , CaseNumber , Subject FROM case WHERE id =:caseId];
```

```
            return caseList;    // if id is given, returns case records of given Id
```

```
        } else if(caseId.contains('case')){
```

```
            List<case> caseList = [SELECT id , CaseNumber , Subject FROM case];
```

```
            return caseList;    // if Id is not given; returns case List
```

```
        } else{
```

```
            return emptyList;    // if nothing is given, returns empty list
```

```
        }
```

```
    }  
    @HttpDelete
```

```
    global static void deleteCase(){
```

```
        RestRequest req = RestContext.request;
```

```
String caseId = req.requestURI.substring(req.requestURI.lastIndexOf('/')+1);
```

```
        Case caseToBeDeleted = [SELECT Id FROM Case WHERE Id =: caseId];
```

```
        delete caseToBeDeleted;
```

```
    }
```

```
    @HttpPost
```

```
    global static Id createCase(string Subject, string Status, string Origin, string Priority){
```

```
        case newCase = new Case();
```

```
        newCase.Subject = Subject;
```

```
        newCase.Status = Status;
```

```
        newCase.Origin = Origin;
```

```
        newCase.Priority = Priority;
```

```
        Database.insert(newCase);
```

```
        return newCase.Id;
```

```
    }
```

```
    @HttpPut
```

```
    global static Id upsertCase(string Subject, string Status, string Origin, string Priority, string Id){
```

```
        case newCase = new Case();
```

```
        newCase.Id = Id;
```

```
        newCase.Subject = Subject;
```

```
        newCase.Status = Status;
```

```
        newCase.Origin = Origin;
```

```
        newCase.Priority = Priority;
```

```
        Database.upsert(newCase);
```

```
        return newCase.Id;
```

```
    }
```

```
    @HttpPatch
```

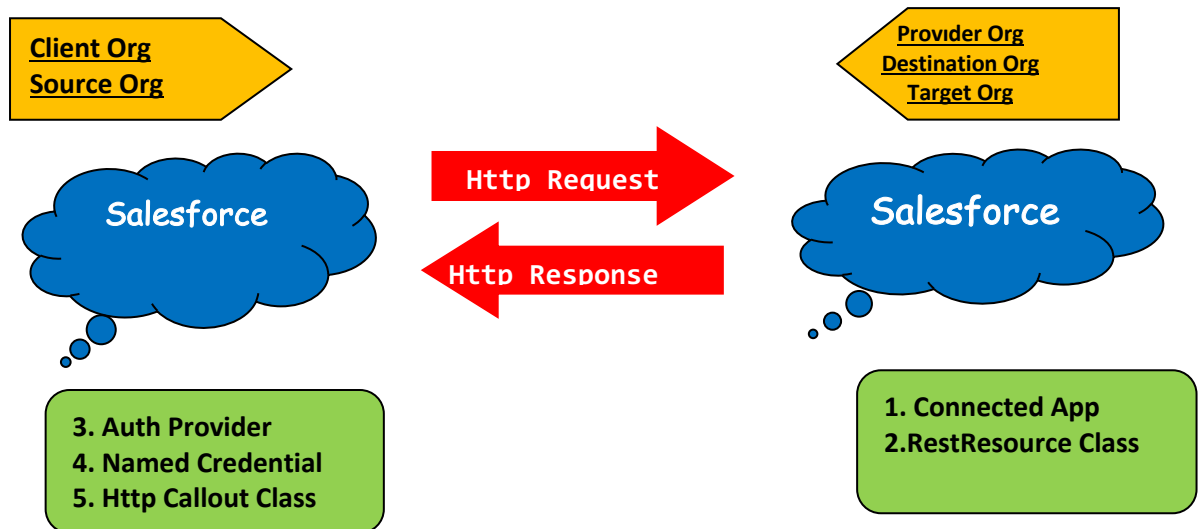
```
    global static Id updateCase(){
```

```
        RestRequest Req = RestContext.request;
```

```
String caseId = req.requestURI.substring(req.requestURI.lastIndexOf('/')+1);
```



## Integrating One Salesforce Org into Another



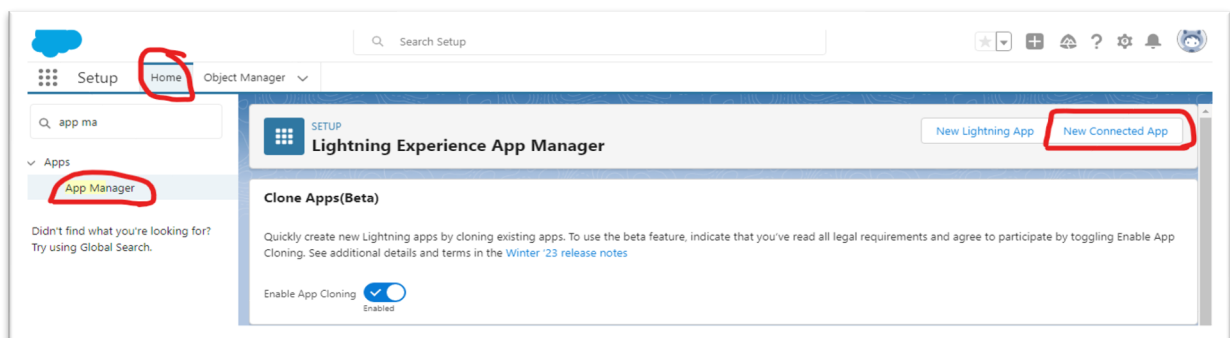
### Step By Step :

- Step 1:** Create connected App in Provider Org.
- Step 2:** Create an Apex RestResource Class in Provider (Target) Org.
- Step 3:** Create AuthProvider in source (Client) Org.
- Step 4:** Create named Credential in Client (source) Org.
- Step 5:** Create Http Callout class in Client (source) Org to fetch or post data from destination (Provider) Org.

### Step 1: Create a Connected App in Provider (Target) Org

- Connected apps are a framework that enables the external app to integrate with Salesforce via APIs.
- It uses OAuth protocols to authenticate and provide tokens to use with Salesforce APIs.

- Go to Setup > App > App Manager in Provider Org.
- Click on the 'New Connected App' Button in the "Connected App" section.



- Enter the **Name** of the Application.





- Enter your **Email** and any further information suitable for your application.
- Enable **OAuth settings** in the API section.
- In “**Callback URL**” enter the temporary Salesforce URL. We will come back again on this step later to provide Callback URL (for example (<https://trailhead.salesforce.com/>) or (<https://localhost.com>)) as callback URL.
- Add **Selected OAuth Scopes**. Here I’m giving “Full access(full).”
- Click on the ‘Save’ button, then ‘Continue’ button.



Search Setup

Setup Home  
Service Setup Assistant  
Multi-Factor Authentication Assistant  
Release Updates  
Lightning Experience Transition Assistant  
New Salesforce Mobile App QuickStart  
Lightning Usage  
Optimizer

ADMINISTRATION  
Users  
Data  
Email

PLATFORM TOOLS  
Apps  
AppExchange Marketplace  
Connected Apps  
Lightning Bolt  
Mobile Apps  
Packaging  
Feature Settings  
Slack  
Einstein

## App Manager

### New Connected App

Save Cancel

**Basic Information**

Connected App Name

API Name

Contact Email

Contact Phone

Logo Image URL

Icon URL

Info URL

Description

**API (Enable OAuth Settings)**

Enable OAuth Settings ☒

Enable for Device Flow ☐

Callback URL

Use digital signatures ☐

Selected OAuth Scopes

**Available OAuth Scopes**

- Access Analytics REST API Charts Geodata resources (eclair\_api)
- Access Analytics REST API resources (wave\_api)
- Access Connect REST API resources (chatter\_api)
- Access Lightning applications (lightning)
- Access Salesforce applications (visualforce)
- Access chatbot services (chatbot\_api)
- Access content resources (content)
- Access custom permissions (custom\_permissions)
- Access the identity URL service (id, profile, email, address, phone)
- Access unique user identifiers (openid)

**Selected OAuth Scopes**

Full access (full)

Add Remove

➤ **After creating the Connected App, here are the steps you need to follow:**

- Click on Manage Consumer Details.
- You will receive an OTP in the Mail for verification.
- Get the “Consumer Key “ and “Consumer secret,” as these details are needed to authenticate the external application.

**API (Enable OAuth Settings)**

Consumer Key and Secret [Manage Consumer Details](#)

Selected OAuth Scopes Full access (full)

Callback URL <https://trailhead.salesforce.com/>

Enable for Device Flow ☐

Require Secret for Web Server Flow ☒

Require Secret for Refresh Token Flow ☒

Enable Client Credentials Flow ☐

Introspect All Tokens ☐

Token Valid for 0 Hour(s)

Include Custom Attributes ☐

Include Custom Permissions ☐

Enable Single Logout Single Logout disabled

**Initial Access Token for Dynamic Client Registration**

Initial Access Token [Generate](#)

**Custom Connected App Handler**

**salesforce**

## Verify Your Identity

You're trying to **Access a Connected App**. To make sure your Salesforce account is secure, we have to verify your Identity.

Enter the verification code we emailed to [||\\*\\*\\*\\*\\*@\\*\\*\\*||.com](#).

Verification Code

[Back](#) [Verify](#)

[Resend Code](#)



Connected App Name  
**mySelfConnection**

« Back to Manage Connected Apps

**Consumer Details**

Consumer Key	3MVG9DREgiBqN9M...
	<a href="#">Copy</a>
Consumer Secret	70522F...
	<a href="#">Copy</a>

**Staged Consumer Details**

Generate staged values for the consumer key and secret. When you apply the staged values, they replace the original consumer details.

Staged Consumer Key	Not generated
Staged Consumer Secret	Not generated


[Generate](#) [Apply](#) [Cancel](#)

- Click on 'Cancel' then Click on 'Manage
- Click on Edit policies.
- Click on Ip Relaxation and select Relax IP Restrictions and Click save.

Connected App Name  
**mySelfConnection**

« Back to List: Custom Apps

[Edit](#) [Delete](#) [Manage](#)



Version	1.0
API Name	mySelfConnection
Created Date	27.12.2022 15:43
By	<a href="#">iker kumperi</a>
Contact Email	<a href="#">ikerkmprli49@gmail.com</a>
Contact Phone	
Last Modified Date	27.12.2022 15:43
By	<a href="#">iker kumperi</a>
Description	
Info URL	

▼ API (Enable OAuth Settings)

Consumer Key and Secret	<a href="#">Manage Consumer Details</a>
Selected OAuth Scopes	Full access (full)
Callback URL	<a href="https://trailhead.salesforce.com/">https://trailhead.salesforce.com/</a>
Enable for Device Flow	<input type="checkbox"/>
Require Secret for Web Server Flow	<input checked="" type="checkbox"/>
Require Secret for Refresh Token Flow	<input checked="" type="checkbox"/>
Enable Client Credentials Flow	<input type="checkbox"/>
Introspect All Tokens	<input type="checkbox"/>
Token Valid for	0 Hour(s)
Include Custom Attributes	<input type="checkbox"/>
Include Custom Permissions	<input type="checkbox"/>
Enable Single Logout	Single Logout disabled

▼ Initial Access Token for Dynamic Client Registration

Initial Access Token	<a href="#">Generate</a>
----------------------	--------------------------

▼ Custom Connected App Handler


Apex Plugin Class	
Run As	

Trusted IP Range for OAuth Web Server Flow [New](#)

Connected App  
**mySelfConnection**

Connected App Detail

[Edit Policies](#)



Quick Find / Search... [Expand All](#) | [Collapse All](#)

**Lightning Experience Transition Assistant**  
Move to the new, more productive Salesforce.  
[Get Started](#)

**Salesforce Mobile Quick Start**


**Home**

**Administer**

- [Release Updates](#)
- [Manage Users](#)
- [Manage Apps](#)
- [Connected Apps](#)
- [Connected Apps OAuth Usage](#)
- [App Menu](#)
- [Manage Territories](#)
- [Company Profile](#)
- [Data Classification](#)
- [Privacy Center](#)
- [Security Controls](#)
- [Domain Management](#)
- [Communication Templates](#)
- [Translation Workbench](#)
- [Data Management](#)

Connected App  
**mySelfConnection**

**Connected App Edit**



Version 1  
Description

**Basic Information**

Start URL		Mobile Start URL	
-----------	--	------------------	--

**OAuth Policies**

Permitted Users	<a href="#">All users may self-authorize</a>	IP Relaxation	<a href="#">Relax IP restrictions</a>
Enable Single Logout	<input type="checkbox"/>	Refresh Token Policy	<input checked="" type="checkbox"/> Immediately expire refresh token

**Session Policies**

Timeout Value	None	<input type="checkbox"/> High assurance session required
---------------	------	--

**Custom Connected App Handler**

Apex Plugin Class		Run As	
-------------------	--	--------	--

**User Provisioning Settings**

☐ Enable User Provisioning

[Save](#) [Cancel](#)



**Step 2: Create an Apex RestResource Class in Provider (Target) Org to Client (Source) Org.** [Click for Apex class with RestResource](#)

**Step 3: Create AuthProvider in Client (Source) Org.**

- Go to Setup > Home > Auth. Provider in Client Org.
- Click on the 'New' Button in the "Auth. Provider" section.

The screenshot shows the Salesforce Setup interface. The left sidebar has a search bar with 'auth' and a list of categories: Multi-Factor Authentication Assistant, Apps, Connected Apps, OAuth Usage, Identity, Auth. Providers (selected), OAuth Custom Scopes, and OAuth and OpenID Connect Settings. The main content area is titled 'Auth. Providers' and shows a table with columns: Name, URL Suffix, Provider Type, and Created Date. A 'New' button is highlighted with a red arrow.

- Select Salesforce as Provider Type.

The screenshot shows the Salesforce Setup interface for editing an Auth. Provider. The left sidebar has a search bar with 'auth' and a list of categories: Multi-Factor Authentication Assistant, Apps, Connected Apps, OAuth Usage, Identity, Auth. Providers (selected), OAuth Custom Scopes, and OAuth and OpenID Connect Settings. The main content area is titled 'Auth. Provider' and shows the 'Auth. Provider Edit' form. The 'Provider Type' dropdown menu is set to '--None--' and is highlighted with a red arrow.

- Then, fill in the required sections.
  - Write a Name for Auth. Provider.
  - Write "Consumer Key" and "Consumer Secret" getting from the 'Connected App' step.
  - In "Default Scope" enter the value as "refresh\_token full".
  - then Save



Auth. Provider

Auth. Provider Edit

Provider Type: **Salesforce**

Name:

URL Suffix:

Consumer Key:

Consumer Secret:

Authorize Endpoint URL:

Token Endpoint URL:

Default Scopes:

Include Consumer Secret in SOAP API Responses: ☒

Custom Error URL:

Custom Logout URL:

Registration Handler:

Execute Registration As:

Portal: **--None--**

Icon URL:

Save Save & New Cancel

- When you save, it will provide you the set of URLs in “Salesforce Configuration” section on the same page. Copy “**Callback URL**” and edit Connected App we created in the previous step and set this URL as Callback URL.

Salesforce Configuration

Test-Only Initialization URL	https://mycompany20-dev-ed.develop.my.salesforce.com/services/auth/test/Sample_Auth_Provider
Existing User Linking URL	https://mycompany20-dev-ed.develop.my.salesforce.com/services/auth/link/Sample_Auth_Provider
OAuth-Only Initialization URL	https://mycompany20-dev-ed.develop.my.salesforce.com/services/auth/oauth/Sample_Auth_Provider
<b>Callback URL</b>	<b>https://mycompany20-dev-ed.develop.my.salesforce.com/services/auth/callback/Sample_Auth_Provider</b>
Single Logout URL	https://mycompany20-dev-ed.develop.my.salesforce.com/services/auth/protocol/logout

Edit Delete Clone

## Step 4: Create named Credential in Client (Source) Org.

Using Named Credential, we can make call out to external system without supplying username or Password.

A named credential specifies the URL of a callout endpoint and its required authentication parameters in one definition. To simplify the setup of authenticated callouts, specify a named credential as the callout endpoint.

- Navigate to Setup -> Named Credentials.
- Click on the ‘New Legacy’ Button.

Setup

Named Credentials

Named Credentials External Credentials

New Legacy

- Enter a label Name.
- In URL, Provide the Salesforce instance URL of Provider org (Given in my domain in setup of Provider Org). Or execute `system.debug(system.url.getSalesforceBaseUrl())` in Anonymous Windows.
- Select “Named Principal” in Identity Type.
- Select “OAuth 2.0” in Authentication Protocol.
- Select the Outh Provider we have created in the previous step in Authentication Provider.



- In scope, enter the value as “refresh\_token full” (not to verify every time)
- Check “Allow Merge Fields in HTTP Body” checkbox.(this is important to add request body to API callout) Finally, Save.

The screenshot shows the 'New Named Credential' configuration page in Salesforce Setup. The page is titled 'Named Credentials' and 'New Named Credential'. It includes a sidebar with navigation links for 'User Interface', 'Rename Tabs and Labels', 'Security', and 'Named Credentials'. The main content area contains the following fields and options:

- Label:** sampleCredential (indicated by a red arrow)
- Name:** sampleCredential (indicated by a red arrow)
- URL:** (indicated by a red arrow)
- Authentication:**
  - Identity Type:** Named Principal (indicated by a red arrow)
  - Authentication Protocol:** OAuth 2.0 (indicated by a red arrow)
  - Authentication Provider:** Sample Auth Provider (indicated by a red arrow)
  - Scope:** refresh\_token full (indicated by a red arrow)
  - Authentication Status:** Pending
  - Start Authentication Flow on Save:** ☒
- Callout Options:**
  - Generate Authorization Header:** ☒
  - Allow Merge Fields in HTTP Header:** ☐
  - Allow Merge Fields in HTTP Body:** ☐ (indicated by a red arrow)
  - Outbound Network Connection:** (indicated by a red arrow)

- After save, Enter Username and Password of the Provider Org.



## Step 5: Create Http Callout class in Client (source) Org to fetch data from destination(Provider) Org.

Main Syntax of Http Callout class:

```
public class MyCalloutClass {  
  
    public static void getFromProvider(){  
        Http http = new Http();  
        HttpRequest req = new HttpRequest();  
        req.setEndpoint('callout:Sample_Named_Credentials/services/apexrest/cases/');  
        req.setMethod('GET');  
  
        HTTPResponse resp = http.send(req);  
    }  
}
```

### Example of Getting Case List From Provider to Client:

```
public class MyCalloutClass {  
  
    public static void getFromProvider(){  
        Http http = new Http();  
        HttpRequest req = new HttpRequest();  
        req.setEndpoint('callout:Sample_Named_Credentials/services/apexrest/cases/');  
        req.setMethod('GET');  
  
        HTTPResponse resp = http.send(req);  
  
        if(resp.getStatusCode() == 200) {  
  
            List<object> jsonTmpList = (List<object>)JSON.deserializeUntyped(resp.getBody());  
  
            // From JSON Array to sObject List. and we used deserializeUntyped() because maybe we don't know the format of response.  
  
            List<case> CaseList = new List<case>();  
  
            for(object obj: jsonTmpList){  
                Map<String, Object> caseParam = (Map<String, object>) obj;  
                case tempCase = new case();  
                tempCase.Status = (String) caseParam.get('Status');  
                tempCase.Origin = 'Web';  
                tempCase.Subject = (String) caseParam.get('Subject');  
                tempCase.Priority = 'Low';  
                CaseList.add(tempCase);  
            }  
            Database.insert(CaseList);  
        }else {  
            system.debug('Error :: ' + resp.getStatusCode() );  
        }  
    }  
}
```



## Example of Posting Case List (as a String) from Client to Provider Org : (By using Trigger)

```
public class MyCalloutClass {
    @future(callout=true)
    public static void sendToProvider(){
        Http http = new Http();
        HttpRequest req = new HttpRequest();
        req.setEndpoint('callout:Sample_Named_Credentials/services/apexrest/cases/');
        req.setMethod('POST');
        req.setHeader('Content-Type', 'application/json;charset=UTF-8');

        String body = '{"Subject" : "Case From Client", "Status":"New", "Origin":"Web", "Priority":"Low"}';
        Req.setBody(body);

        HTTPResponse resp = http.send(req);

        if(resp.getStatusCode() == 201 || resp.getStatusCode() == 200 ) {

            System.debug('response ' + resp.getBody());

        }else if (resp.getStatusCode() == 302) {

            Req.setEndpoint(resp.getHeader('Location'));
            HTTPResponse respons = new http().send(req);

        }else {
            system.debug('Error :: ' + resp.getStatusCode() );
        }
    }
}
```

### sendToProviderTrigger

```
trigger sendToProviderTrigger on Case(after insert, after update){

    For(Case cs: Trigger.new){

        if(cs.insertToGoRest__c == true){

            MyCalloutClass.sendToProvider();

        }

    }

}
```

When we want to run the method with the trigger, we should run it with the `@future(callout=true)` method.

`insertToGoRest` checkbox field is created in case previously.