IBM Integration Bus

Integration with Salesforce

Featuring:

- Salesforce.com
- Creating a REST API using IIB Toolkit
- Using Mapping Nodes with REST API
- Salesforce Request Nodes

April 2016
Hands-on lab built at product
Version 10.0.0.4

1. Introduction

IBM Integration Bus V10 Fix pack 4 enables you to interact with your Salesforce.com account in the cloud, by using a Salesforce request node in a message flow. You can use this capability to create, retrieve, update, and delete records in your Salesforce system directly from IBM Integration Bus.

1.1 Outline of Lab

In this lab you will create a REST API to interface with the Contacts object in Salesforce.com. The REST API will be created with three operations:

1) an insertContact operation that will be used to Create a new contact in Salesforce.com. This operation will use data passed in the body of the request as input into the Salesforce request.
2) a retieveAll operation that will be used to retrieve all contacts available to your Salesforce user. No parameters will be required for this operation
3) a retrieveContact operation that can be used to retrieve a specific. This operation will require a (mandatory) Id parameter to be passed in the path of the request.

The lab guide will also guide you through the Salesforce verification process for its webUI as well as security requirements for userid and password client access.

Finally you will test the three operations and successfully add and retrieve contacts in Salesforce.com.

2. Configure IIB node for Salesforce

2.1 Change the Operation mode

In order to deploy a message flow that contains a Salesforce node, it is necessary to change the operation mode of the IIB Node to run in either "**Application Integration Suite**" mode or "**developer**" mode.

Note: at the time of writing this Lab Guide an "Application Integration Suite" license is required in order to run in this mode. For more information on the IIB licensing requirements please contact your local IBM representative.

_1. In an Integration Console enter the following command:

```
mqsimode TESTNODE_iibuser
         -o applicationIntegrationSuite
```

Expected response:

```
BIP1809I: Changing the mode of integration node 'TESTNODE_iibuser' to
'applicationIntegrationSuite'...
BIP8071I: Successful command completion.
```

2.2 Create a Security Identity

In order for a Salesforce node to connect successfully a security identity must be configured in IIB. This avoids re-configuring all Salesforce nodes that use the same userid and password when for example the password changes.

Configure the Security Identification used in this lab guide (configuring the identity in this way also avoids exposing the salesforce.com userid and password to the IIB Application developer).

_1. In an Integration Console enter the following command:

```
mqsisetdbparms TESTNODE_iibuser
               -n salesforce::SF1
               -u <yoursalesforceId>
               -p <yourSalesforceIDpassword>
```

Expected response:

```
BIP8071I: Successful command completion.
```

Note: if you are using the Salesforce ids supplied during the IIB Workshop these values are "appconnect-lab-*XX*@uk.ibm.com" (XX from 01 to 40) with a password of "appc0nn3ct"

2.3 Set the IIB node to run CORS enabled

Set the node to run Cross Origin Resource Sharing (CORS). If you don't do this you will have problems accessing the REST API and will see the error message "*Can't read from server. It may not have the appropriate access-control-origin settings.*"

_1. Check IIB Node "TESTNODE_iibuser" is running.

_2. In an Integration Console enter the following command:

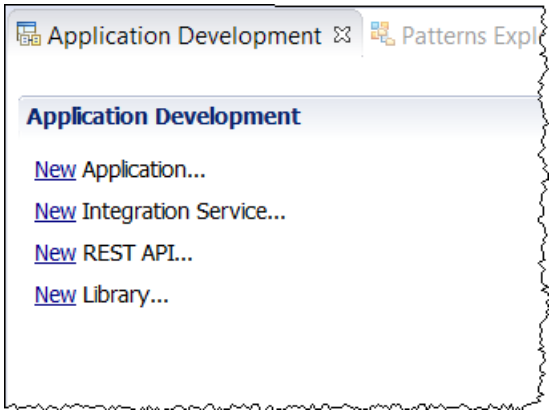
```
mqsichangeproperties TESTNODE_iibuser
                     -e default
                     -o HTTPConnector
                     -n corsEnabled
                     -v true
```

_3. Stop and restart TESTNODE_iibuser for the changes to take effect

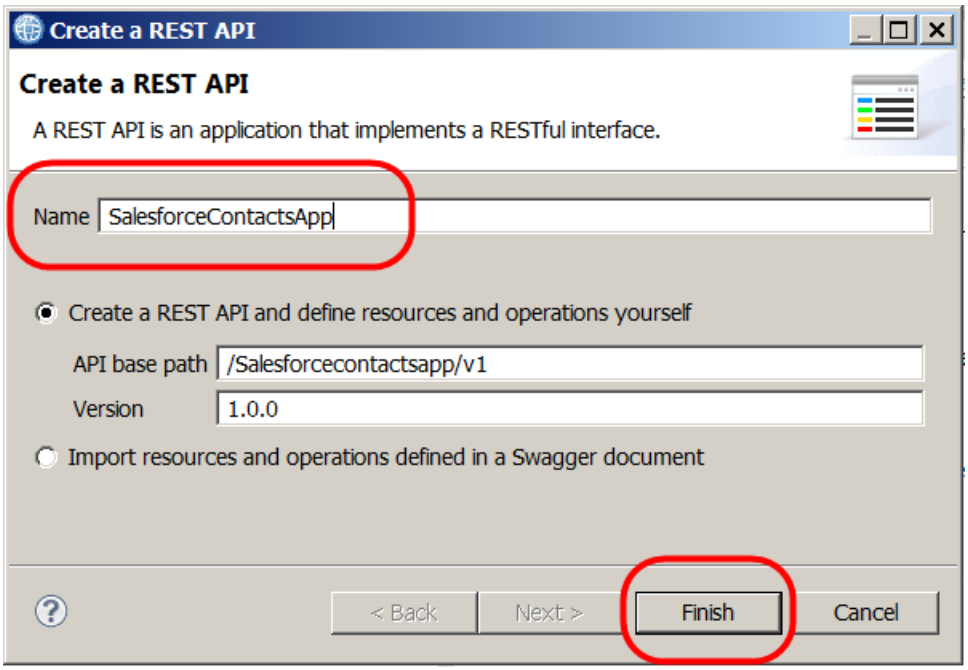
3. Create the SaleforceContactsApp REST API

_1. In the Integration Toolkit, create a new workspace called Workspace_salesforce.

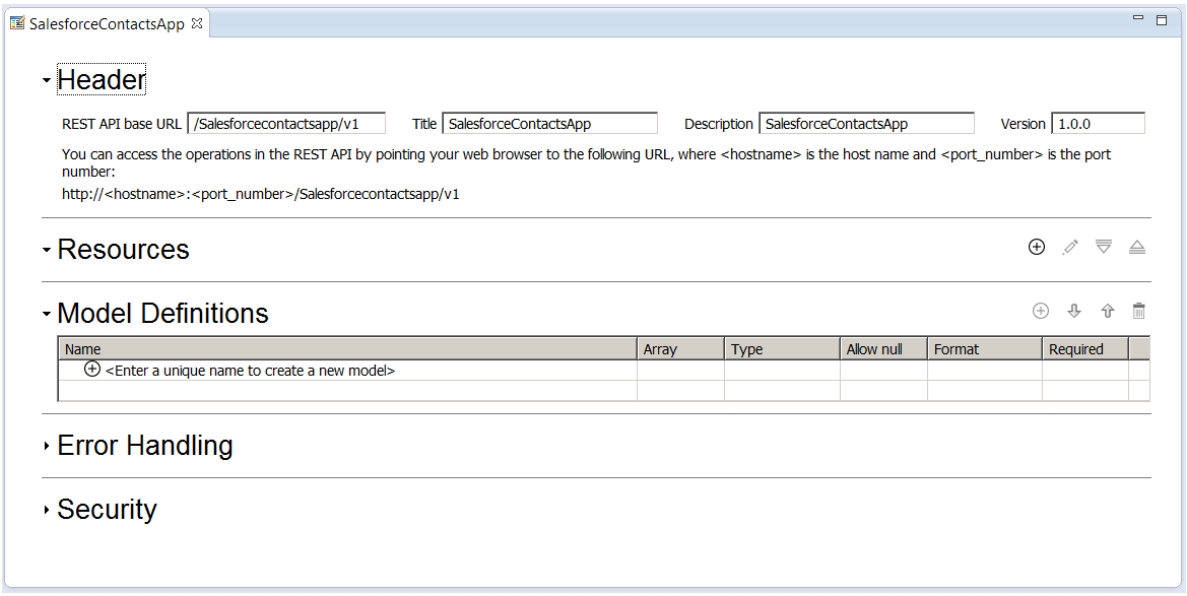
_2. When the Integration Toolkit restarts in the new workspace, click “New REST API ….”



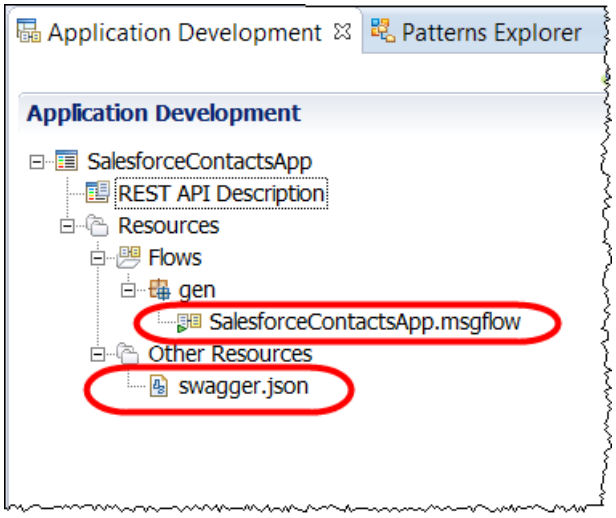
_3. Call the REST API “`SalesforceContactsApp`” and click Finish:



_4. The REST API development configuration will open:



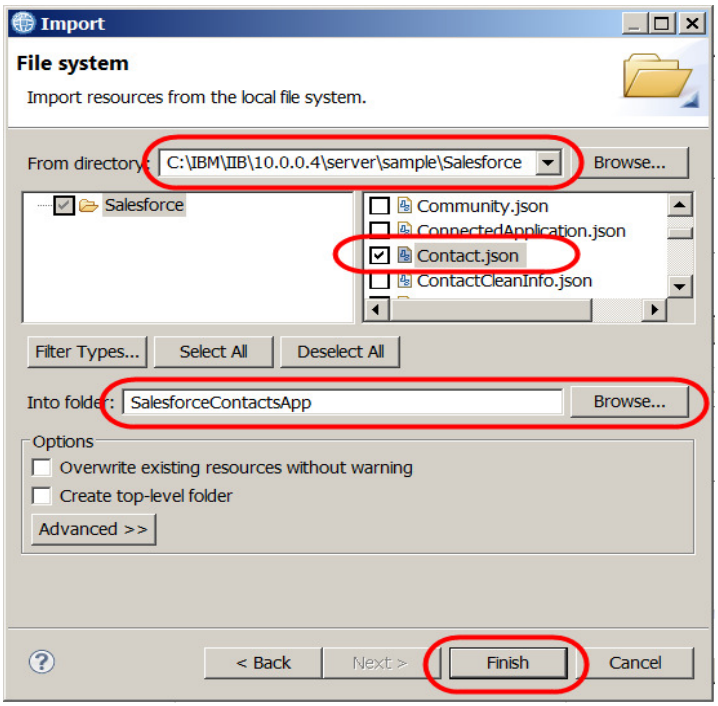
_5. Note creating the REST API has automatically created a message flow and a swagger.json file:



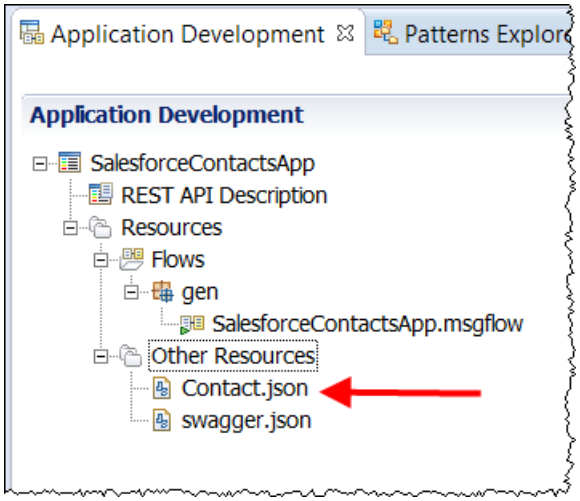
3.1.1 Import Salesforce Contacts Schema

IIB provides JSON schemas for all of the Saleforce object definitions. These are located in `<InstallDirectory>\server\sample\Salesforce\`. The schema for Salesforce Contacts is called Contacts.json, you will now import this schema.

_1. Click Import and select General > File system.
In the wizard navigate to "`C:\IBM\IIB\10.0.0.4\server\sample\Salesforce`" and select `Contact.json`.
Import the json schema into the SalesforceContactsApp and click Finish.



_2. A copy of the Contact.json schema will now appear in the REST API:



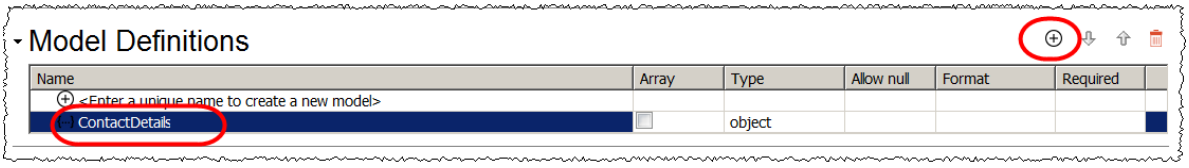
3.2 Configure the REST API Description

3.2.1 Create ContactDetails Model Definition

The contact.json schema has many fields, not all of them mandatory. Rather than use the contact.json as input to the REST API you will define a radically cut down version of the schema as input to the REST API.

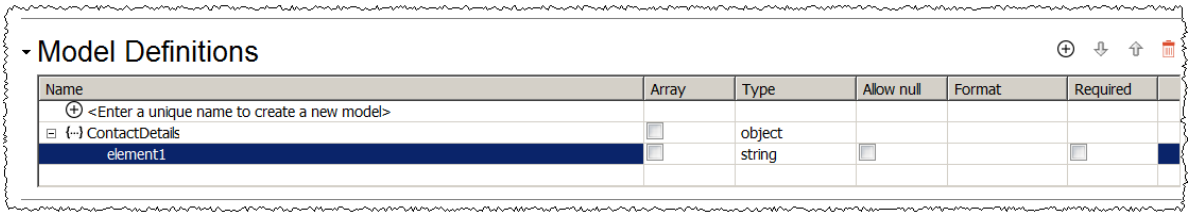
You will now create a model definition in the REST API consisting of three elements. This information will be used as the source of the input data in the SalesforceContactsApp when adding a contact to Salesforce.

_1. In the Model Definitions section of the REST API configuration screen, over type “<Enter a unique name to create a new model>” with name “**`ContactDetails`**” and press Enter:

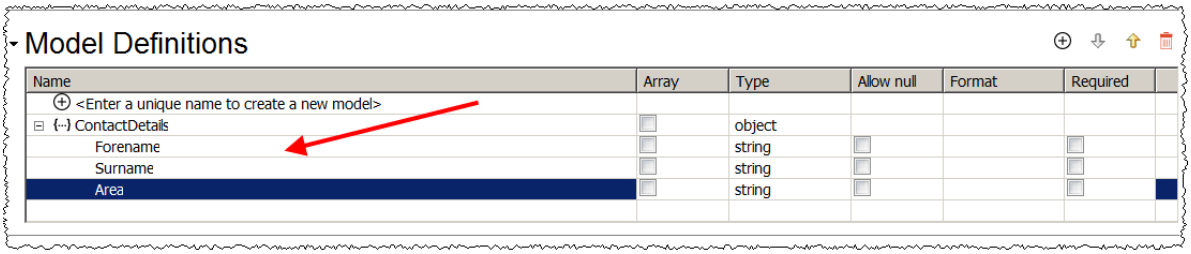


_2. With “ContactDetails” highlighted, click the plus sign to add a child to the ContactDetails object:

Overtype “element1” with Forename:



_3. Repeat the previous step to create string objects “Surname” and “Area” (*ensure “ContactDetails” is highlighted before you click the plus sign*) :



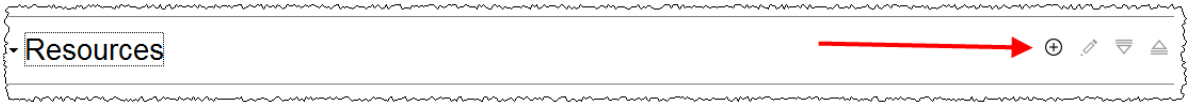
3.2.2 Create Resources

In the following section you will add two resources:

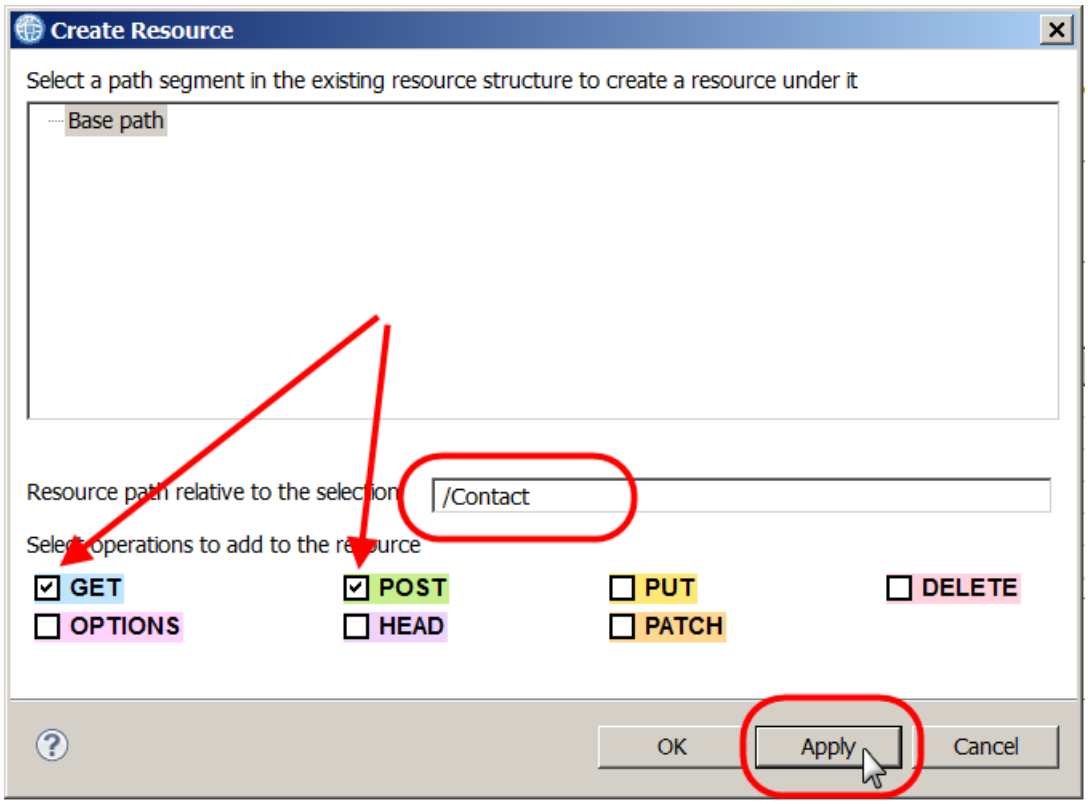
- **"/Contacts"** will have two operations defined:
  - GET which will retrieve all contacts;
  - POST that will be used to add a contact

- **"/Contacts/{id}"** will have one operation defined:
  i. GET operation which will be used to retrieve a specific contact.

_1. Click the plus sign in Resources:

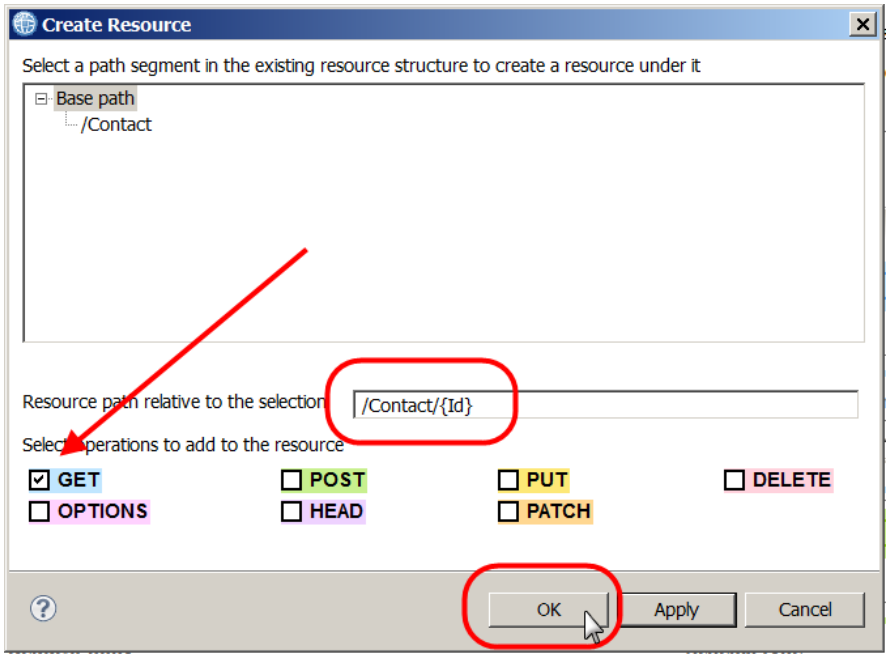


_2. In the Create Resource window enter "/Contact" and click the GET and POST tick boxes. Click Apply.



In the background you will see the Blue (GET operation) and Green (POST operation) created under Resources. The Create Resource window will remain open.

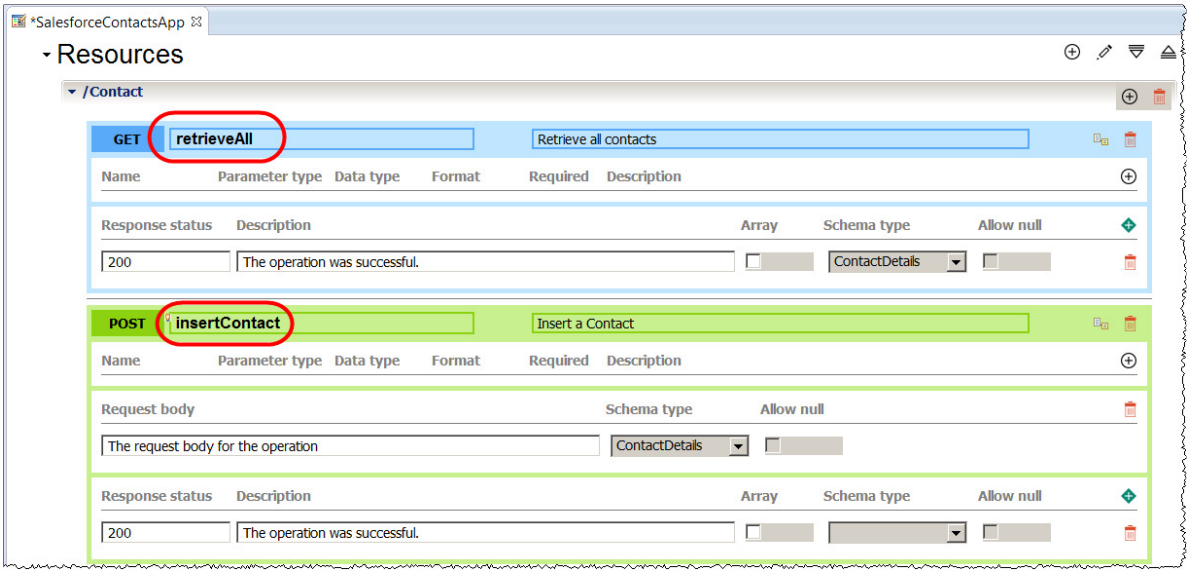
_3. Add another resource called "`/Contact/{Id}`", select the Get operation and click OK:



_4. Expand the resource "/Contact". Note that two operations (one POST and one GET) have been defined.

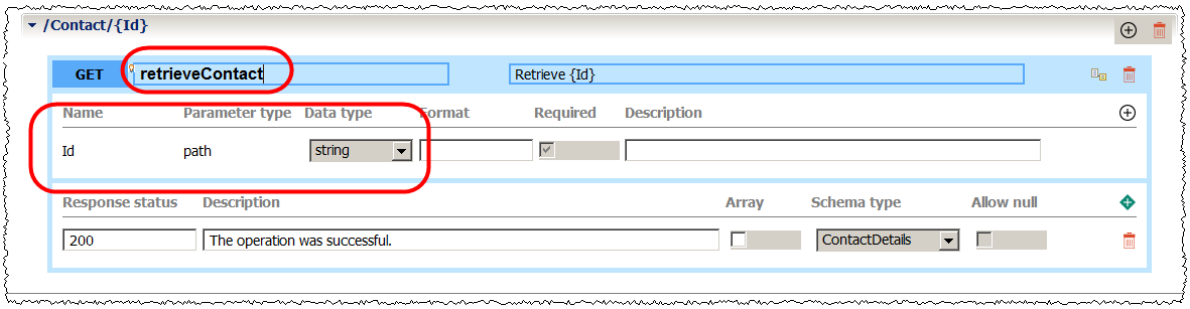
In the GET operation, rename the default name from "`get1`" to "`retrieveAll`".

In the POST operation rename the default name for the operation from "`post1`" to "`insertContact`".



_5. Expand "/Contact/{Id} and note that a GET operation has been defined. Rename "`get1`" to "`retrieveContact`":

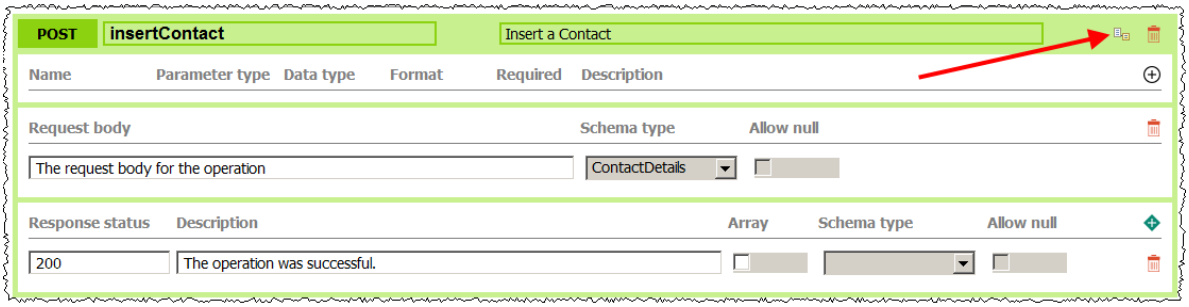
(Note that "Id" has been automatically created as a parameter defined to be in the path)



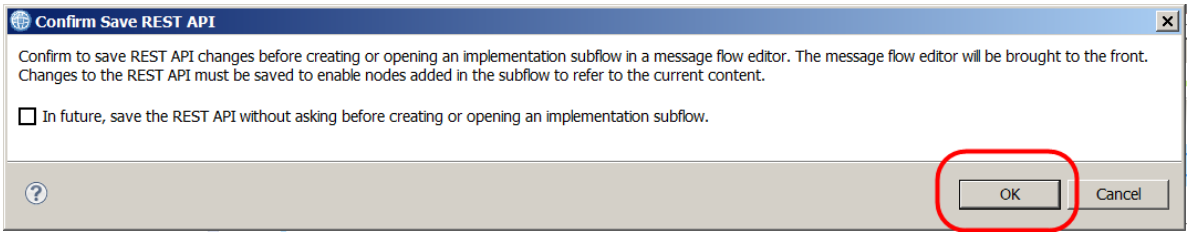
3.3 Configure the insertContact operation

All operations in IBM Integration Bus REST APIs are implemented using a subflow. You will now implement the subflows for each of the operations you have defined in the REST API.

_1. In the insertContact (POST) operation, click the icon to “Create a subflow for the operation”:



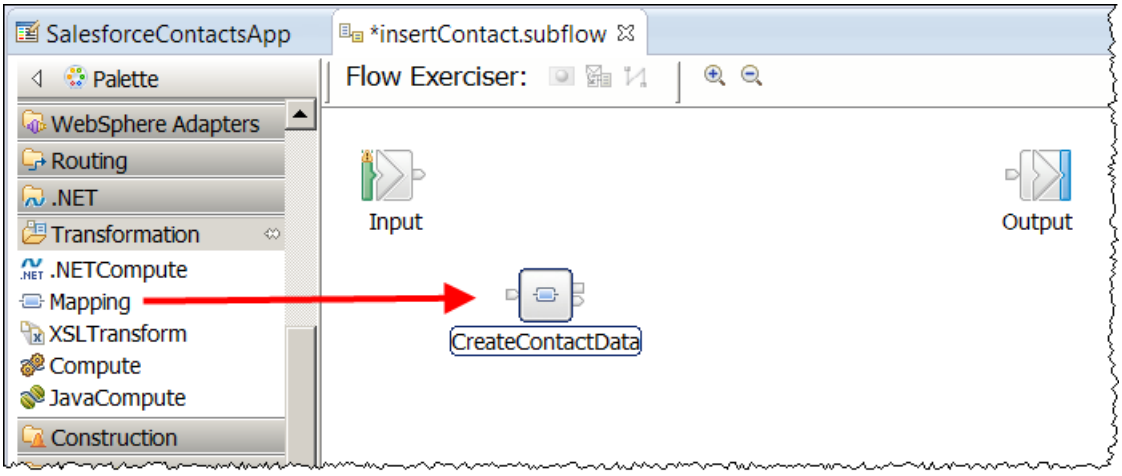
_2. Click OK on the “Confirm Save REST API” window:



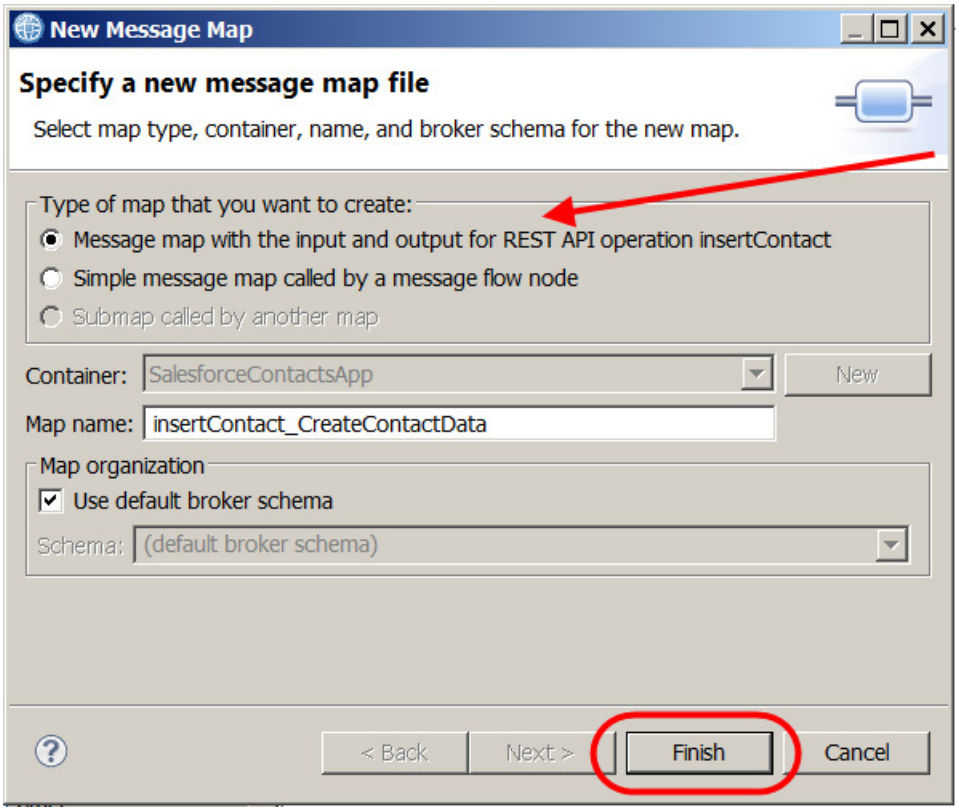
_3. The subflow editor will open.

3.3.1 Add CreateContactData Mapping Node

_1. Add a mapping node (in the Transformation Folder) to the subflow editor. Call the mapping node “CreateContactData”:

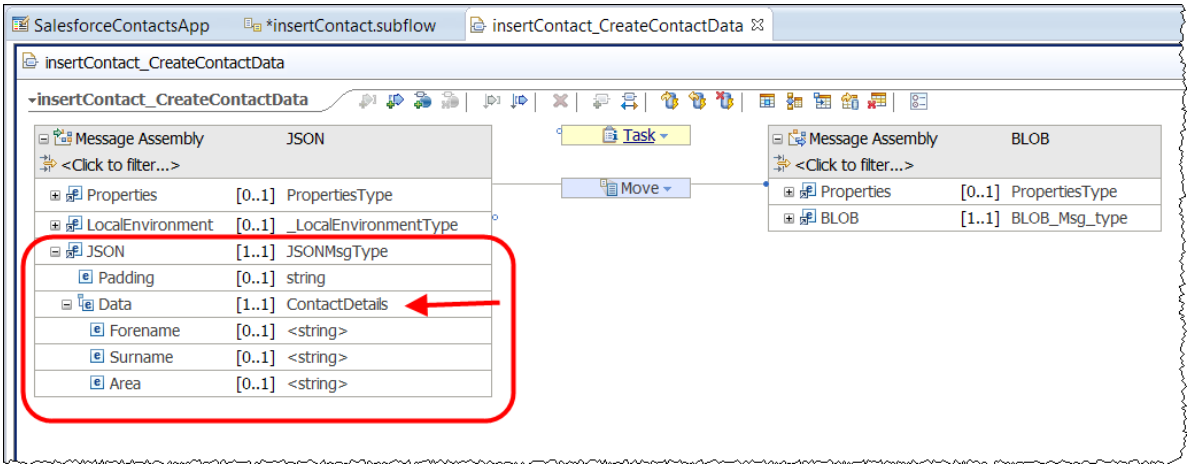


_2. Double click on the mapping node to open the mapping editor. *Note the mapping node is aware of the REST API definitions and has used this to default the type of map being created.* Accept the defaults and click Finish:



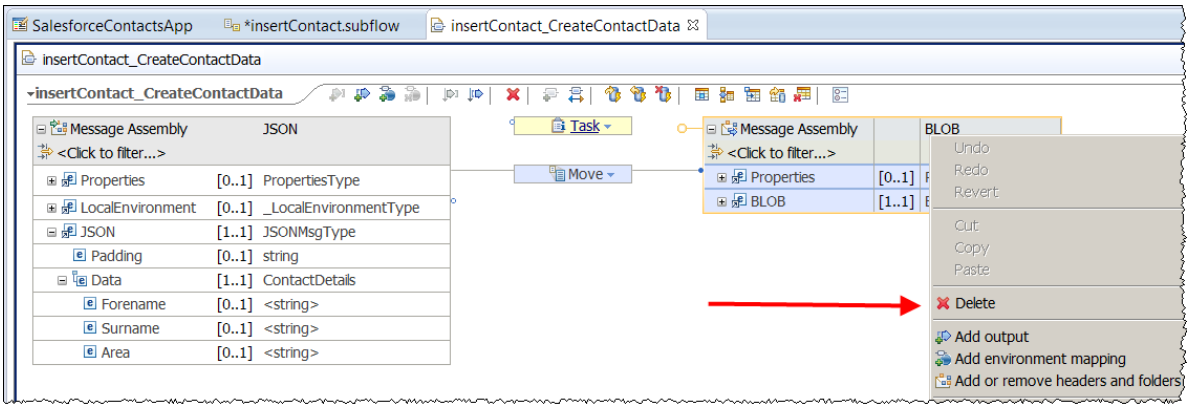
NOTE: The maps that are created in this way can not be moved as they have internal configuration relating to the operations they were originally configured in.

_3. When the mapping editor opens, expand the input Message Assembly and note that the "ContactDetails" model definition has automatically been added to the input message:

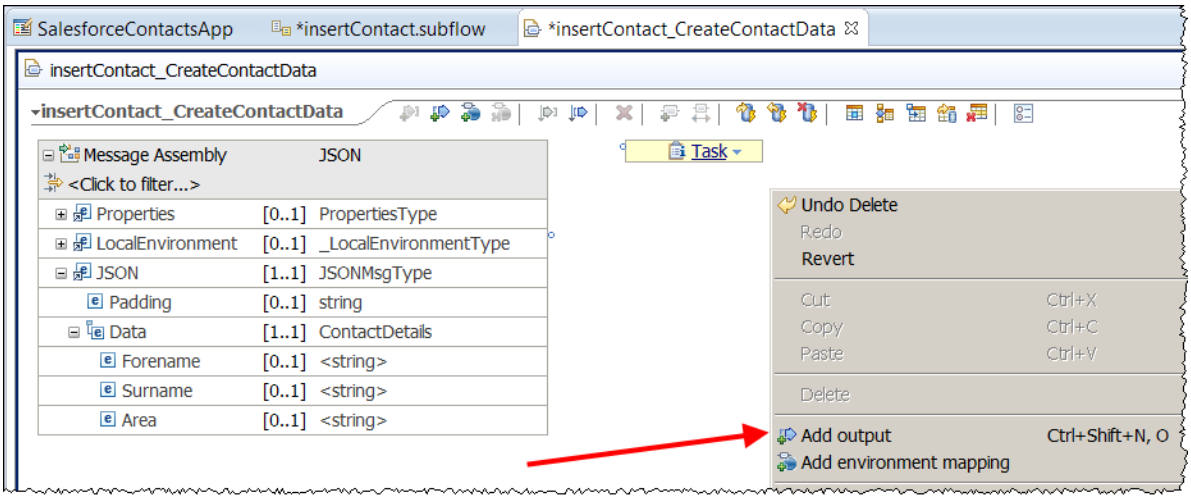


_4. The default output on the Message Assembly for a POST operation is a BLOB. The output from this map needs to be the Contact JSON schema required by Salesforce.

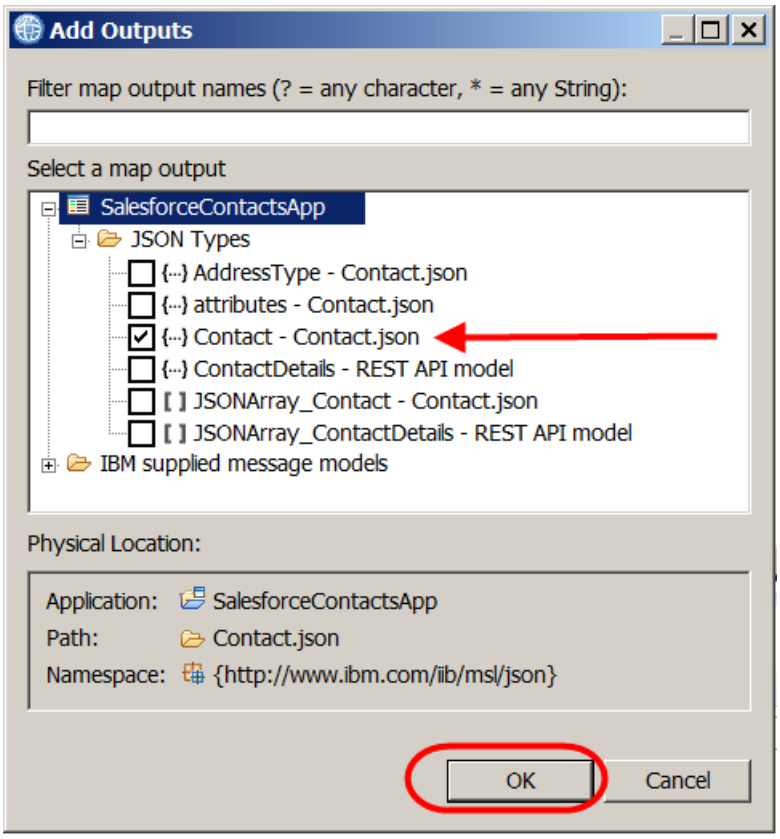
Right click on the “Message Assembley BLOB” and click delete from the options list:



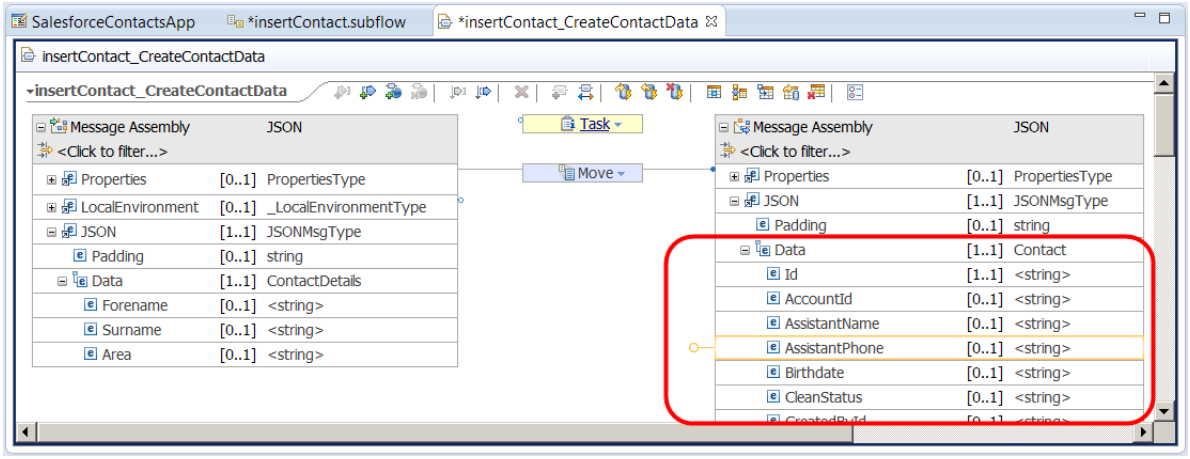
_5. Right click on the white background and select “Add Output”



_6. In the Add Outputs window, expand SalesforceContactsApp > JSON Types and select Contact-Contact.json (you added the Contact.json file to the REST API earlier):



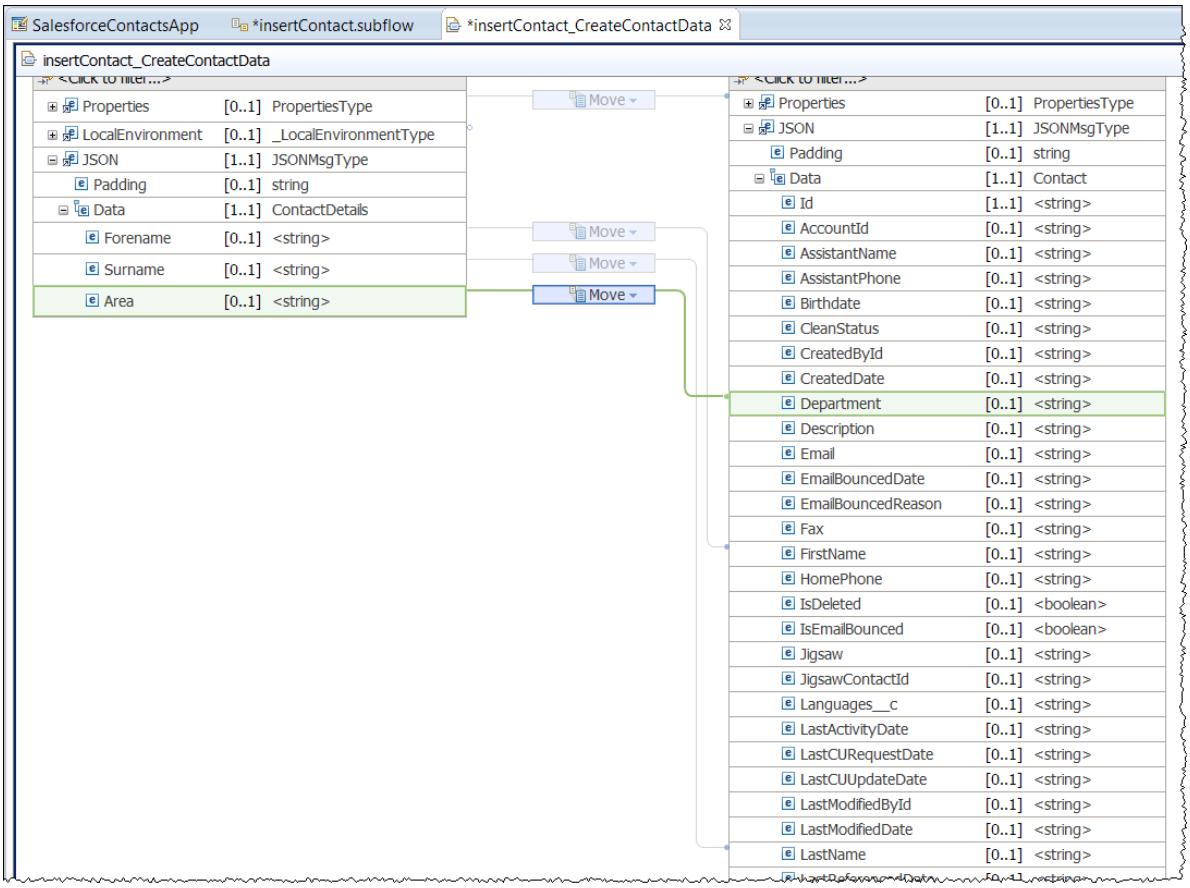
_7. Note the Contact json schema required by Salesforce has now been added to the Output Message Assembly:



_8. Map the following Input/Output fields:

Forename -> FirstName
Surname -> LastName
Area -> Department

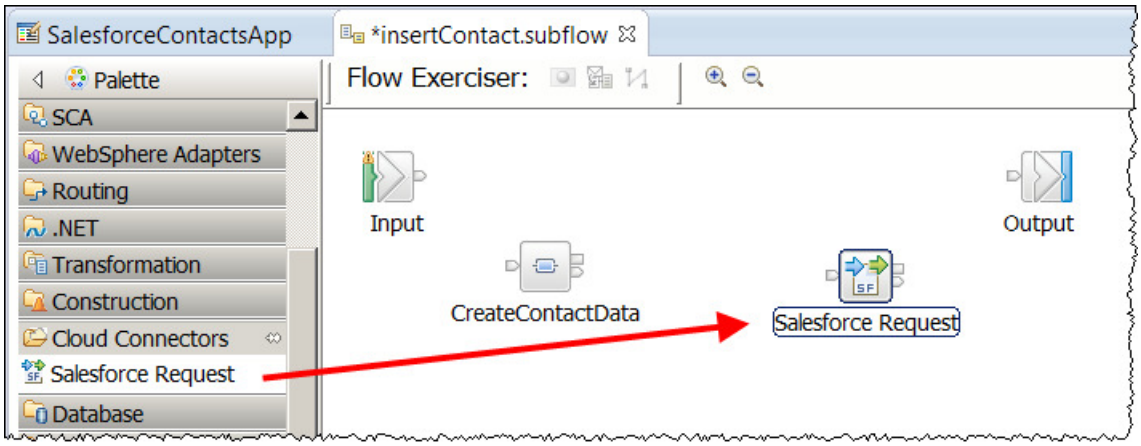
_9. The Map will look like this when complete:



_10 Save and close the map.

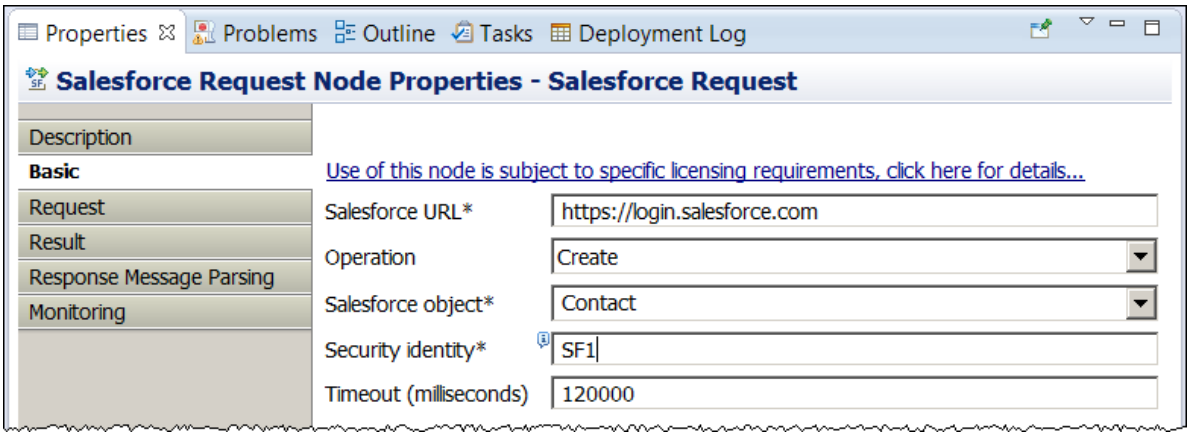
3.3.2 Configure Salesforce Request node

_1. In insertContact.subflow add a Salesforce Request node (Cloud Connectors) to the subflow:

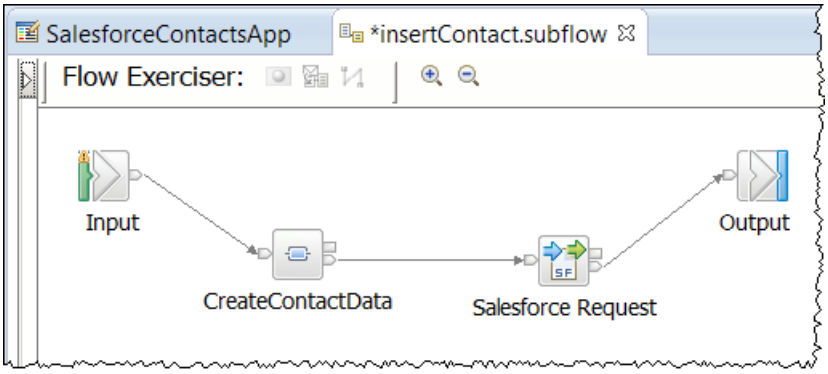


_2. In the basic tab configure the following:

Salesforce URL : `https://login.salesforce.com`
Operation : `Create`
Salesforce object : `Contact`
Security identity : `SF1`



_3. Join the node output terminals in the subflow as follows:

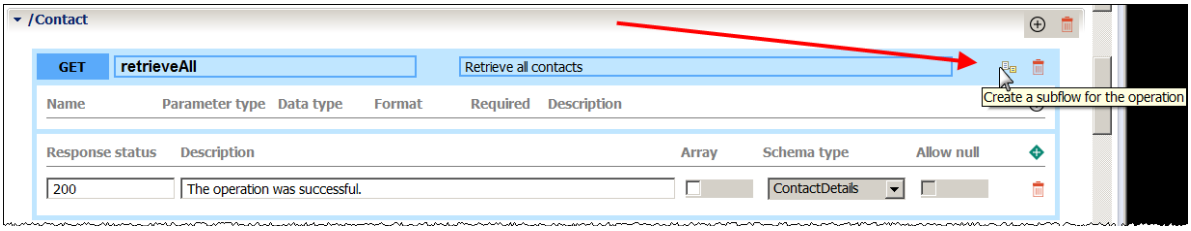


_4. Save and close the subflow.

3.4 Configure the retrieveAll Operation

The retrieveAll operation will obtain all Contacts available to your userid in Salesforce.com. No parameter is required to obtain these details, so no mapping node is required in the implementation of the operation. The Salesforce Request data forms part of the LocalEnvironment Header and does not require specific configuration before a Salesforce request node in the subflow. The implementation only requires a Salesforce Request node.

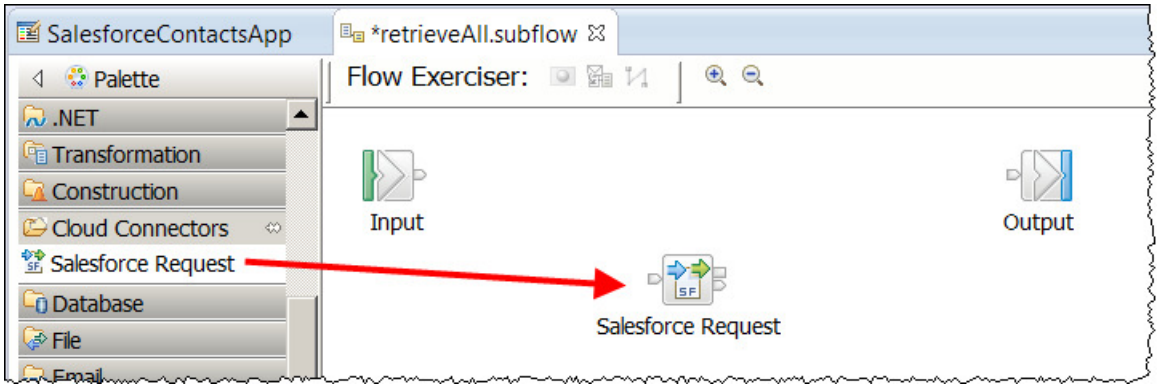
_1. In the retrieveAll operation, click the Create subflow for operation icon:



_2. Confirm the Save REST API message by pressing OK. The Subflow editor will open.

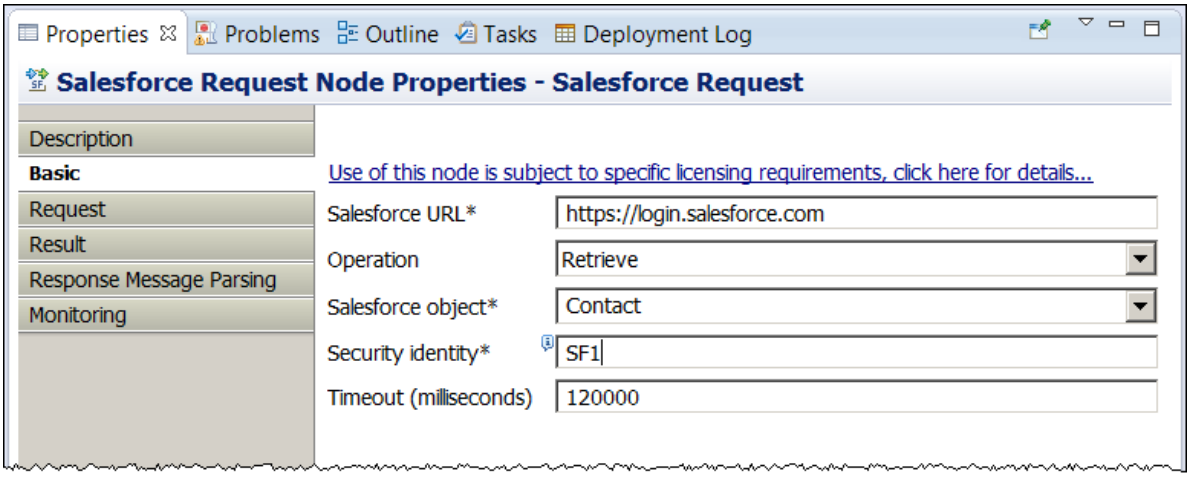
3.4.1 Configure a Salesforce Request node

_1. In retrieveAll.subflow, add a Salesforce Request node (Cloud Connectors) to the subflow:

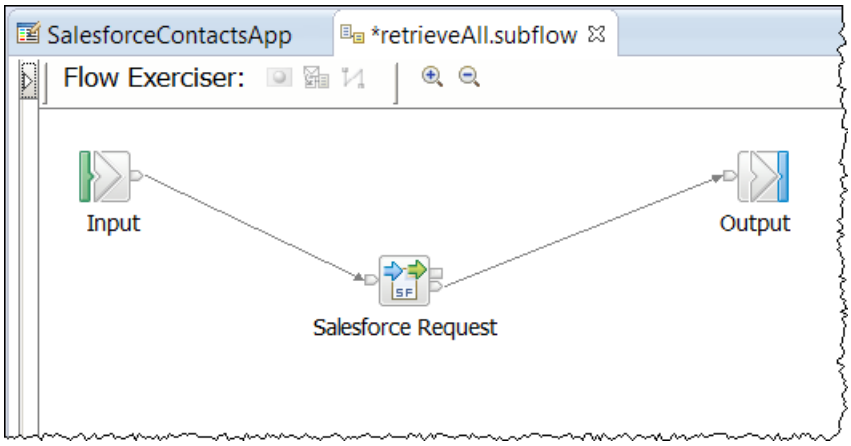


_2. In the basic tab configure the following:

Salesforce URL : `https://login.salesforce.com`
Operation : `Retrieve`
Salesforce object : `Contact`
Security identity : `SF1`



_3. Join the node output terminals in the subflow as follows:



_4. Save and close the subflow.

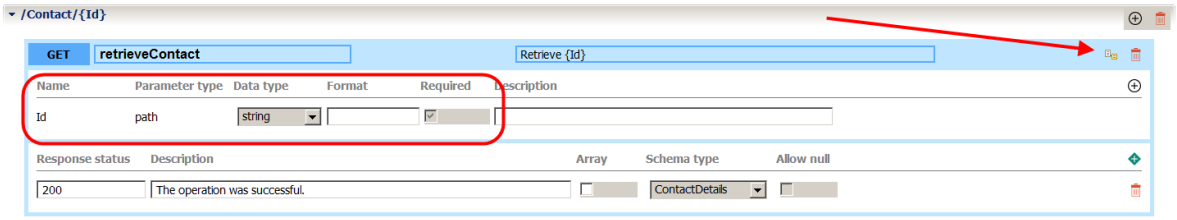
3.5 Configure the retrieveContact Operation

The retrieveContact operation will obtain details of a specific Salesforce contact. The value of the Id will be passed as a parameter in the path of the URL of the request. You will pass this value to the request that is sent to Salesforce by including it in the Salesforce element of the LocalEnvironment message header.

_1. In the list of Resources for the SalesforceContactsApp expand /Contact/{Id} .

Note: "Id" has automatically been added to the operation as a "path" parameter and the Required column has been ticked.

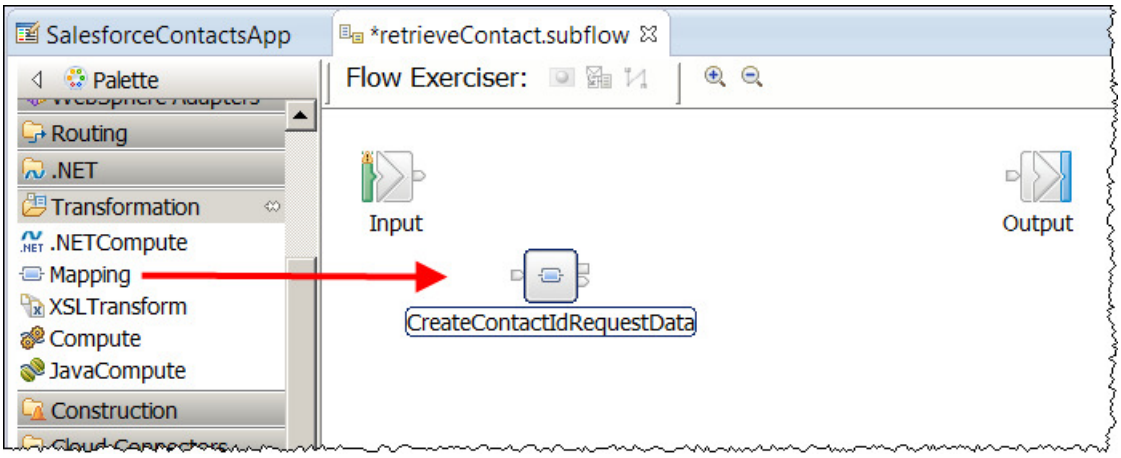
Click the "Create subflow for operation" icon:



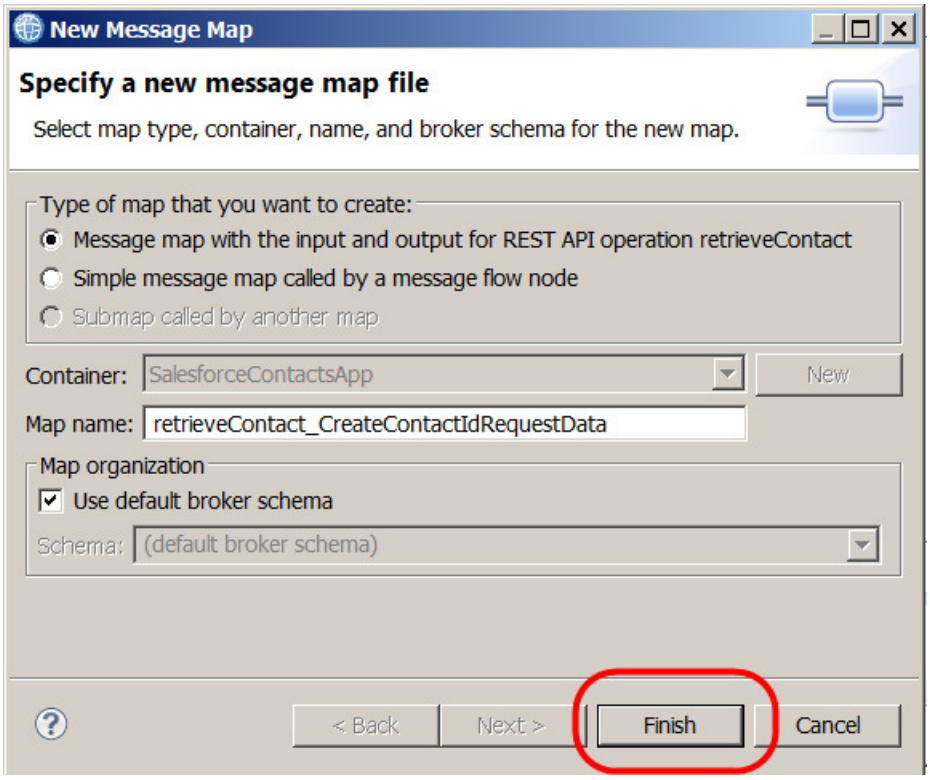
_2. The subflow editor will open.

3.5.1 Configure CreateContactIdRequestData Mapping Node

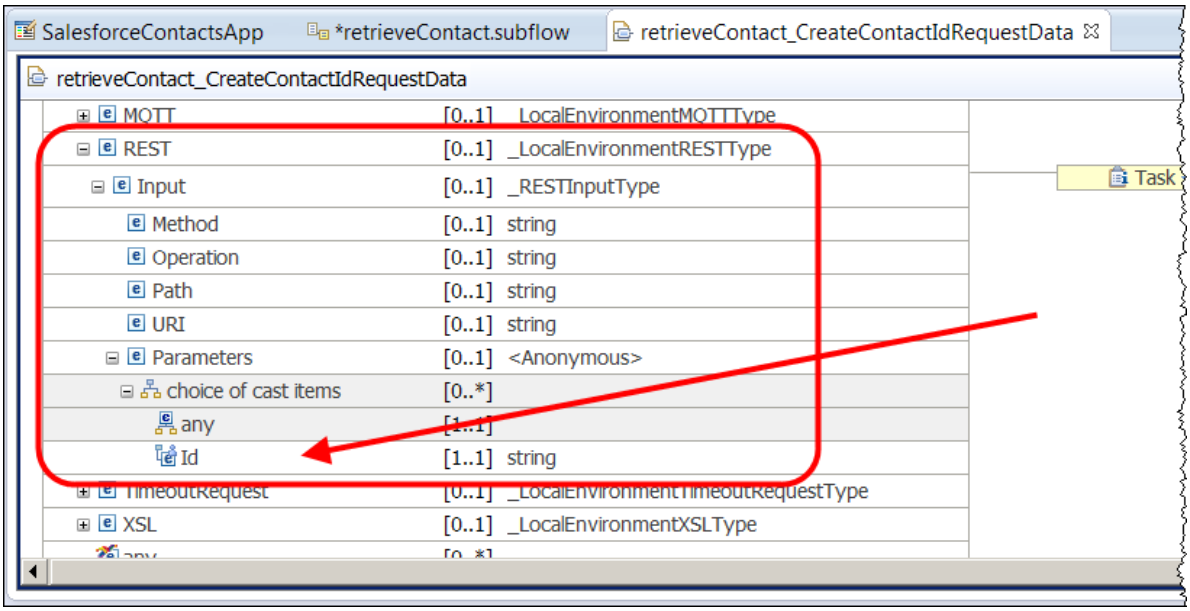
_1. In the Subflow editor add a Mapping node; call the node "`CreateContactIdRequestData`":



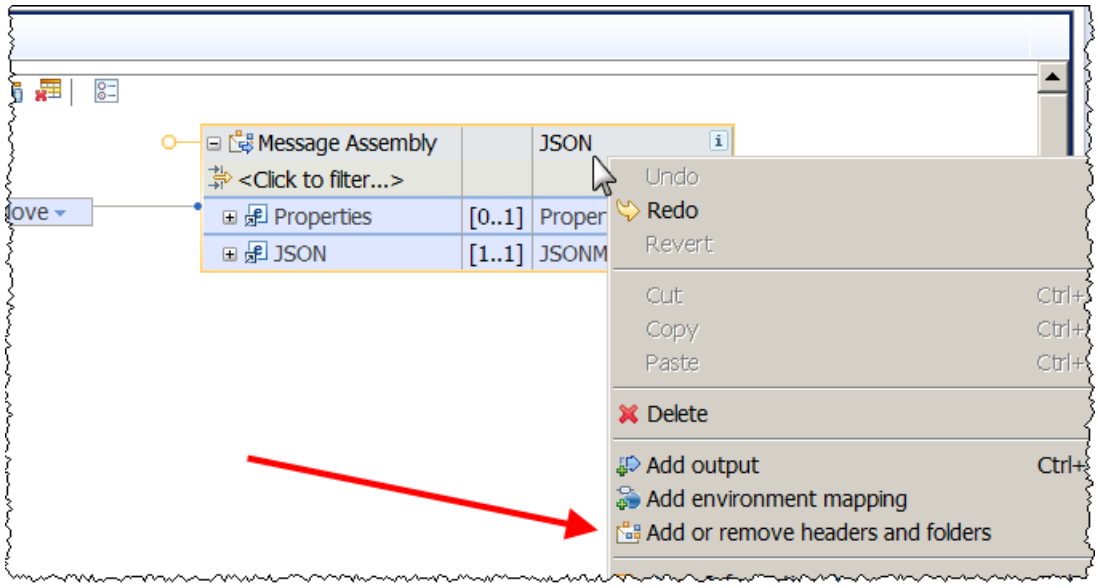
_2. Double click on the map, leave the default settings and click Finish:



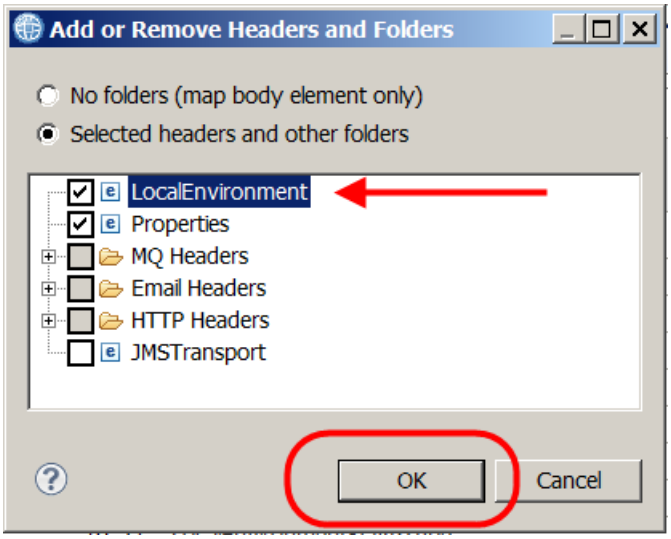
_3. When the mapping editor opens, expand "**LocalEnvironment > REST > Input > Parameters > Choice of cast items**" and note the Id parameter has been automatically added:



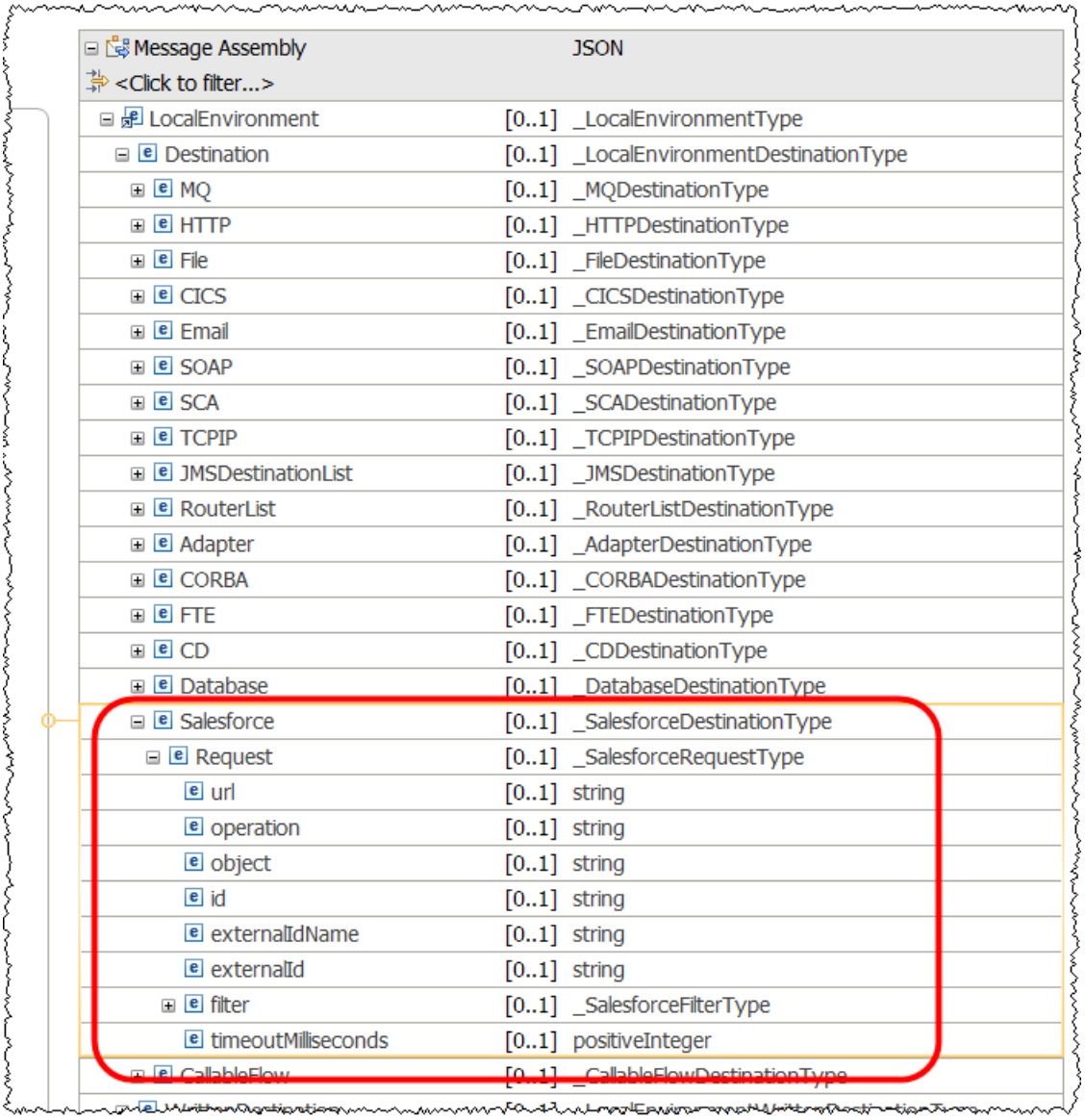
_4. For the Output Message Assembly, right click on Message Assembly “JSON” and select “Add or remove headers and folders”:



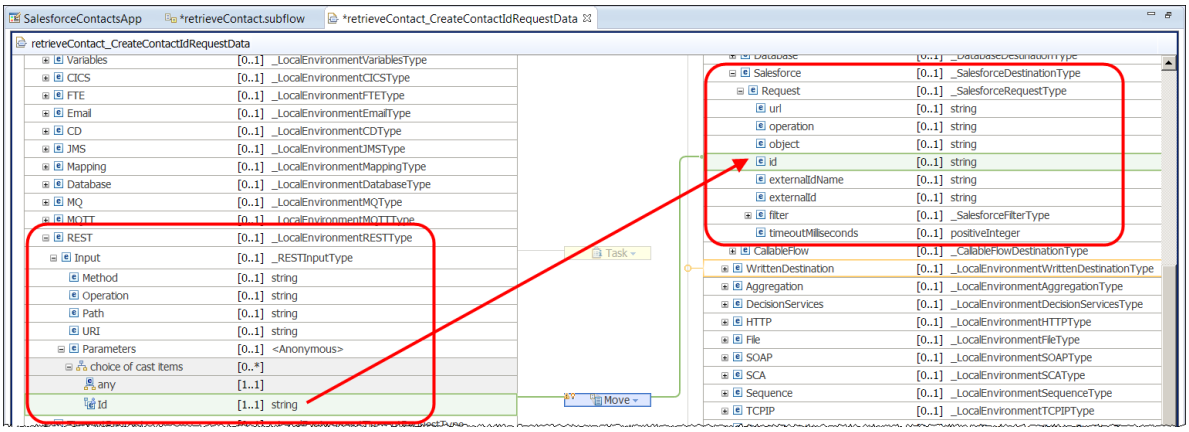
_5. Add LocalEnvironment to the Output Message Assembly:



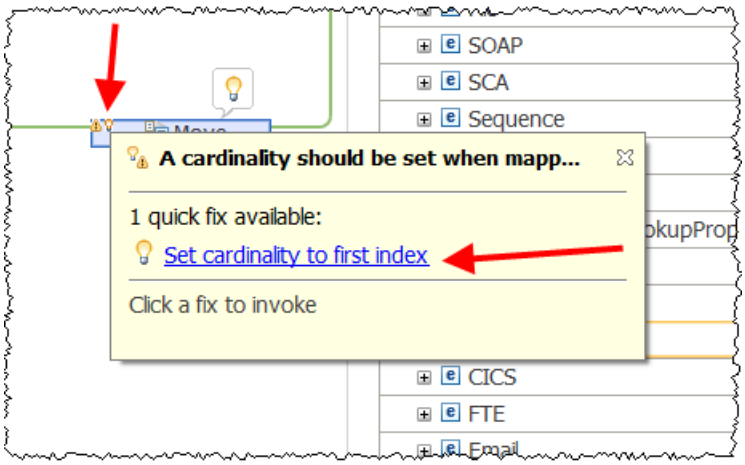
_6. Expand "**LocalEnvironment > Destination > Salesforce > Request**" and note the Salesforce Request elements in the Output Message Assembly:



_7. Map the Id element in the input Message Assembly (`LocalEnvironment > REST > >Input> Parameters > Id `) with the Id element in the output Message Assembly (`LocalEnvironment > Destination > Salesforce > Request > Id`) :



_8. Hover over the small light bulb on the move box and click “Set cardinality to first index” to remove the warning message:



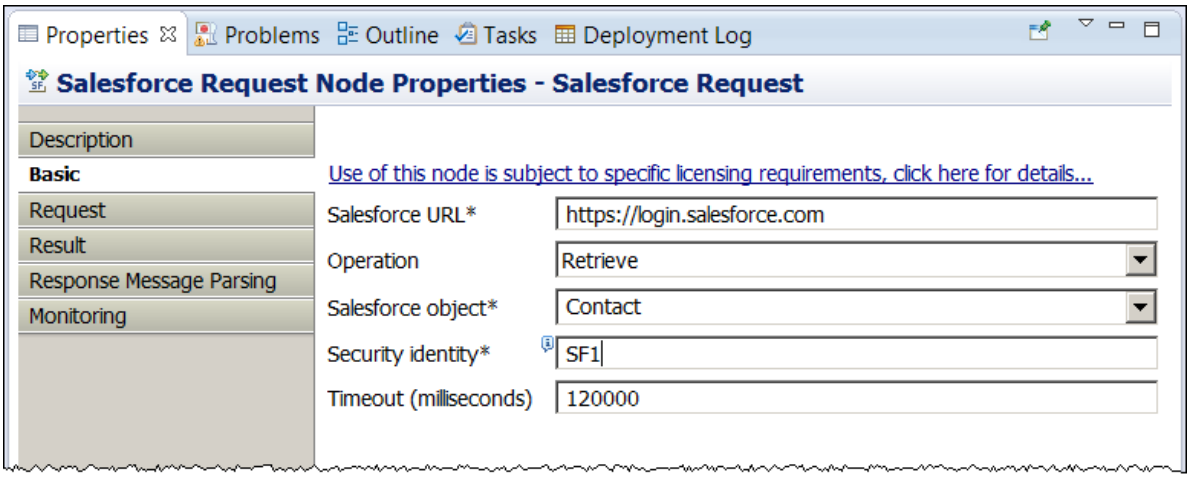
_9. Save and close the Map.

3.5.2 Configure a Salesforce Request node

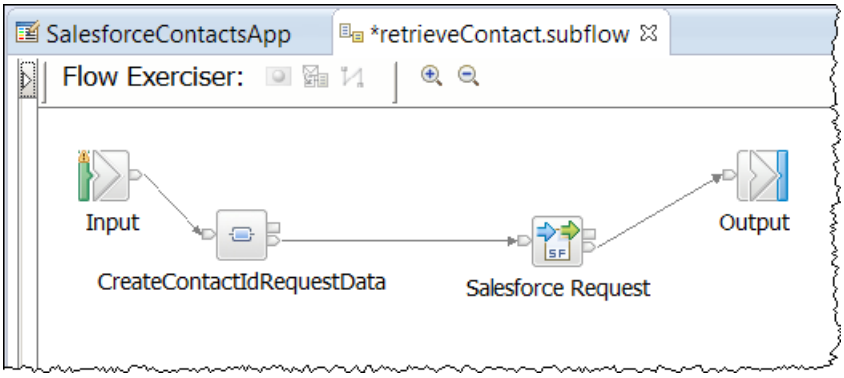
_1. In retrieveContact.subflow, add a Salesforce Request node (Cloud Connectors) to the subflow:

_2. In the basic tab configure the following:

Salesforce URL : `https://login.salesforce.com`
Operation : `Retrieve`
Salesforce object : `Contact`
Security identity : `SF1`



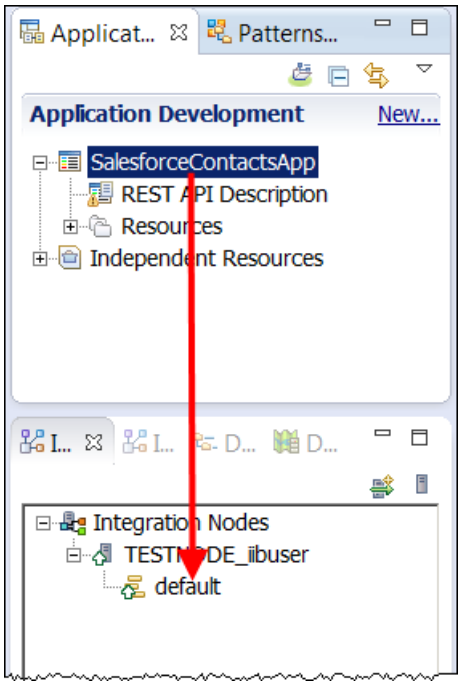
_3. Join the node output terminals in the subflow as follows:



_4. Save and close the subflow.

4. Deploy the Salesforce REST API

_1. Deploy the Salesforce REST API to the default Integration Server.



5. Test the insertContact Operation

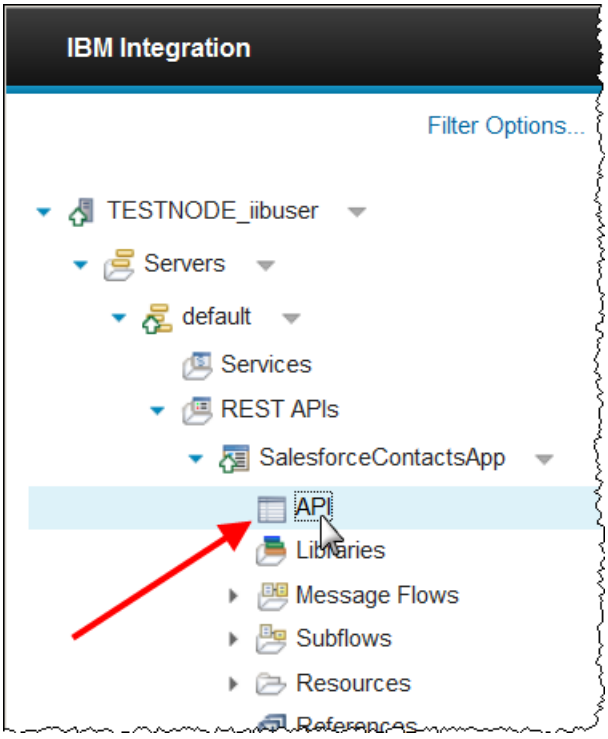
You are now ready to test the Salesforce “Contacts” REST API. You will see from testing the REST API that it will fail when accessing Salesforce.com for security reasons. You will then perform the necessary configuration for it to successfully work.

_1. In a browser open the Swagger UI tool. In the Firefox browser on the IIB class VMware image, this URL is booked marked in the “REST” folder. It will default to the Swagger Petstore application.

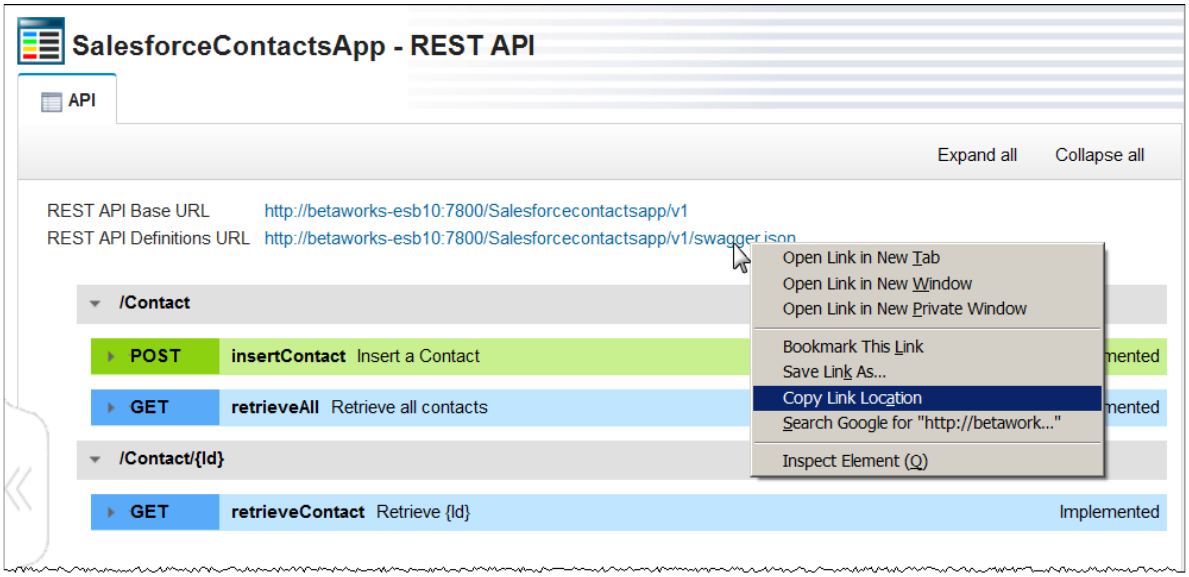
_2. In the Firefox browser, open another tab. Open the IIB WebAdmin UI for TESTNODE_iibuser (in IIB Nodes folder).

Copy these details and paste them into the SwaggerUI.

_3. In the left nav bar, navigate to the API section of the SalesforceContactsApp and click API:

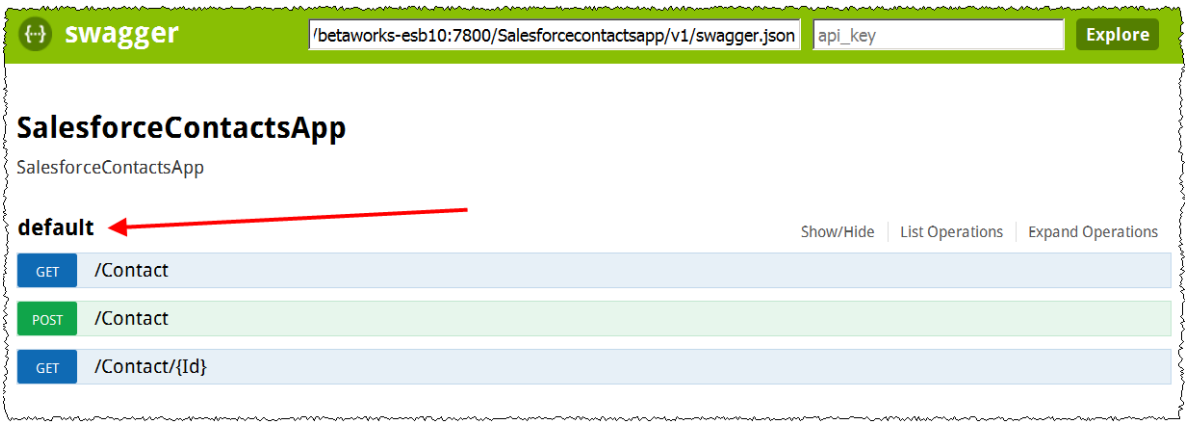


_4. Right click on the **REST API Definitions URL** and select “Copy link location”



_5. Switch to the SwaggerUI tab and paste the URL into the swagger URL field and press enter.

_6. Click on the word “default” to expose the Resources you created in the REST API:

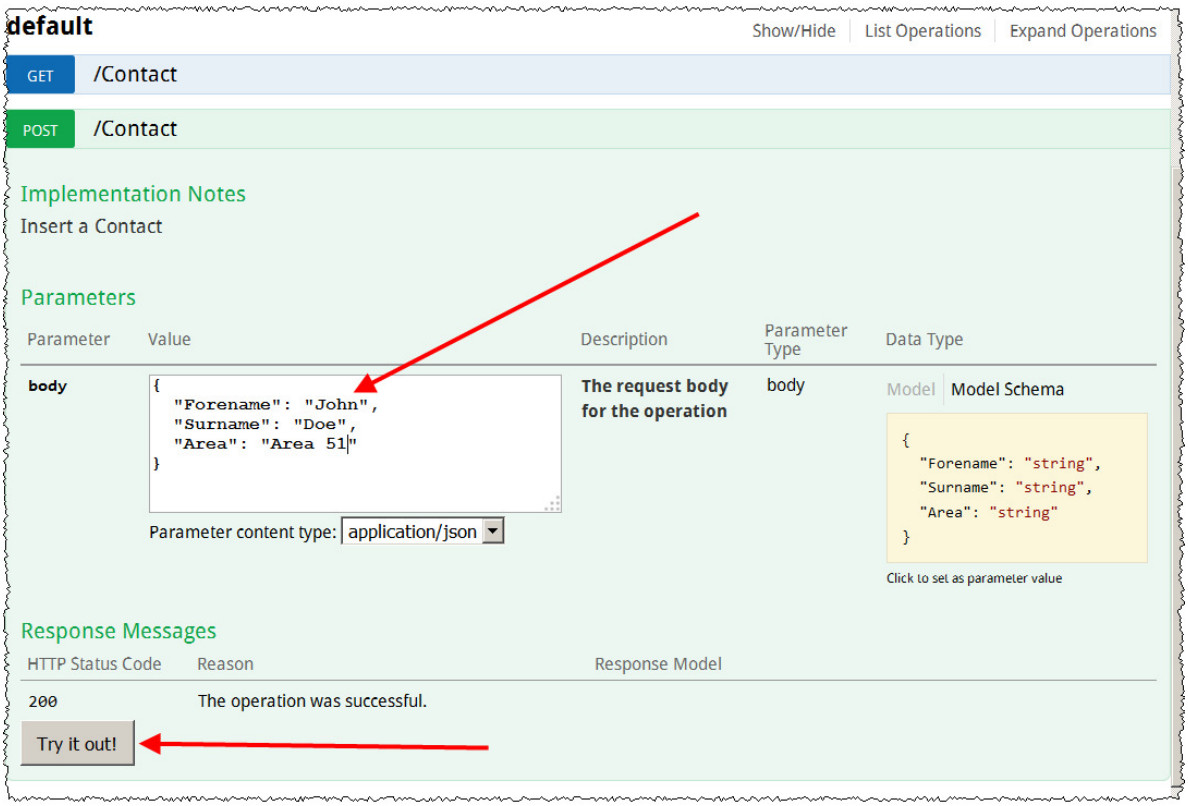


_7. Click the POST associated with the /Contact resource.

Click on “Forename” in the model Schema, this will copy the model schema definition into the value of the body parameter.

Replace the three occurrences of “string” with “John”, “Doe”, “Area 51”.

Click the “Try it out!” button:



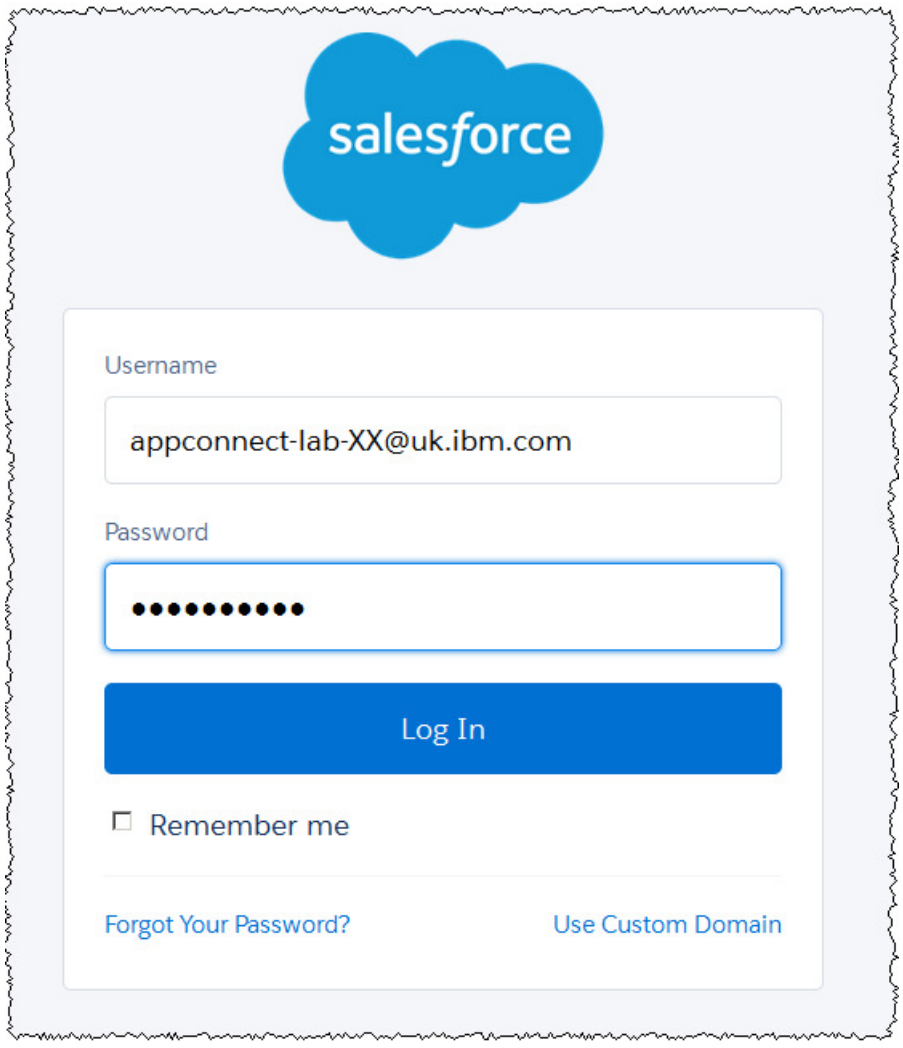
_8. After a few seconds you will see the request will have failed with a Response code 500. The Response Body will show the error: ”

```
BIP3858E: ( TESTNODE_iibuser.default ) The SalesforceRequest node
received error code ''Error'' while performing the ''Create'' operation.
Error: ''LOGIN_MUST_USE_SECURITY_TOKEN: Invalid username, password,
security token; or user locked out. Are you at a new location? When
accessing Salesforce--either via a desktop client or the API--from
outside of your company's trusted networks, you must add a security token
to your password to log in. To get your new security token, log in to
Salesforce. From your personal settings, enter Reset My Security Token in
the Quick Find box, then select Reset My Security Token.''. [22/03/2016
22:16:39]
```

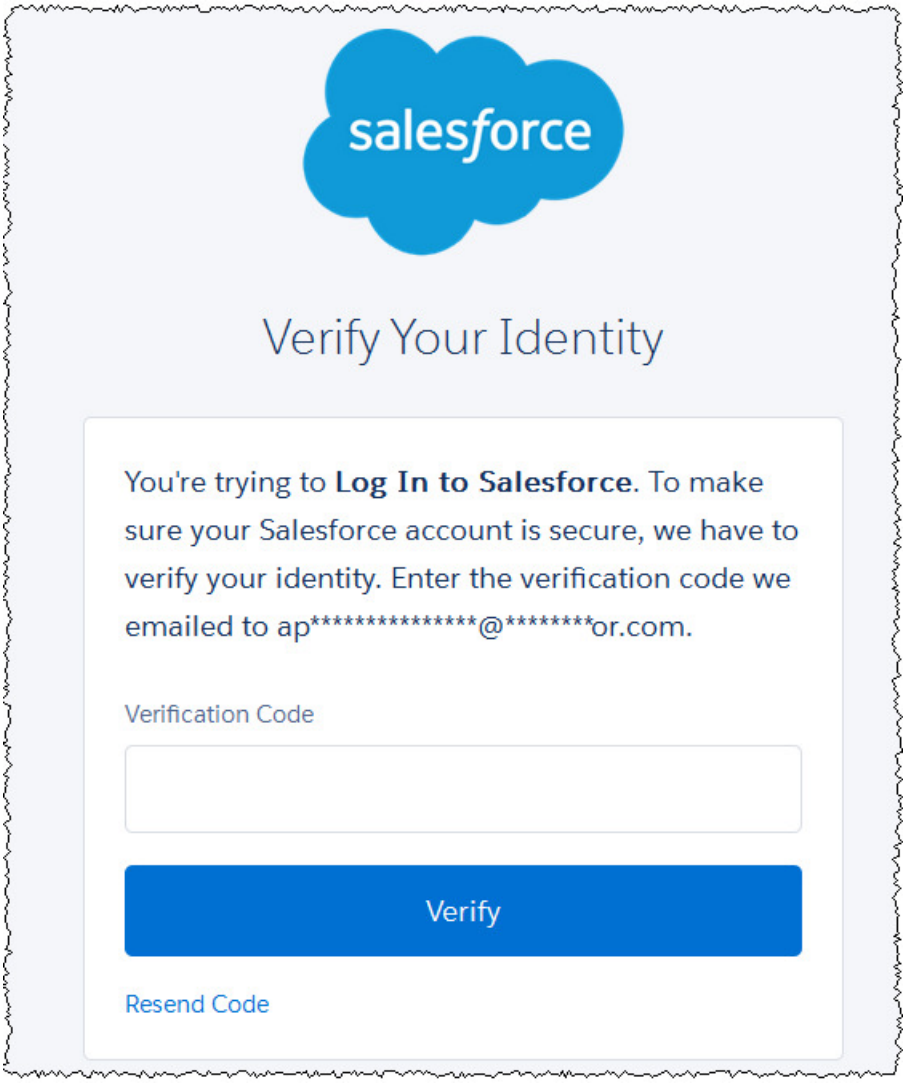
5.1 Set the Security Token in Salesforce

_1. Using your Salesforce id, logon to the Salesforce.com Web UI at https://login.salesforce.com/

Note: if you are using the Salesforce ids supplied during the IIB Workshop these values are "`appconnect-lab-XX@uk.ibm.com`" (XX from 01 to 40) with a password of "`appc0nn3ct`"

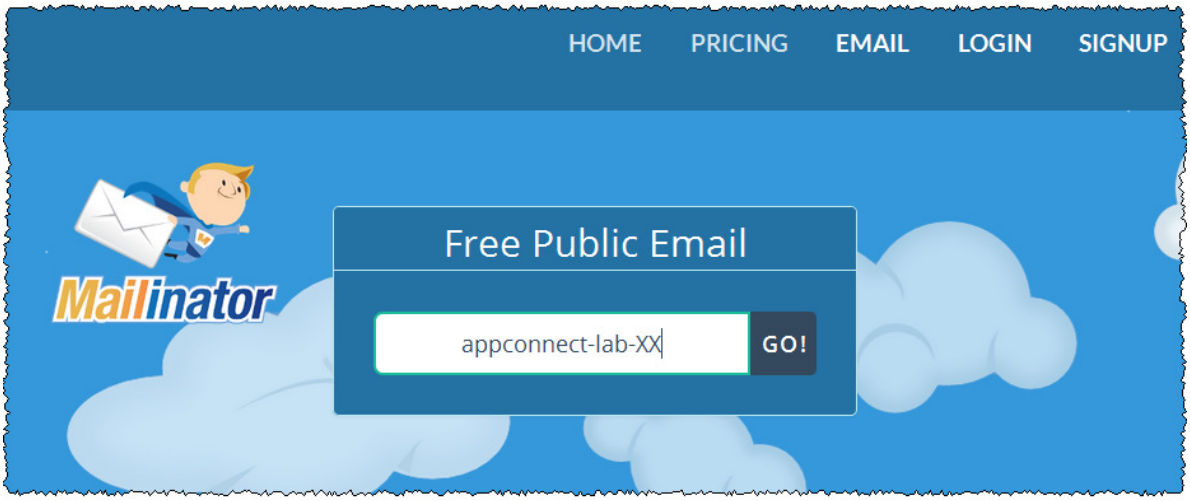


_2. You will see that Salesforce requires you to verify your identity:

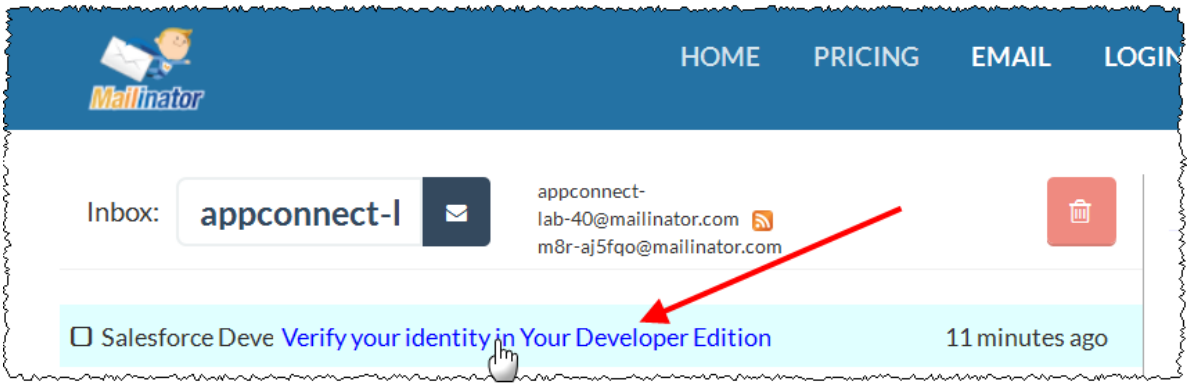


_3. All the Salesforce ids being used in the IIB class have been set up to use “mailinator.com”. Open a Firefox tab and go to mailinator.com.

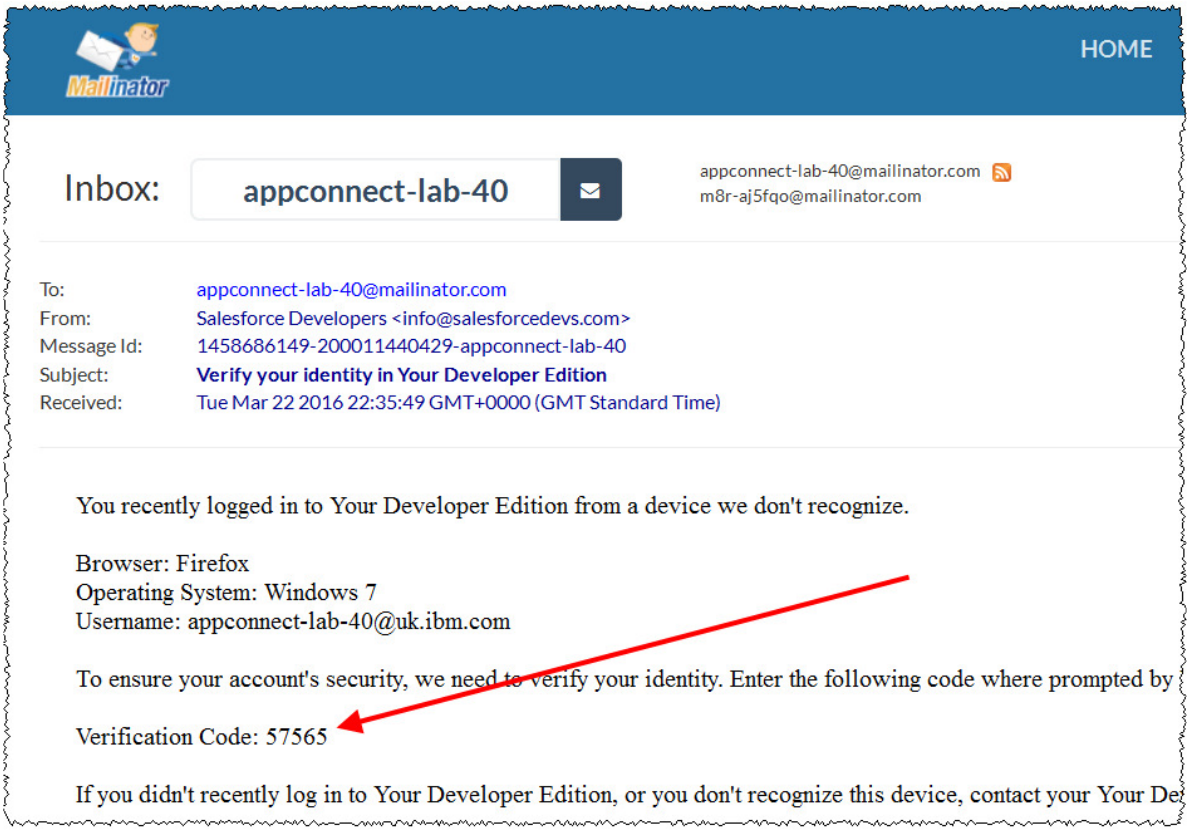
Type your ID in the field in the space provided (replace the XX with your number) and press GO!



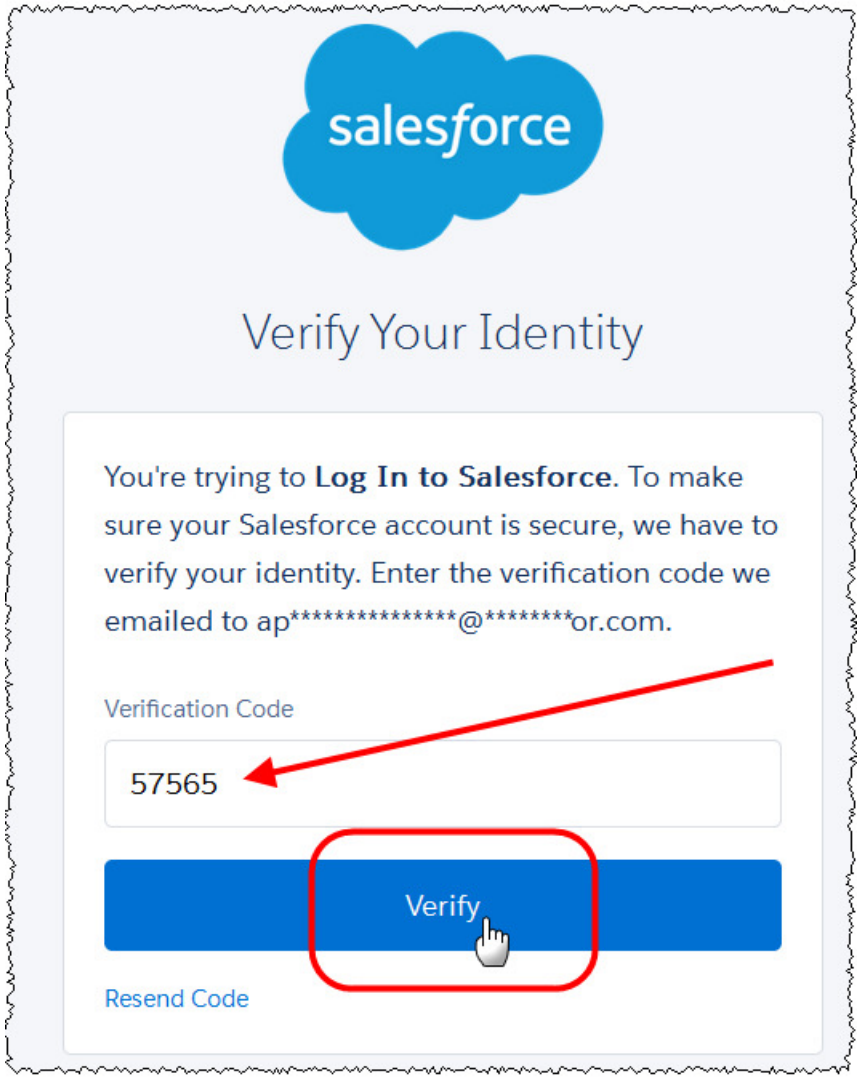
_4. Click the link showing that your ID has an email from the Salesforce Development team:



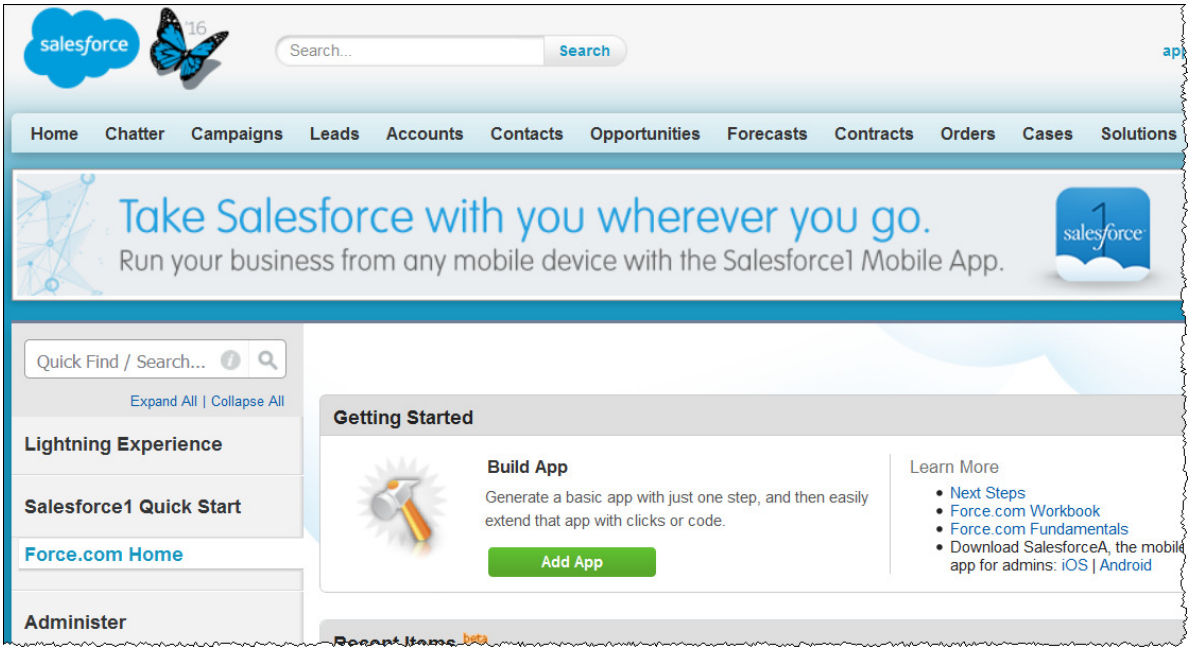
_5. The email will show you the Verification Code you need to enter on the login screen:



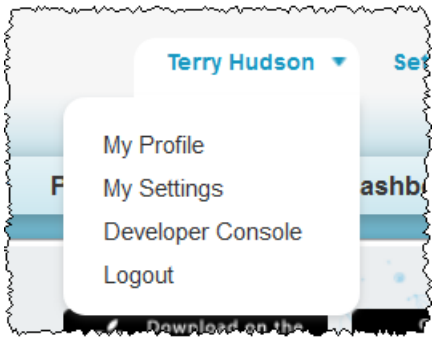
_6. Switch back to the Salesforce web UI and enter the Verification Code:



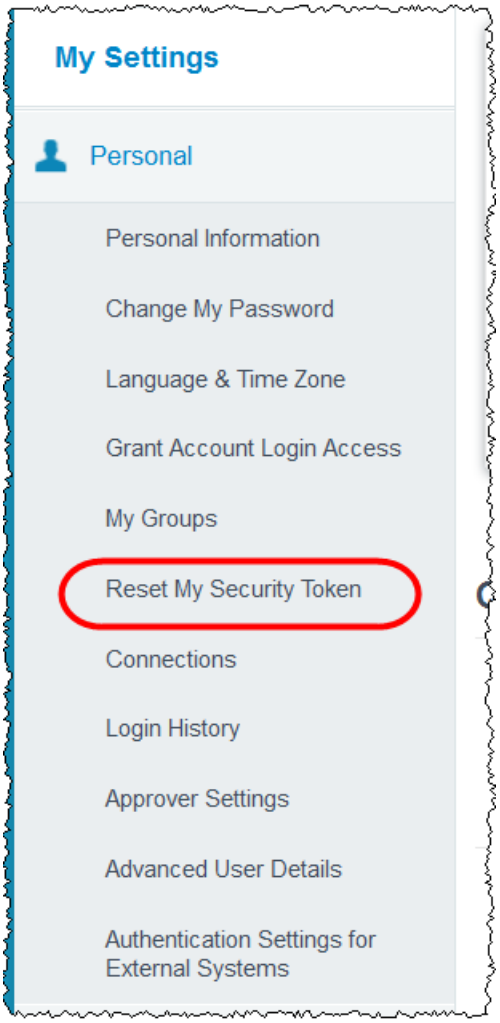
_7. You should see the Salesforce.com Home page:



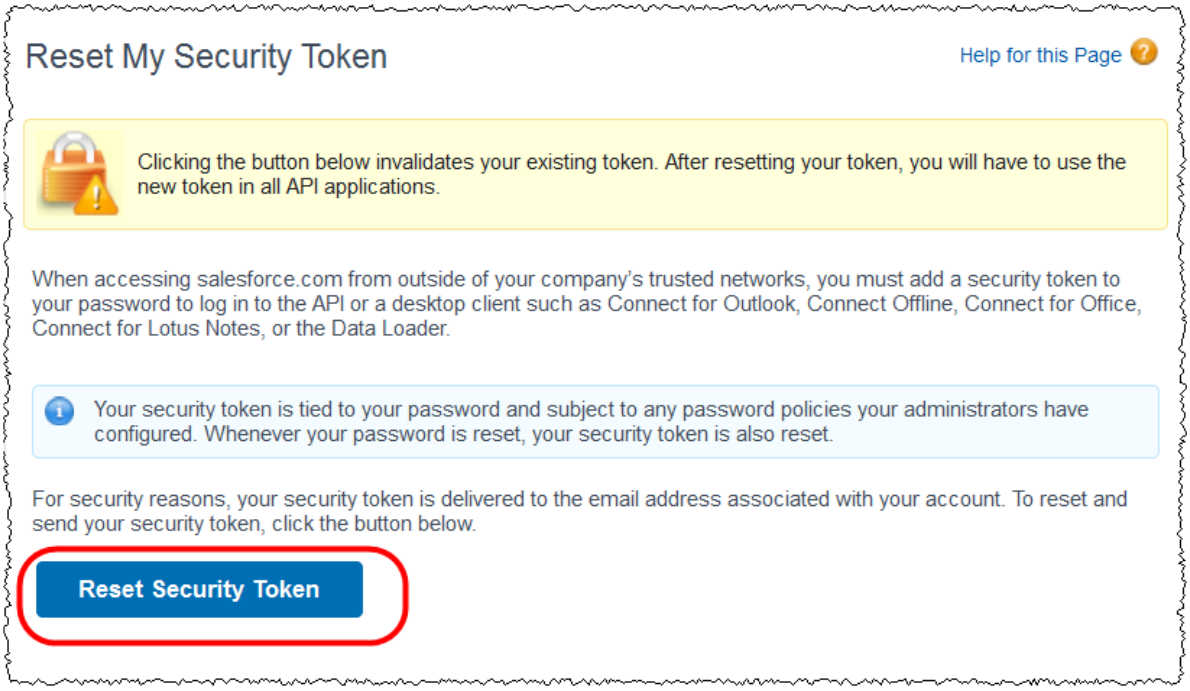
_8. Go to “My Settings”:



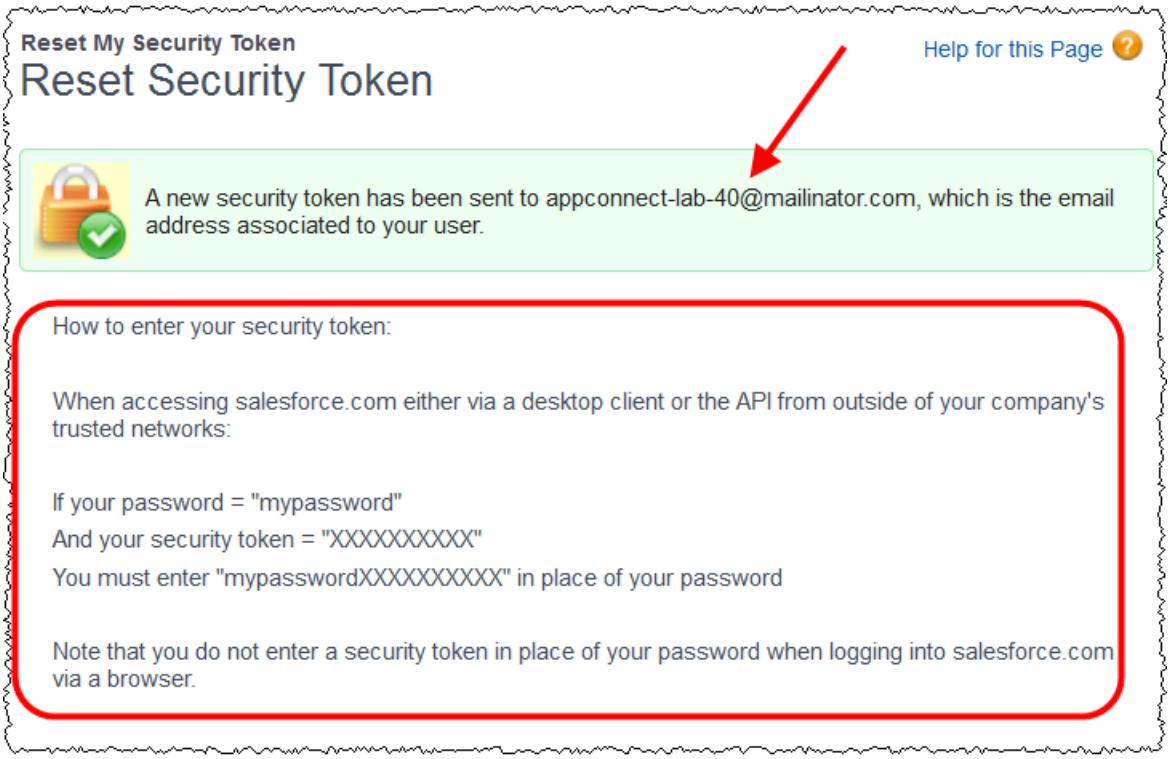
_9. Expand Personal and select “Reset My Security Token”:



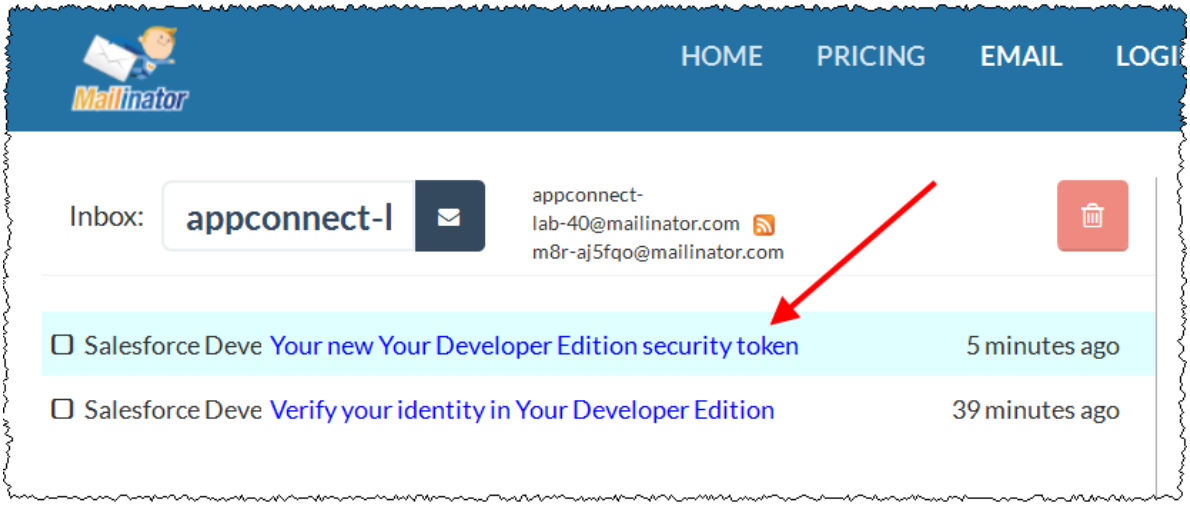
_10 Click “Reset Security Token”



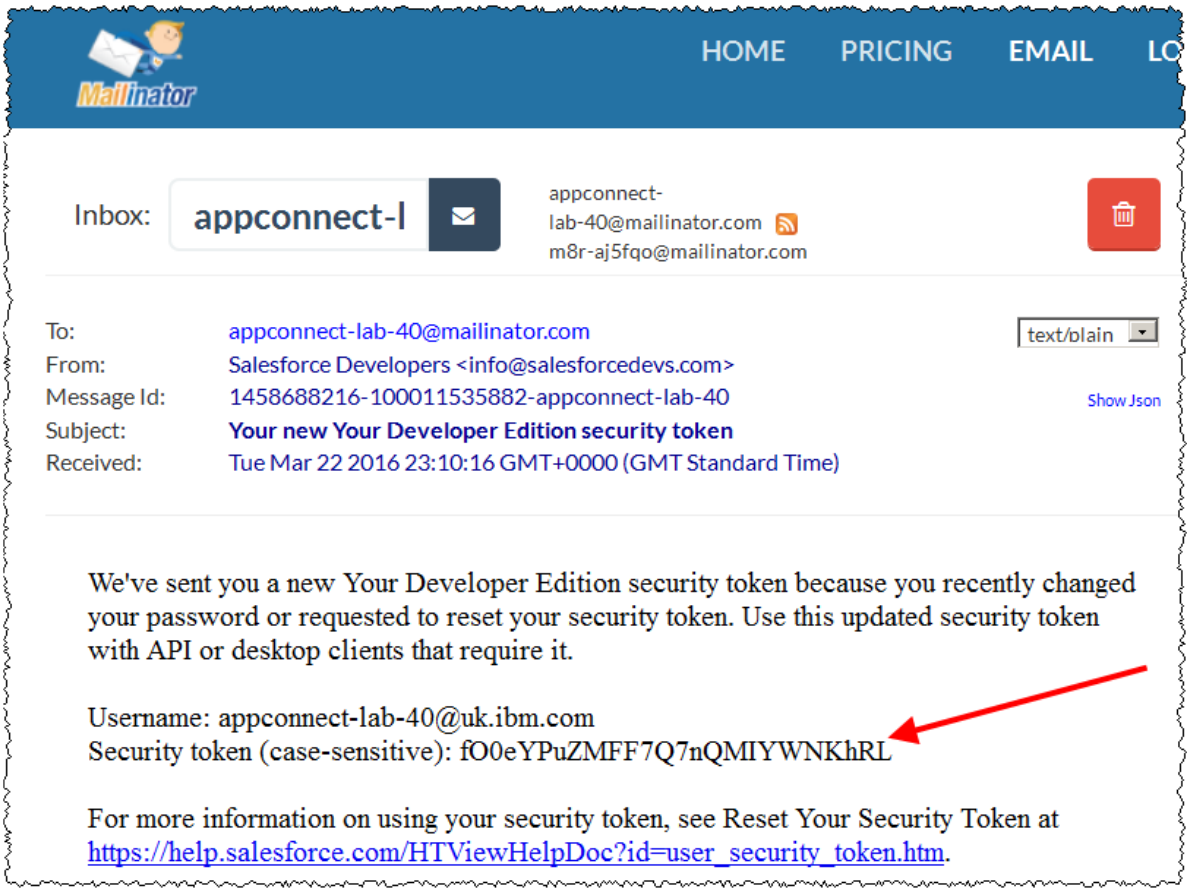
_11 Note the reset has triggered the security token to be sent to the mailinator.com id you are using (in this case #40):



_12 Switch back to the mailinator.com tab and review the second email sent from Salesforce.com (use the back button to see the new email:



_13 Open the email (click on the link) and copy the (case sensitive) security token:



_14 In an Integration Console, enter the command:

```
mqsisetdbparms TESTNODE_iibuser
               -n salesforce::SF1
               -u appconnect-lab-XX@uk.ibm.com
               -p appc0nn3ctfO0eYPuZMFF7Q7nQMIYWNKhRL
```

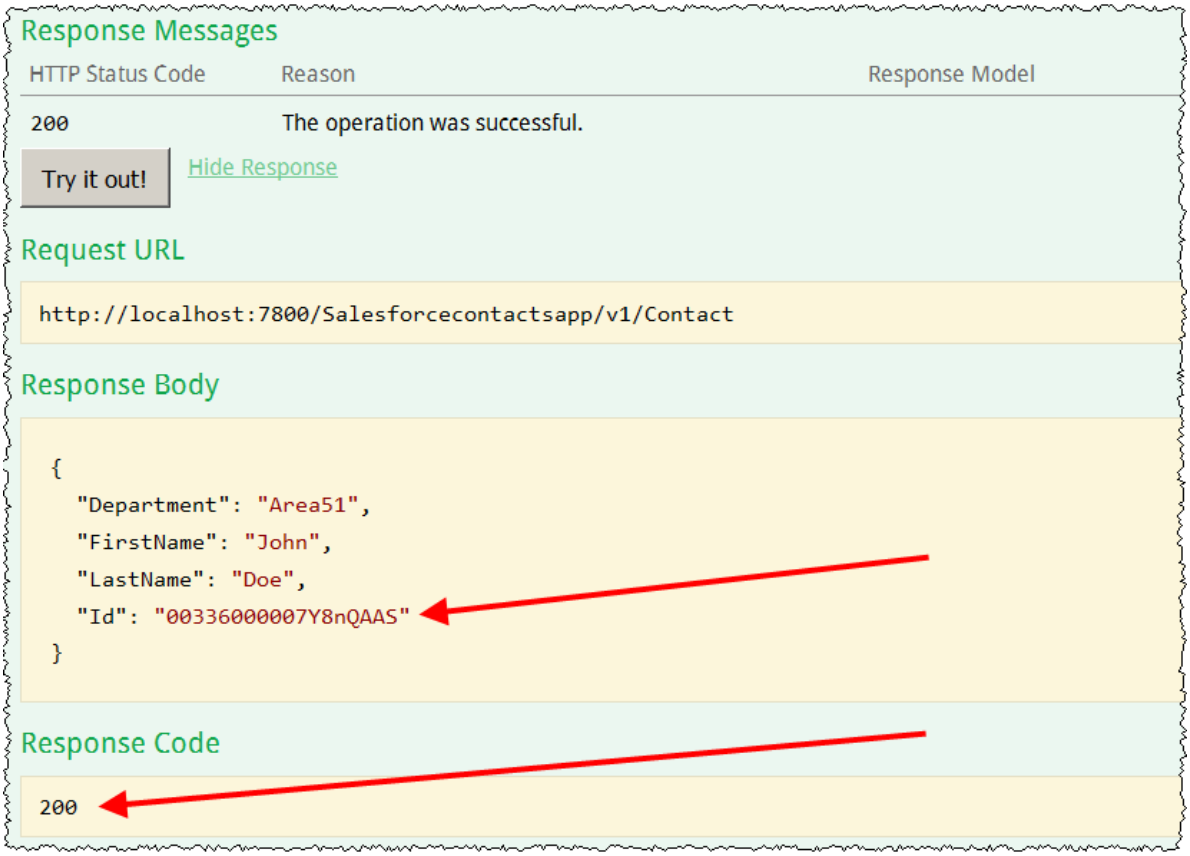
Note replace:
XX with your Salesforce Id number
fO0eYPuZMFF7Q7nQMIYWNKhRL with the Security token sent from Salesforce.com for your Id

_15 Stop and restart the default Integration Server for the changes to take effect.

5.2 Retest the insertContact Operation

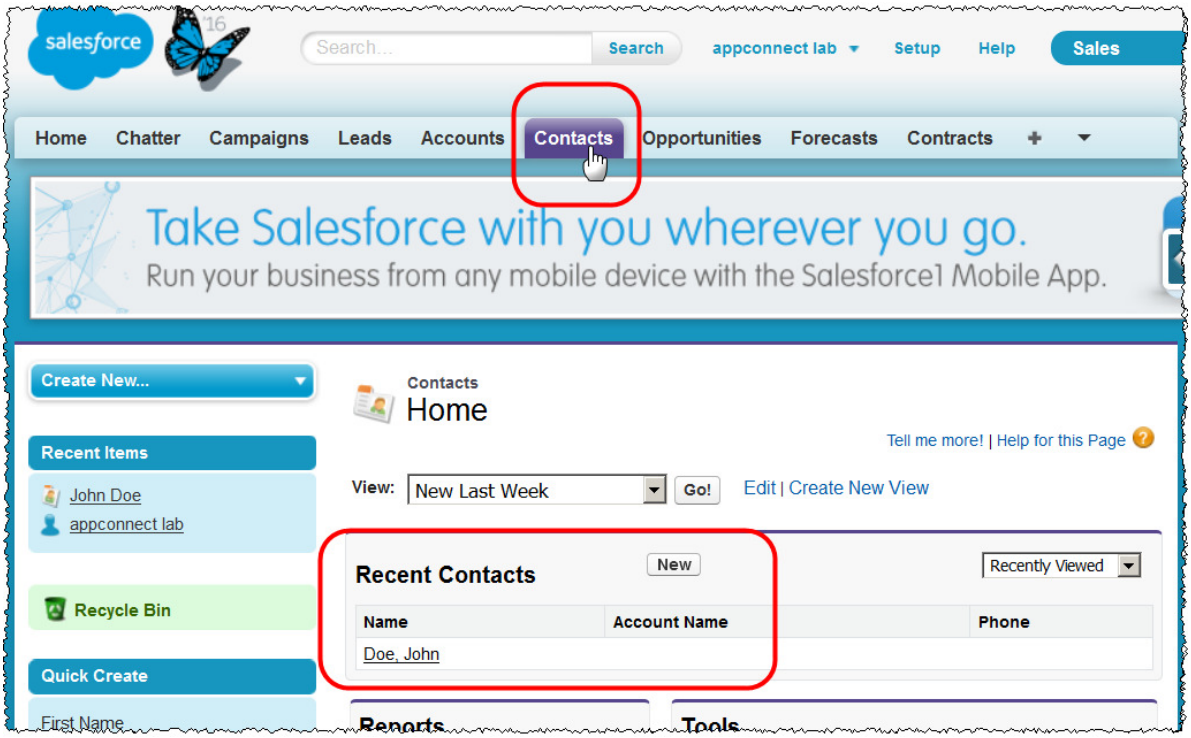
_1. Switch back to the SwaggerUI in the browser and press the “Try it out!” button again.

_2. This time the request should complete successfully with a response code 200 and the Response body should pass back an Id value from Salesforce.com:



5.3 Verify the Contact in Salesforce.com

_1. Switch back to the Salesforce.com web UI. Click Contacts from the Main Menu to see the User that was added:

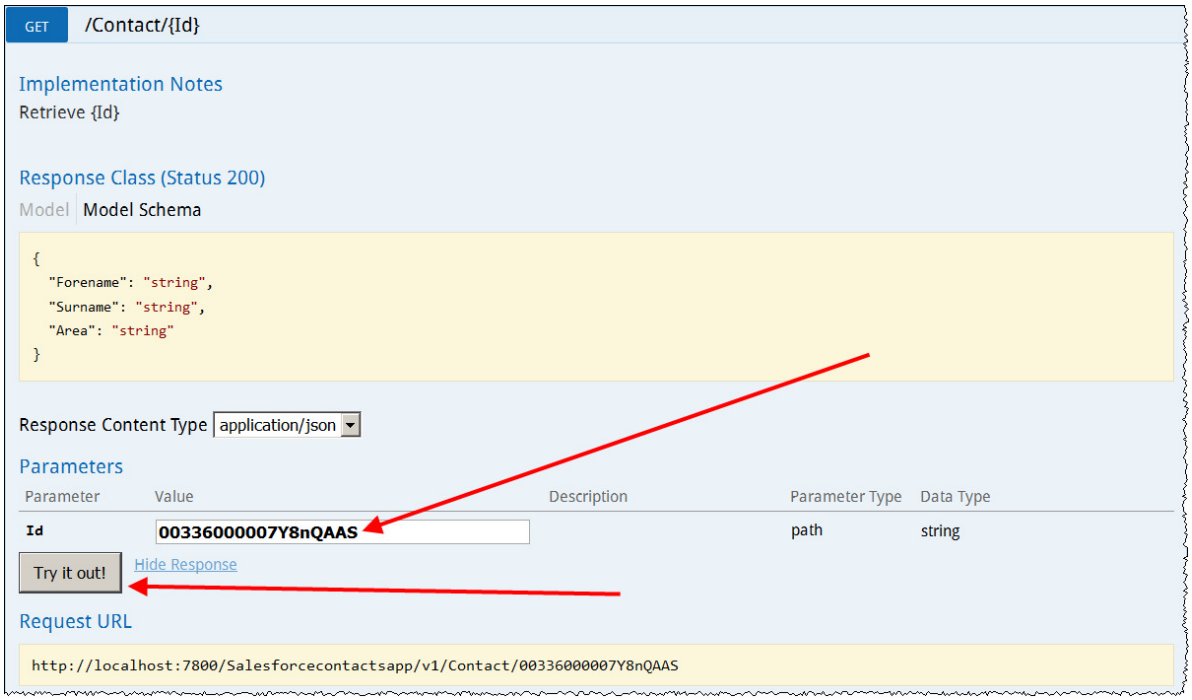


Now that you have been able to successfully connect to Salesforce.com and add a Contact, the other operations that you have configured in this lab guide should also work successfully. The Salesforce verification and Security Token setup should not need to be configured again.

6. Test the retrieveContact Operation

_1. In Swagger UI, copy the value of the “Id” passed back from Salesforce.com for the user you just added.

_2. Navigate to the retrieveContact operation and paste the Id value passed back from Salesforce into the value field for the Id parameter in retrieveContact. Click “Try it out!”:

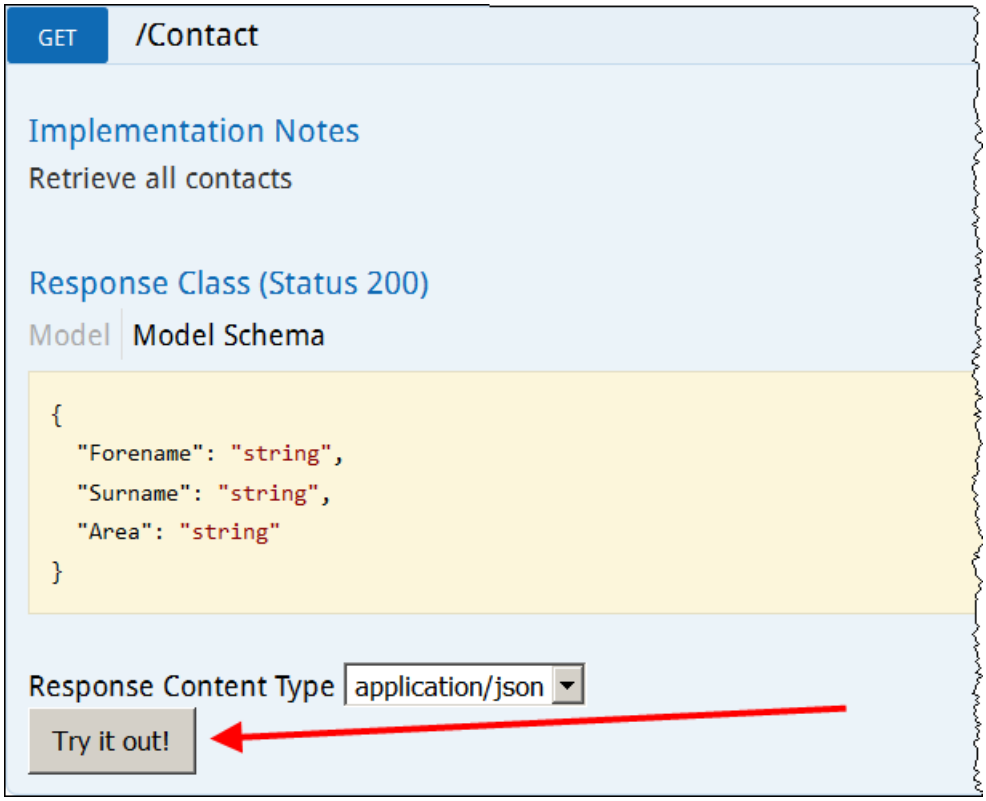


_3. After a few seconds the request should complete with Response Code 200. The Response Body should pass back details of the user you added in the response body:

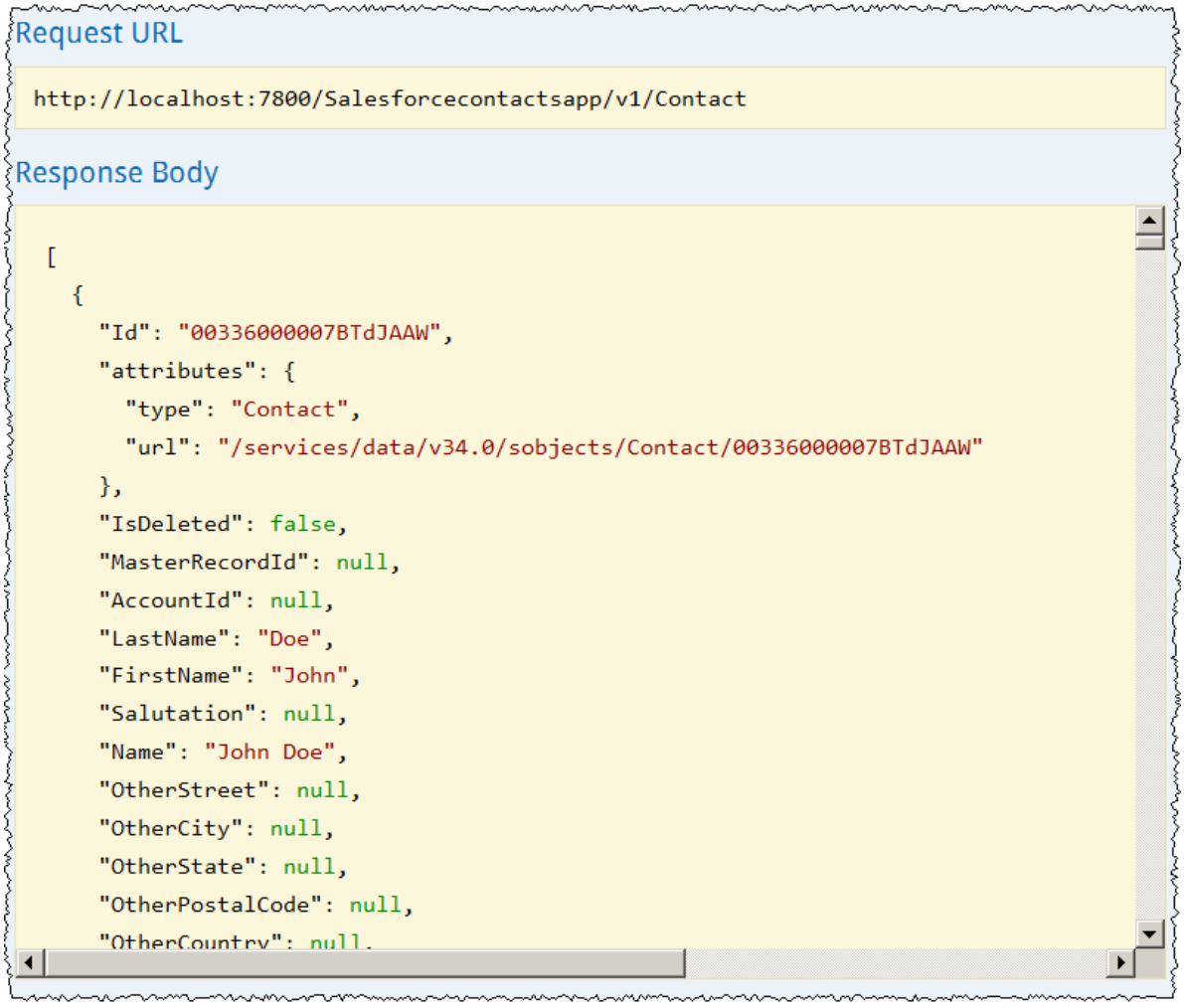
7. Test the retrieveAll Operation

_1. In Swagger UI, collapse all the operations and expand the GET operation associated with the /Contact resource:

Click Try it out!:



_2. After a few seconds the request should complete with Response Code 200. The Response Body will contain all Contacts that your Salesforce user has access to:



END OF LAB GUIDE