IBM Integration Bus

IBM Integration Bus on Cloud Connecting to local MQ and DB2 Systems using Endpoint Connectors

Featuring:

- REST API
- Using a REST API Description in a Shared Library
- Creating a generic map using JSON Models
- MQ and DB2 Endpoint Connectivity
- Using IIB on Could with Secure Connector Agents
- Testing using IIB Flow Exerciser and Swagger UI

September 2016

Hands-on lab built at product Version 10.0.0.6

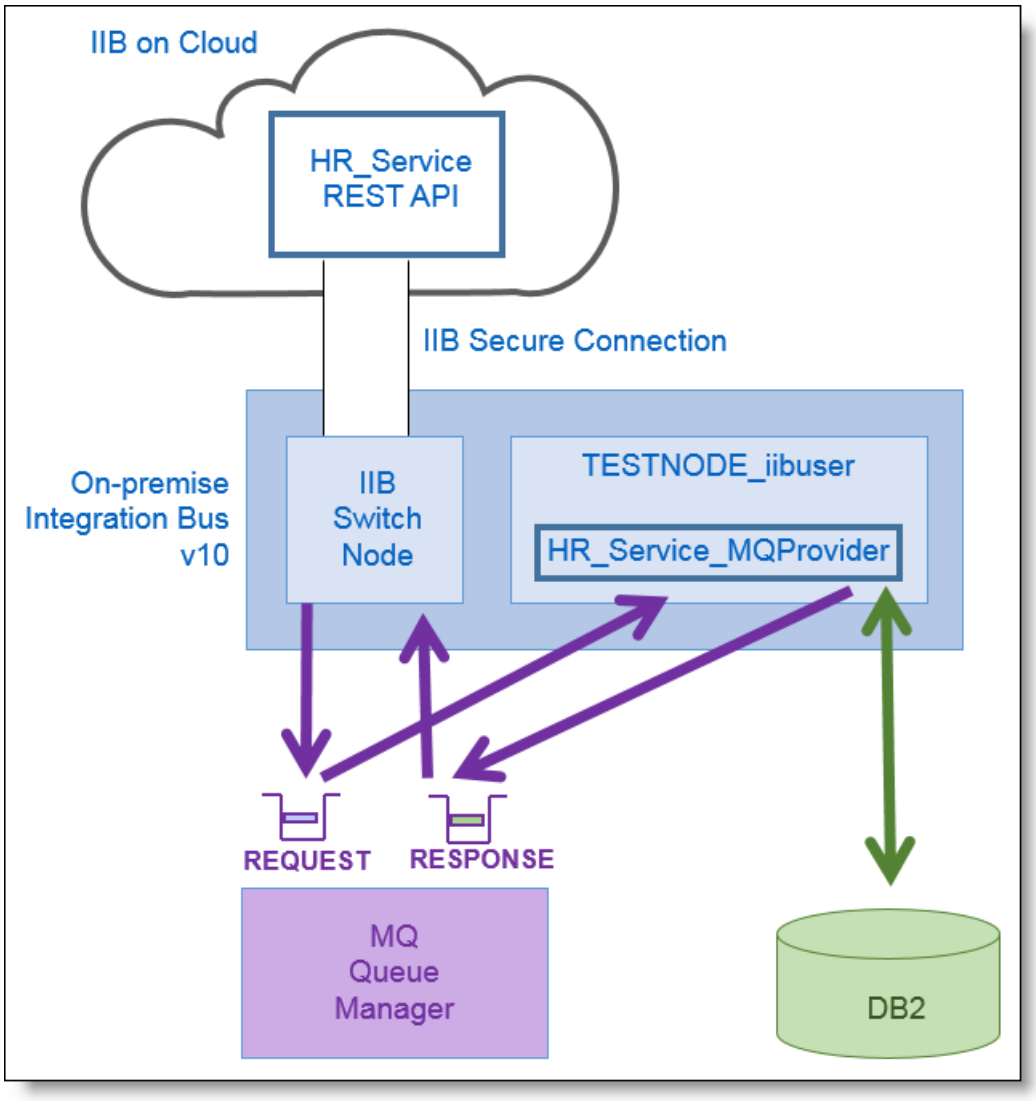
1. Introduction

1.1 Scenario Overview

In this lab guide you will develop a scenario to show how a REST API (created using IBM Integration Bus) can run on IIB on Cloud and access local (on premise) MQ and DB2 systems.

Whilst developing this scenario the lab guide will demonstrate how json models created as part of process of defining the REST API can also be re-used by mapping nodes defined in a Shared Library. When a REST API description is imported into a Shared Library, any graphical data mapping node also created and stored within the Shared Library has access to the json models defined within the imported REST API Description. This enables applications to easily process data produced from (or intended for) a REST API.

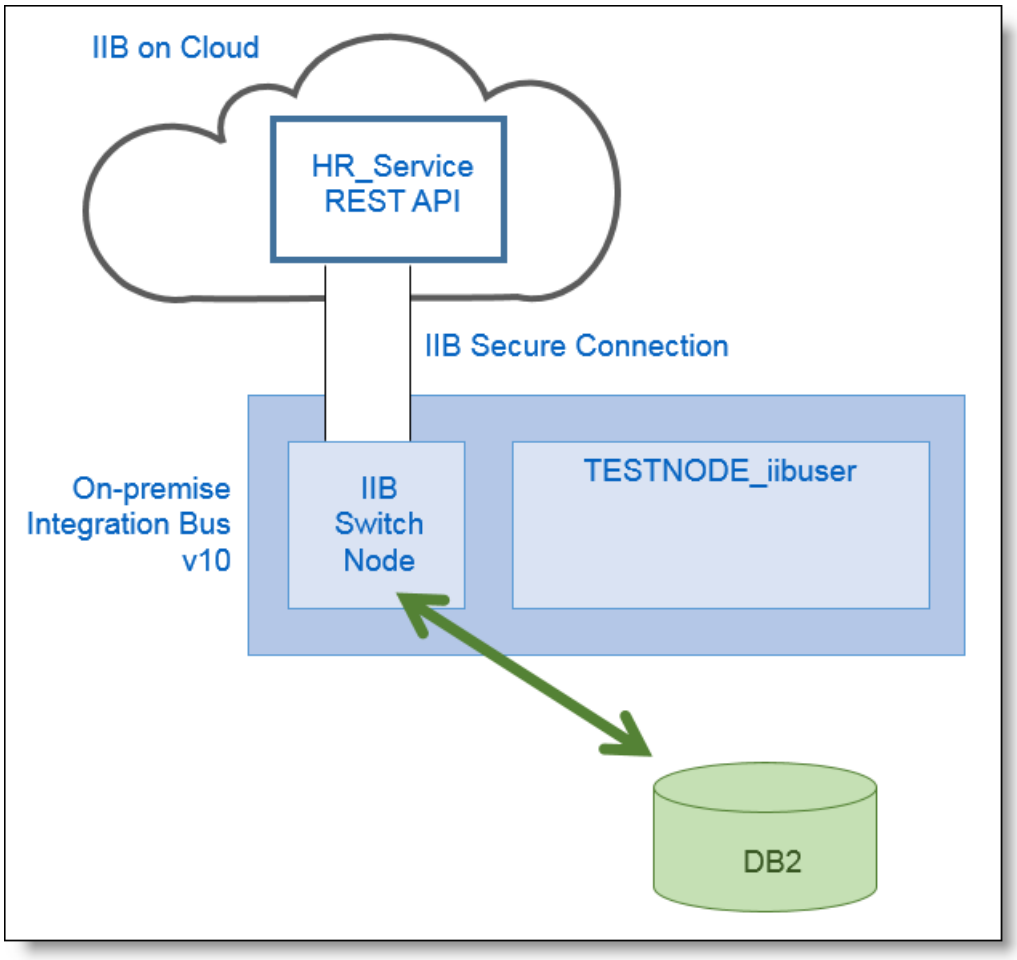
1. The MQ Scenario:



You will import and review a REST API solution "HR_Service" into your workspace. The MQ scenario has already been implemented using a resource **/employees/{employeeNumber}/MQEndpoint**. After reviewing this implemented solution, you will then import the REST API Description used by this REST API into a Shared Library and create a map that an MQ Application (created as an IIB Application) can use to process data (requests) from and send data (responses) to the REST API. Local MQ queues will be used to pass requests and responses between the HR_Service REST API and the MQ application. Once working correctly on your local system, the HR_Service REST API will

be deployed to IIB on Cloud where you will configure MQ Endpoints and a Secure connector to enable IIB on Cloud to communicate with your local MQ system.

2. The DB2 (ODBC) scenario:



You will review the implementation of an additional resource in HR_Service, **/employees/{employeeNumber}/cloudODBC**. The implementation uses ESQL in a Compute node to access your local DB2 HRDB database directly from IIB on Cloud. A DB2 Endpoint is configured on IIB on Cloud to enable this resource to access the HRDB DB2 database using the same Secure Connector configured for the MQ scenario.

2. Initial Environment Preparation

Important note

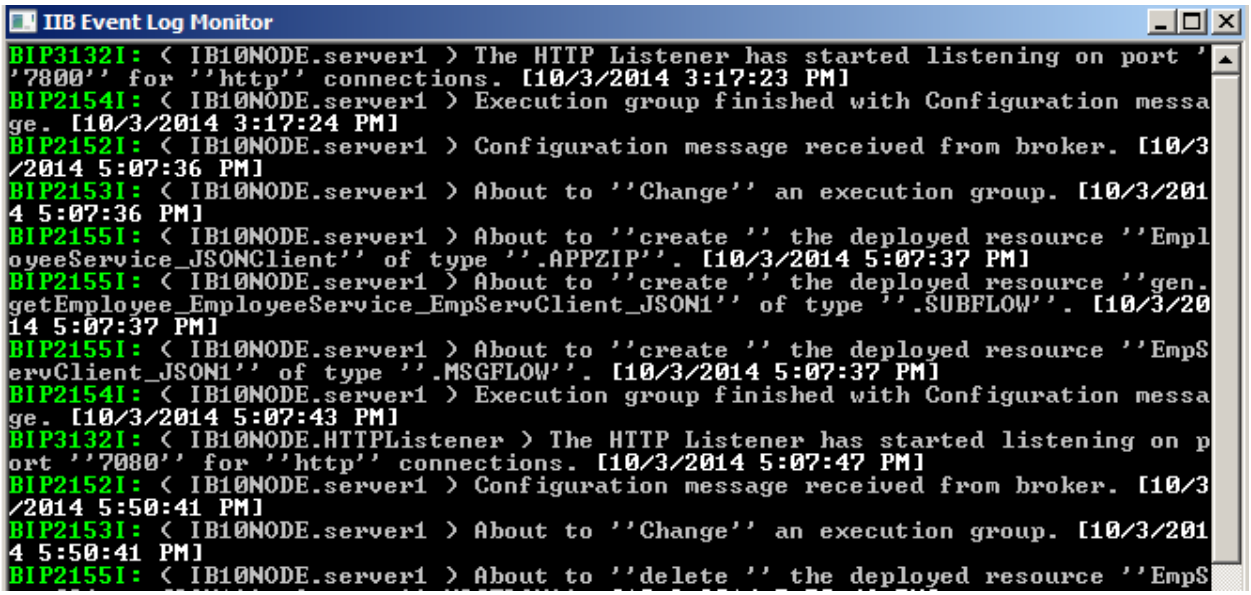
This lab will require an 'IIB on Cloud' account. For the lab you can use the Free Trial version, which you can request at *http://www-03.ibm.com/software/products/en/ibm-integration-bus-on-cloud*.

You should use the Windows user "iibuser". This user is a member of mqbrkrs and mqm, but is not a member of Administrators. The user "iibuser" can create new IIB nodes and do all required IIB development work. However, installation of the IIB product requires Administrator privileges (not required in this lab).

2.1 Open the Windows Log Monitor for IIB

A useful tool for IIB development on Windows is the IIB Log Viewer. This tool continuously monitors the Windows Event Log, and all messages from the log are displayed immediately.

From the Start menu, click IIB Event Log Monitor. The Monitor will open; it is useful to have this always open in the background.



This tool is not shipped as part of the IIB product; please contact us directly if you would like a copy.

2.2 Configure TESTNODE_iibuser for REST APIs

If you have already done a previous lab involving the REST APIs in this series of lab guides, you can skip to the next heading.

The IIB support for the REST API requires some special configuration for the IIB node and server.

_1.	Login to Windows as the user "iibuser", password = "passw0rd". (You may already be logged in). Start the IIB Toolkit from the Start menu.
_2.	Ensure that TESTNODE_iibuser is started.
_3.	Enable Cross-Origin Resource Scripting for REST. This is required when testing with the SwaggerUI test tool. See http://www.w3.org/TR/cors/?cm_mc_uid=09173639950214518562833&cm_mc_sid_50200000=1452177651 for further information. (Helpful hint - the VM keyboard is set to UK English. If you cannot find the "\" with your keyboard, use "cd .." to move the a higher-level folder in a DOS window), or change the keyboard settings to reflect your locale.) In an IIB Command Console (shortcut on the Start menu), run the command: `mqsichangeproperties TESTNODE_iibuser -e default -o HTTPConnector -n corsEnabled -v true`
_4.	Restart the IIB node

2.3 Configure Integration Bus node to work with DB2

If you have already done a previous lab involving the HRDB database in this series of lab guides, you can skip to the next heading.

To run this lab, the Integration Bus node must be enabled to allow a JDBC connection to the HRDB database.

1. Open an IIB Command Console (from the Start menu), and navigate to

   `c:\student10\Create_HR_database`

2. Run the command

   `3_Create_JDBC_for_HRDB`

   Accept the defaults presented in the script. This will create the required JDBC configurable service for the HRDB database.

3. Run the command

   `4_Create_HRDB_SecurityID`

4. Stop and restart the node to enable the above definitions to be activated

   `mqsistop TESTNODE_iibuser`

   `mqsistart TESTNODE_iibuser`

This will create the necessary security credentials enabling TESTNODE_iibuser to connect to the database.

Recreating the HRDB database and tables
The HRDB database, and the EMPLOYEE and DEPARTMENT tables have already been created on the supplied VMWare image. If you wish to recreate your own instance of this database, the command **`1_Create_HRDB_database.cmd`** and **`2_Create_HRDB_Tables.cmd`** are provided for this. If used in conjunction with the VM image, these commands must be run under the user "iibadmin". Appropriate database permissions are included in the scripts to GRANT access to the user iibuser.

2.4 Create Request/Response Queues

For this lab you need to create two MQ queues MQREQUEST_CLOUD and MQRESPONSE_CLOUD which are used in the MQ Scenario. In this next section you will run a script to create these queues on IB10GMGR (note: *This lab guide will assume that IB10QMGR is the default MQ Queue Manager, you may need to customize this value throughout this lab guide if you are running this in your own environment*).

1. In an Integration Console navigate to:

   C:\student10\IIBoC\EndpointConnectors\install

2. Enter the command:

   DefineQs.cmd

   Respond to the prompts (accept the default IB10QMGR). Ensure both queues are created successfully:

```
C:\student10\IIBoC\EndpointConnectors\install>defineqs
Enter Queue manager name (default is IB10QMGR):
The queues: MQREQUEST_CLOUD and MQRESPONSE_CLOUD on "IB10QMGR" will be
recreated,
 Ok to proceed? Use Ctrl-C to terminate.
Press any key to continue . . .

5724-H72 (C) Copyright IBM Corp. 1994, 2015.
Starting MQSC for queue manager IB10QMGR.

     1 : define ql(MQRESPONSE_CLOUD) replace
AMQ8006: WebSphere MQ queue created.
     2 : define ql(MQREQUEST_CLOUD) replace
AMQ8006: WebSphere MQ queue created.
2 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

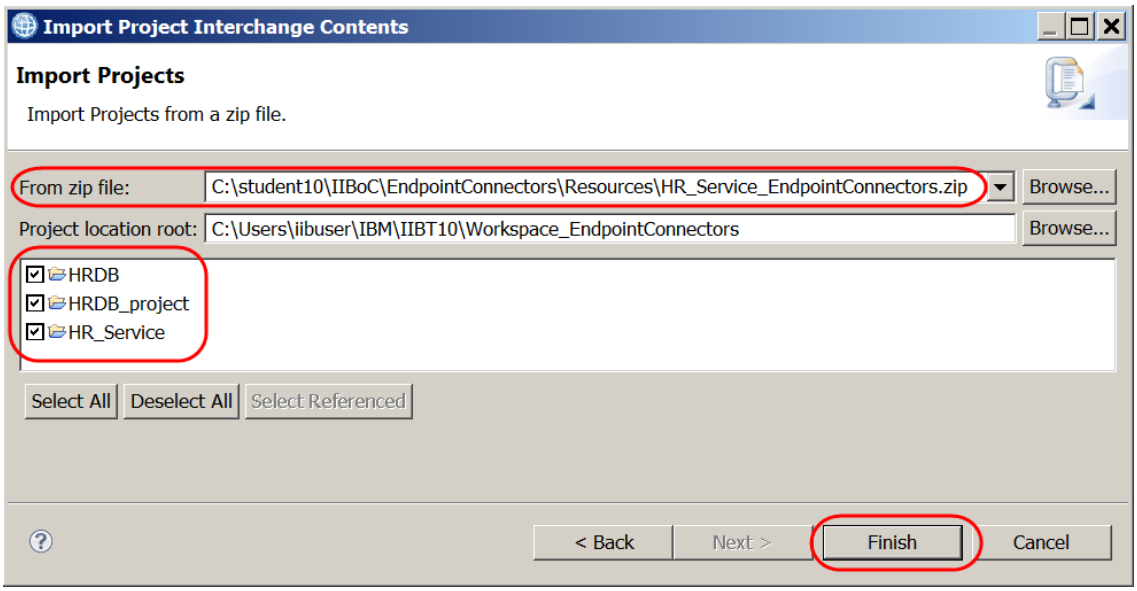
3. Review HR_Service Solution

In this section you will review the REST API implementation in HR_Service for the MQ and DB2 scenarios outlined in the introduction.

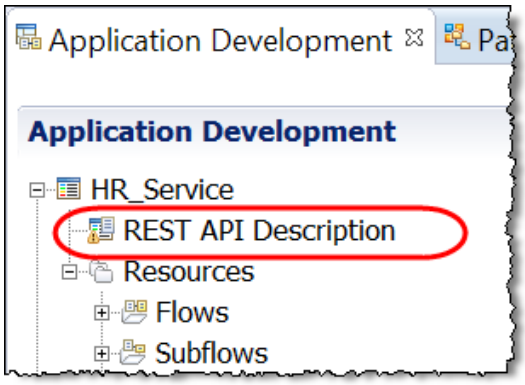
1. In the Integration Toolkit right click on the Application Development background and select Import.

   Navigate to `c:\student10\IIBoC\EndpointConnectors\Resources\` and import the PI file:

   `HR_Service_EndpointConnectors.zip`



2. Double click on the REST API Description to see the resources that are defined in this REST API:

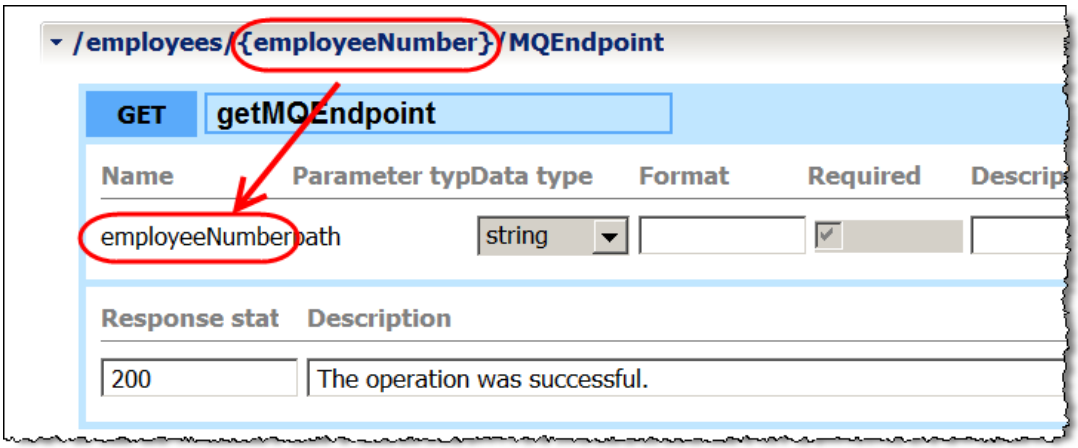


3.1 Review /employees/{employeeNumber}/MQEndpoint

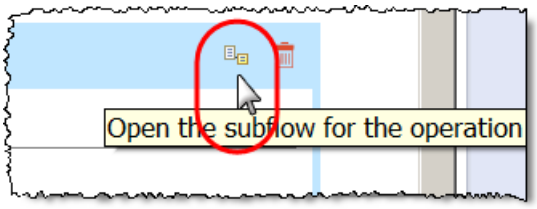
In this next section you will review the Resource that will demonstrate the ability for Resources running on IIB on Cloud to connect to an MQ queue defined locally (on premise) using an MQ Endpoint Policy.

1. Scroll down the list of Resources and expand **/employees/{employeeNumber}/MQEndpoint**:

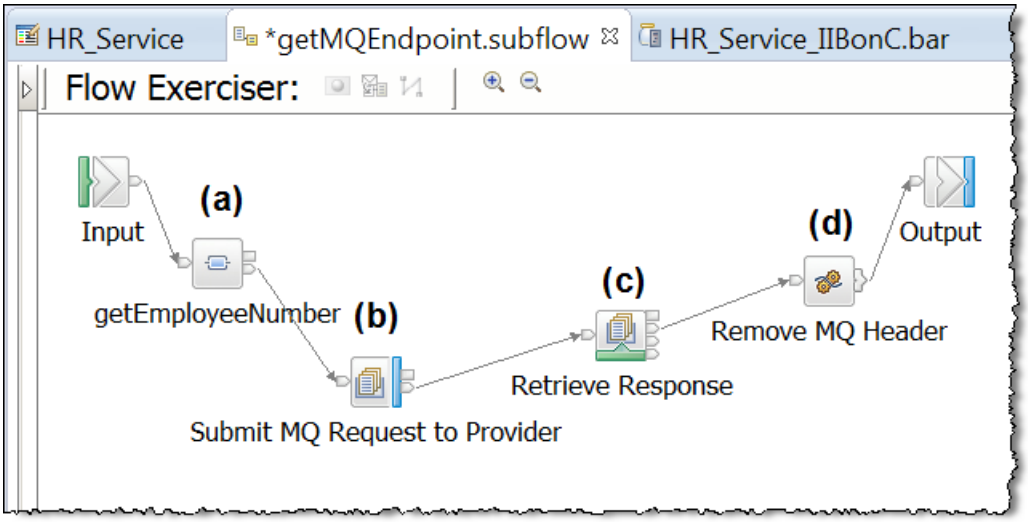
2. Note that employeeNumber is a required parameter of the resource because it is part of the path of the resource (so will be passed in the LocalEnvironment tree to sub flows that implement this Resource):



3. Click the following icon to open the sub flow that implements this resource:



4. The sub flow that implements this Resource performs the following function:



(a) A map is executed that takes the REST API input Parameter “employeeNumber” and puts the value in the message tree as a JSON object, for example the data output from this map could be `"{"employeeNumber":"000010"}`

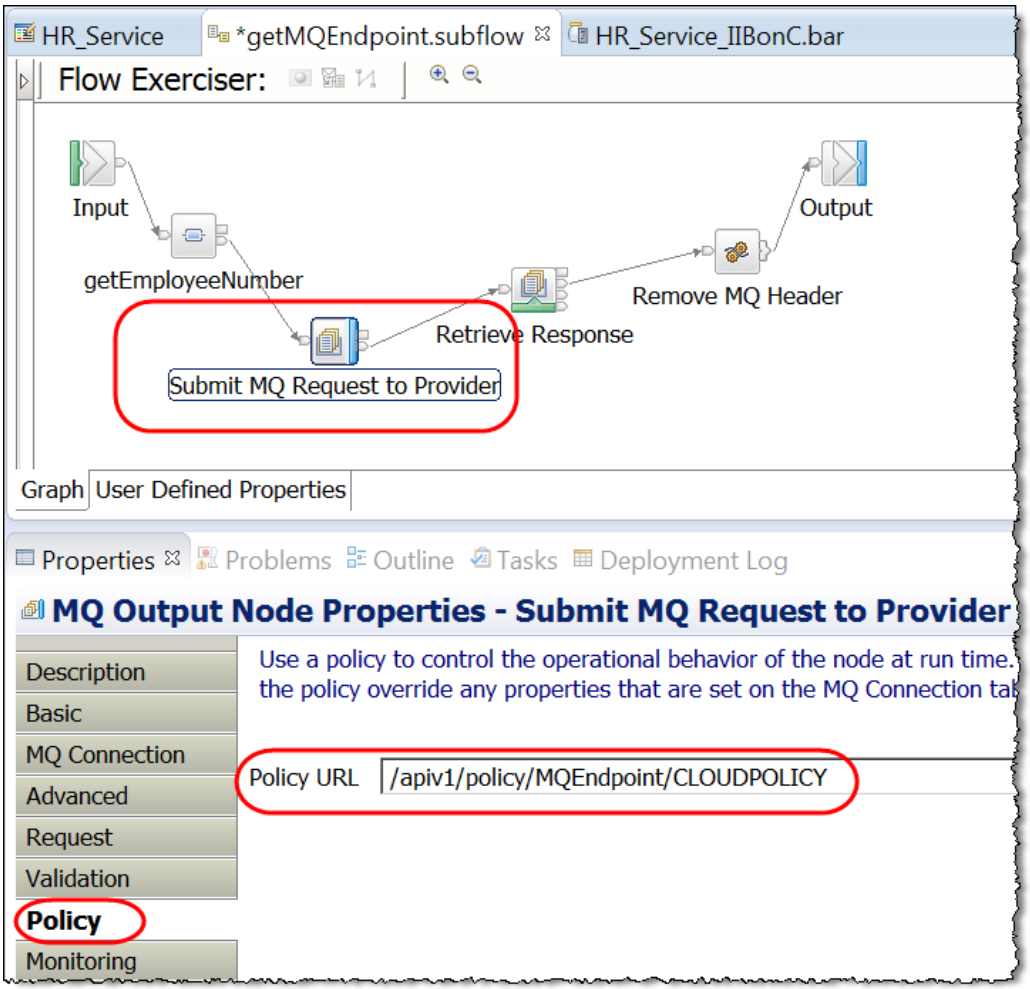
(b) Output from the map is written to an MQ queue – this forms the request input for the MQ Application that will obtain the data from the database.

(c) The Response from the MQ Application is obtained via an MQ Get node (the format of the response message is in JSON (of type EmployeeResponse – as defined in the REST API Definition for HR_Service).

(d) MQ Headers are removed from the message payload, **NB** at the time of writing this lab guide MQ Header nodes were not supported on IIB on Cloud so a small piece of ESQL node was written to remove the MQMD from the message.

5. Click on the MQ Output node “Submit MQ Request to Provider” and click the MQ Connection tab (in properties). Notice the connection is configured using an MQ Client connection (no connection details are specified.

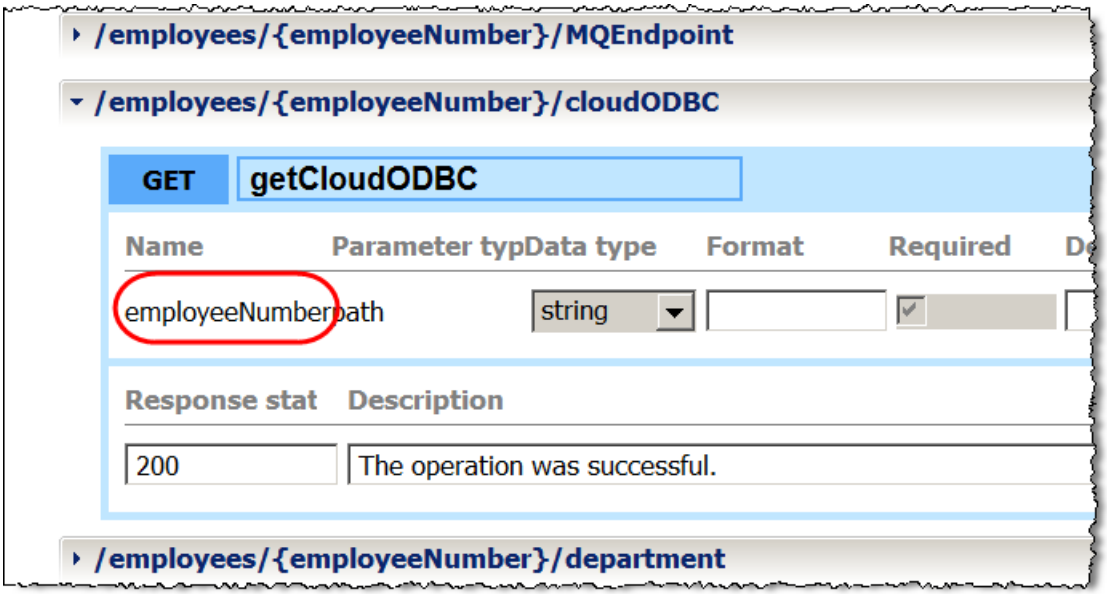
6. Click the Policy tab and note that the node has a Policy configured (*the MQ Get node has the same policy configured*). To run this REST API on IIB on Cloud this policy does not need to exist locally (on-premise), however the policy URL does need to have the same format as if it were created locally (ie the name prefixed by "**/apiv1/policy/MQEndpoint**". You will configure this policy on IIB on Cloud later in this lab guide:



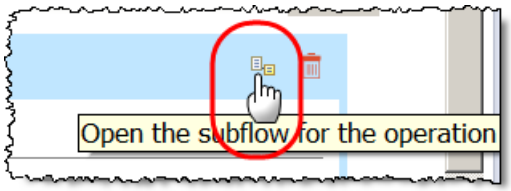
3.2 Review /employees/{employeeNumber}/cloudODBC

In this next section you will review the Resource that will demonstrate the ability for Resources running on IIB on Cloud to connect to a DB2 database running locally (on premise) using ODBC.

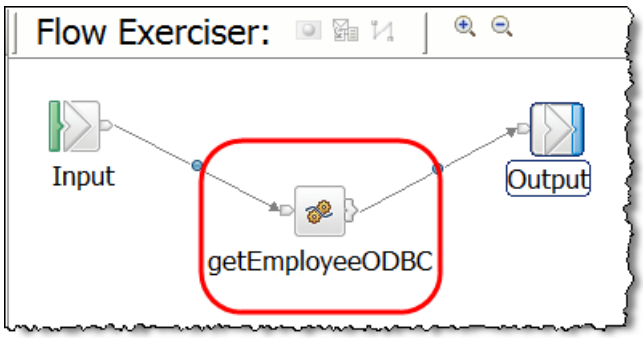
1. In the HR_Service REST API Description (In the Integration Toolkit) locate and open **/employees/{employeeNumber}/cloudODBC**, (note that the GET for this resource has the same **employeeNumber** parameter):



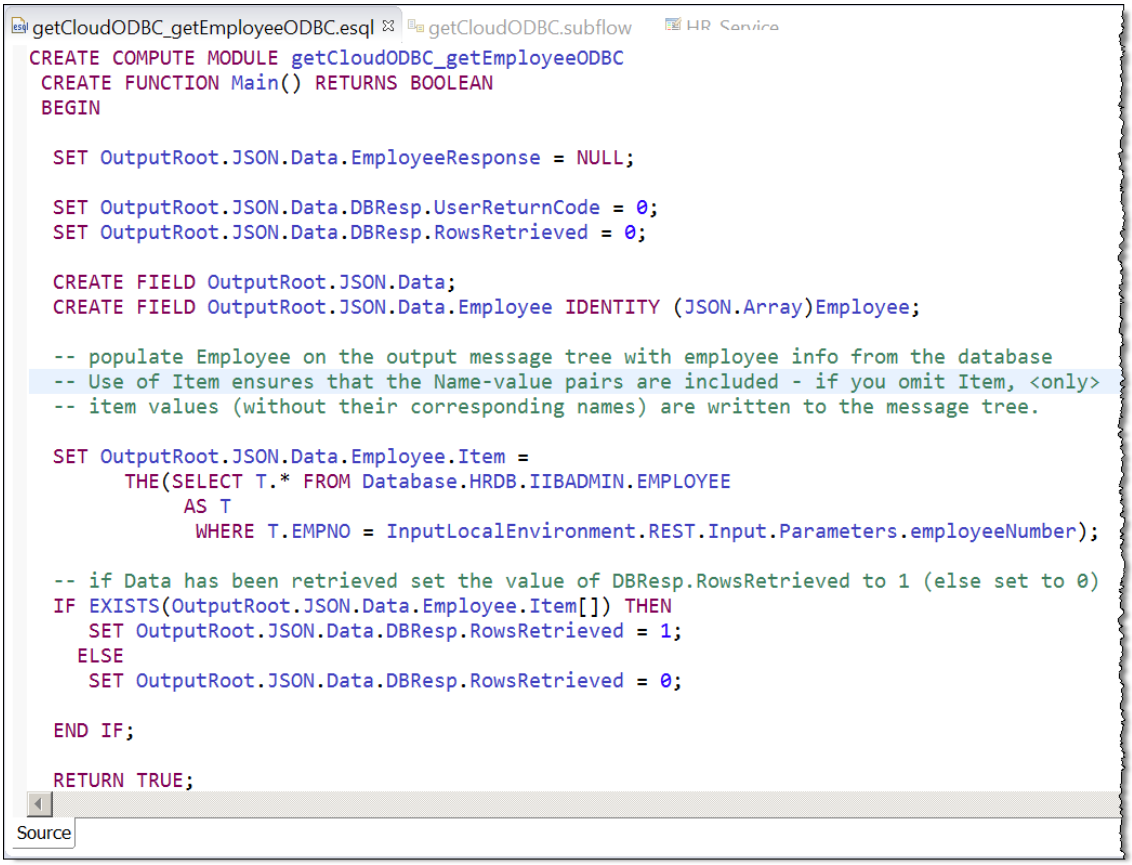
2. ". Open the subflow that implements this operation



3. The resource is implemented using an ESQL Compute node. Double click on **getEmployeeODBC** to open the ESQL that implements the operation:



4. After you have reviewed the code, close the ESQL editor.



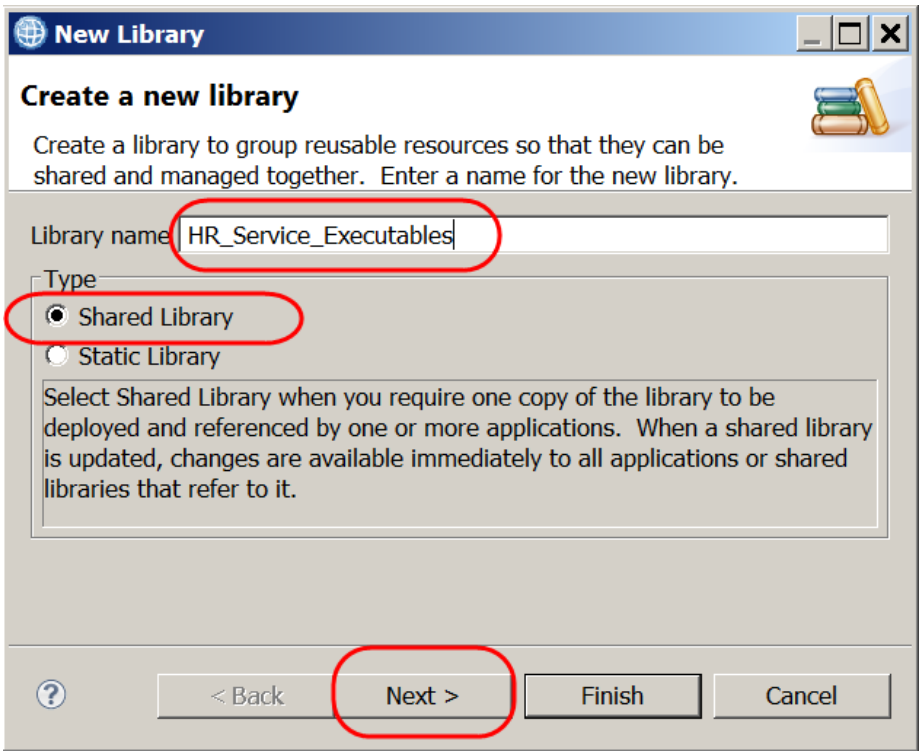
5. Close the subflow editor and the HR_Service REST API Description.

4. Create HR_Service_Executables Shared Library

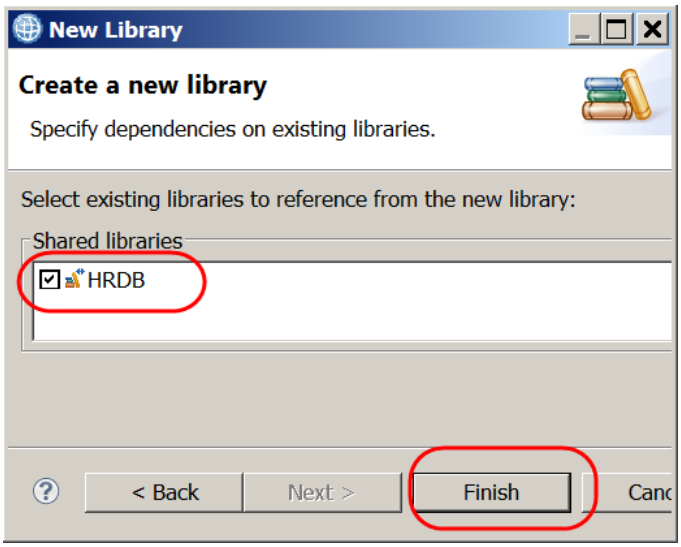
The map that you will create as part of this lab guide will be stored in a Shared Library ready for re-use by other applications in this series of lab guides. In this next section you will create this Shared Library.

1. In the Integration Toolkit, create a new Shared Library. Right click on the Application Development window and select New > Library:

   Call the Shared Library **HR_Service_Executables**, Click Next:



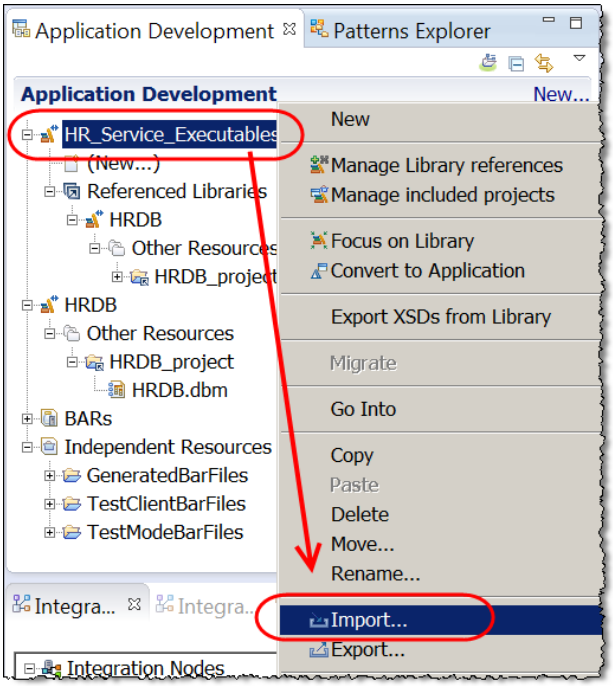
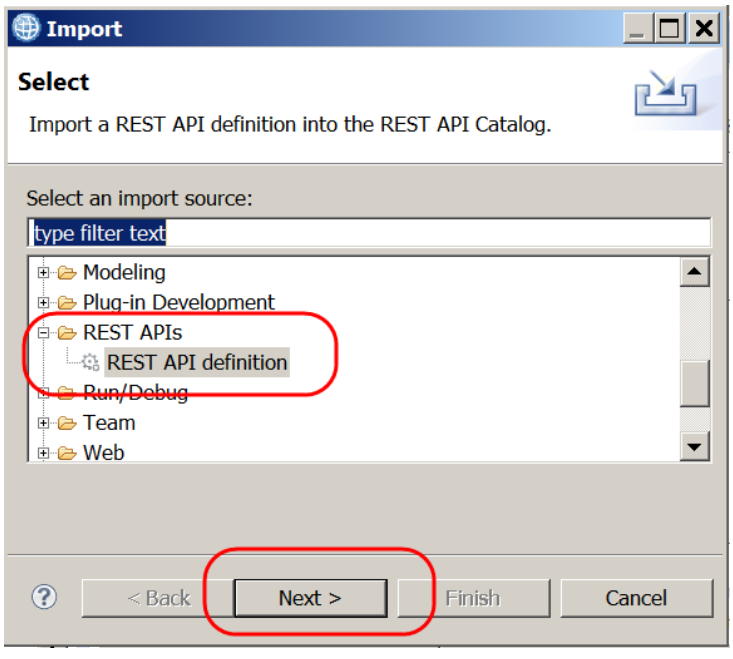
2. Select HRDB from the list of existing Libraries and click Finish:



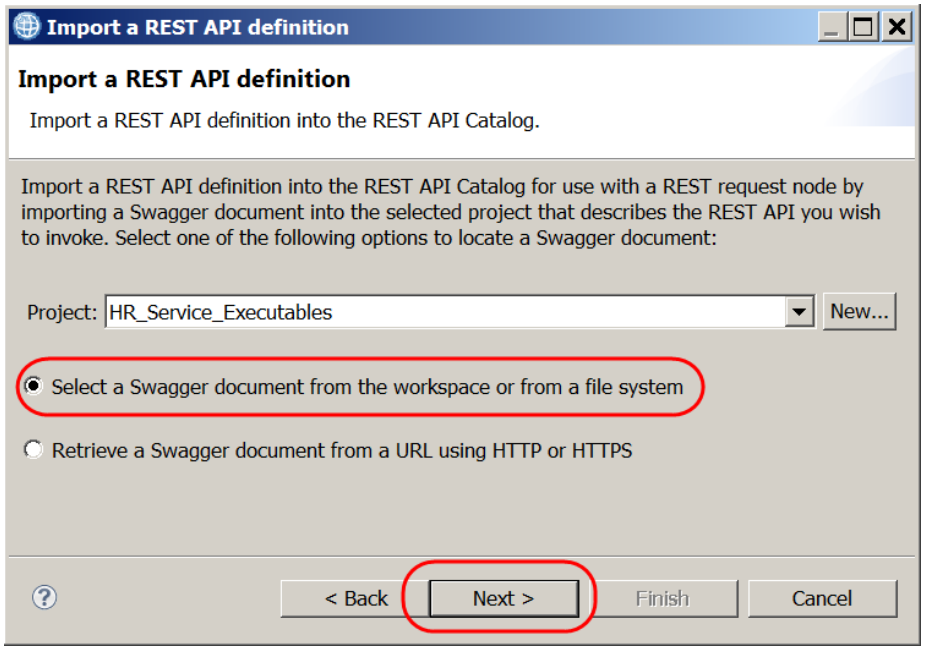
   Note: this is required in order for the map to have access to the HRDB database definition (it is stored in the HRDB Shared Library).

4.1 Import the REST API description

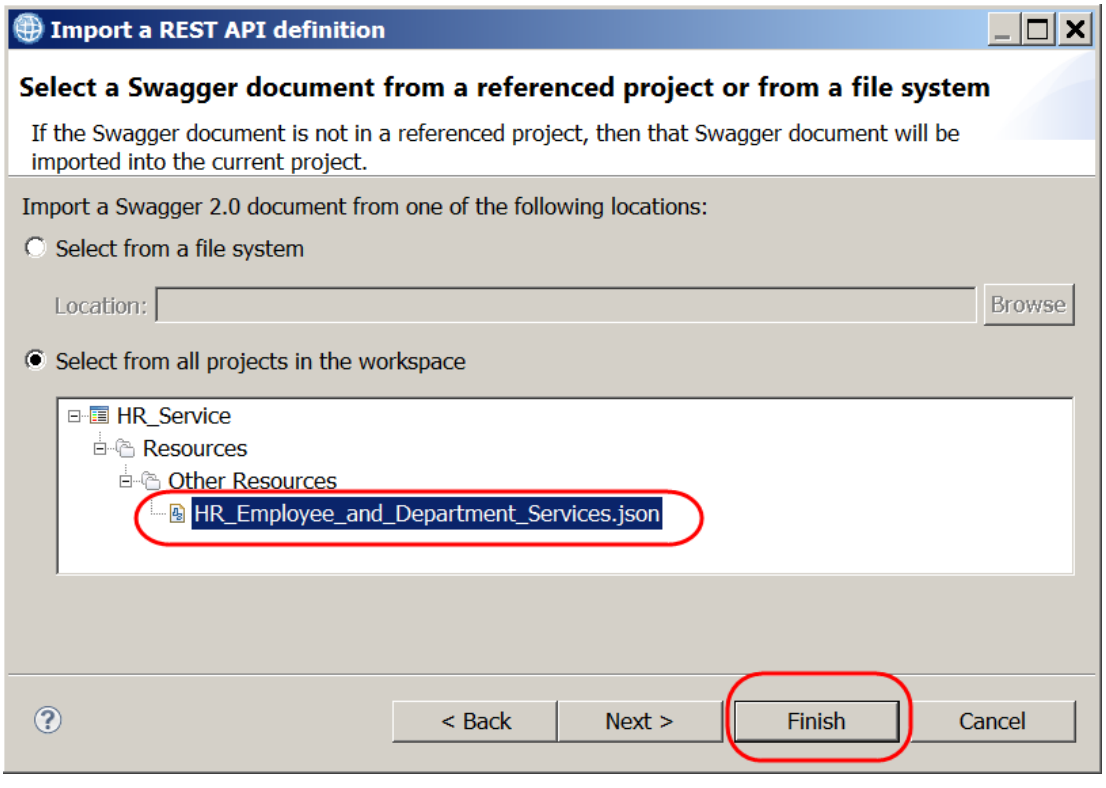
The map that you will create as part of this lab guide will require access to this JSON models that were created as part of the Lab guide that created HR_Service (Lab 01 and Lab 02). In this next section you will import the swagger file associated with HR_Service into the Shared Library. The result of doing this will mean that the map (that will also be created in the Shared Library) will have access to these JSON models.

1.	Right click on the Shared Library HR_Service_Executables and select Import: 
2.	In the input source window, Select REST APIs > REST API Definition, then click next : 

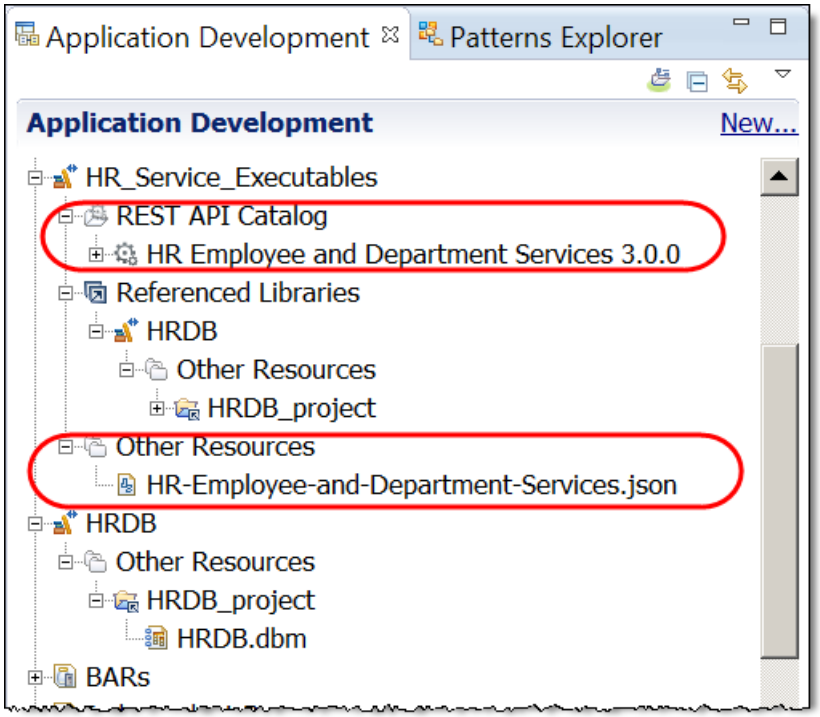
3. Click **Select a Swagger Document from the workspace,** then click next:



4. Choose to "**select from the all projects in the workspace**" and click Finish:



5. The REST API Catalog will now appear in the Shared Library:



Message flows created in this Shared Library will now have access to the REST API resources (for example when using the REST Request node to call a resource defined in the REST API).

Maps created in this Shared Library will also have access to the JSON model definitions stored in the swagger file.

4.2 Create getEmployeeJSON map

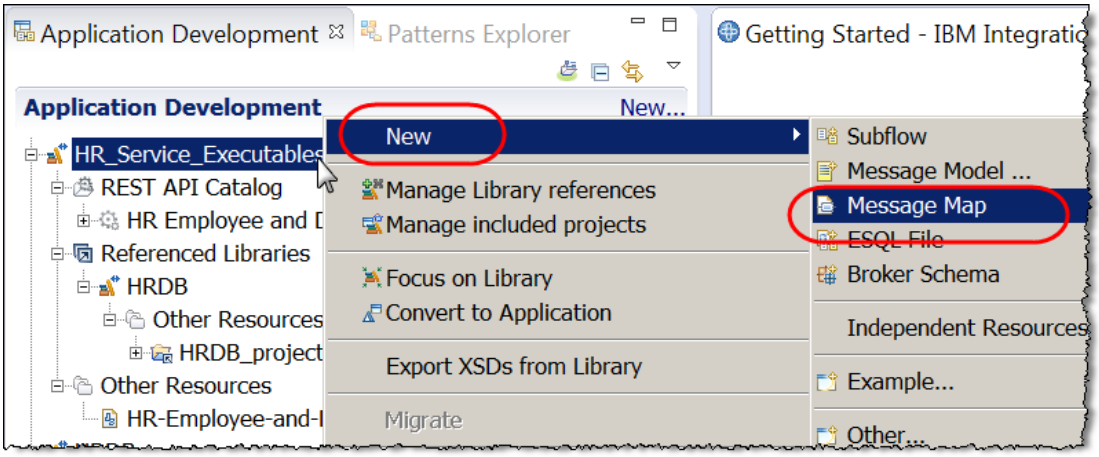
You will now create the getEmployeeJSON map and store it in the HR_Service_Executables Shared Library. The Map will process input from a JSON message with the following format:

```
{"employeeNumber":"nnnnn"}
```

Where "*nnnnn*" is the key of the database user to find.

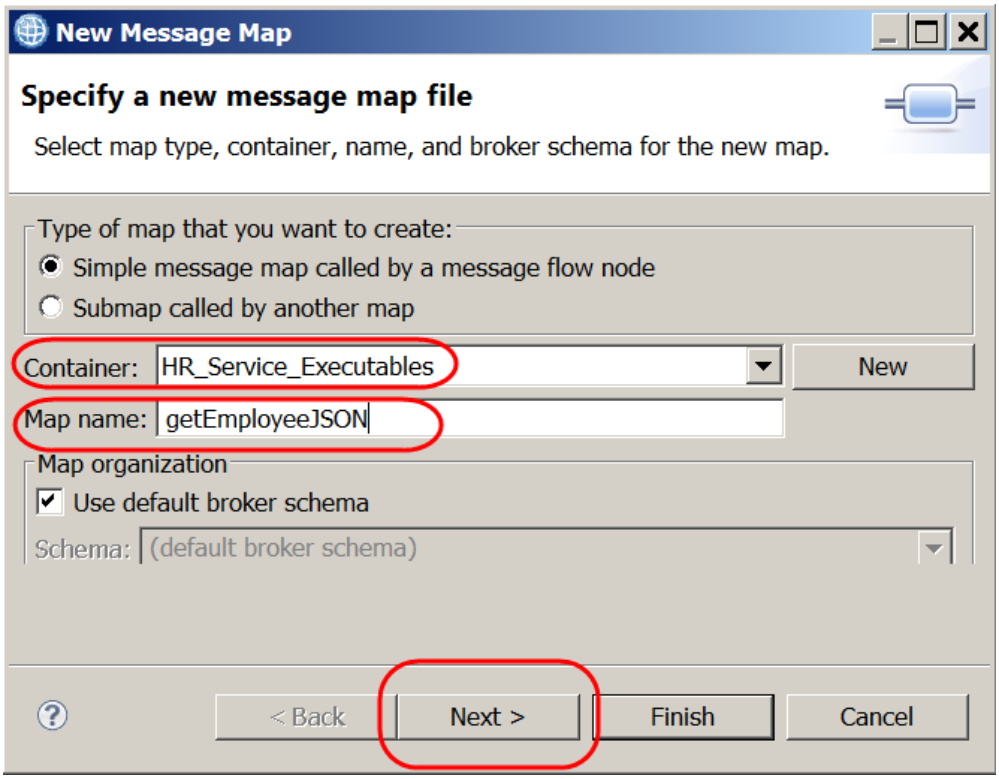
The output of the map will be in the format of the json object EmployeeResponse (defined as a json object in the REST API description – the map will have access to the definition of the json object because the swagger file has been imported into the Shared Library.

1. Right Click on HR_Service_Executables and select New > Message Map:



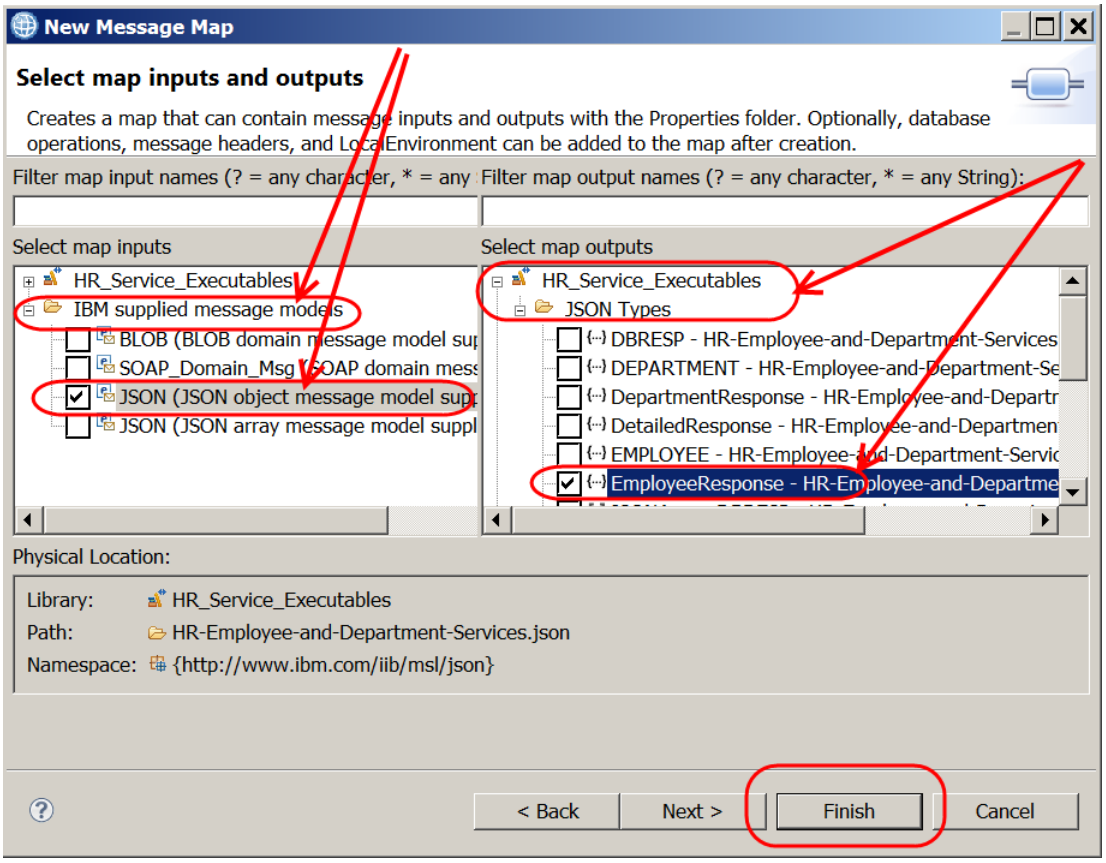
2. Make sure the Container name is **HR_Service_Executables**.

Call the map **getEmployeeJSON** and click next:



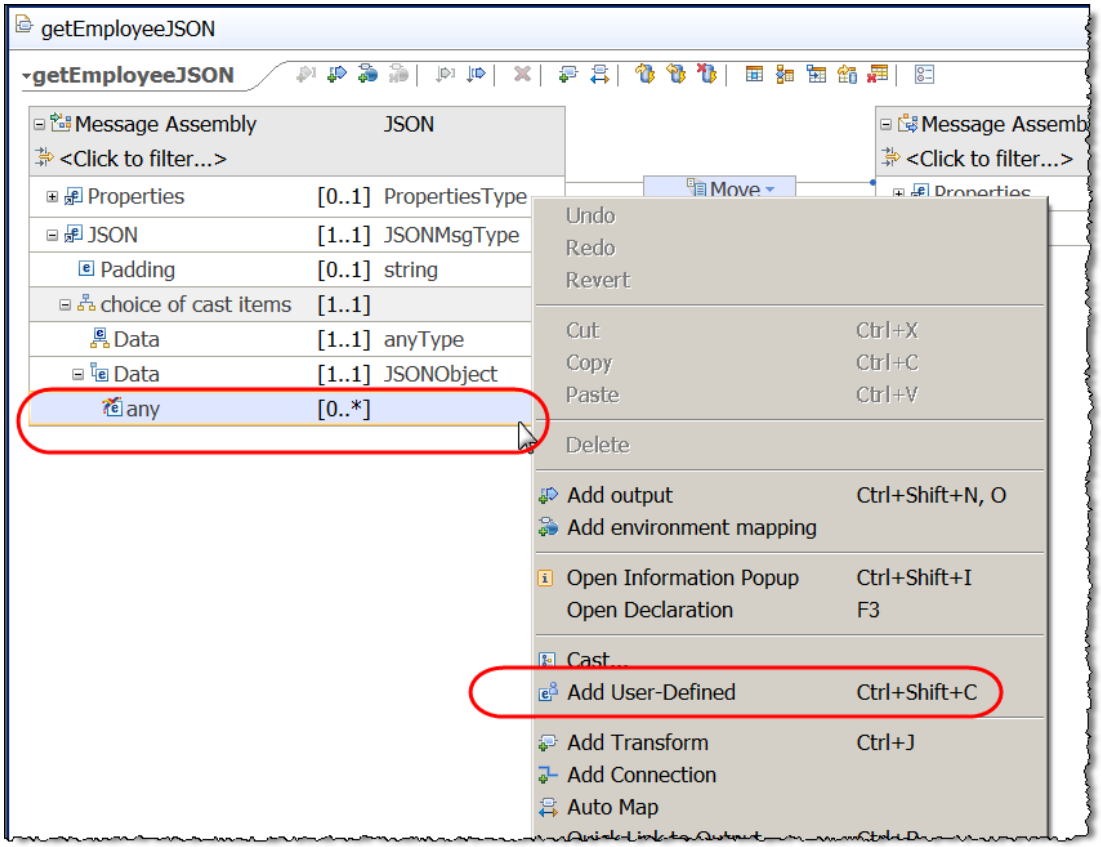
3. On the "**Select Map inputs**", expand IBM supplied message models and select JSON Object Model .

   On "**Select map outputs**", expand HR_SERVICE_Executables > JSON Types and select EmployeeResponse, (note these JSON message models appear here because earlier you imported the REST API Definition file into the Shared Library which describes the HR_Service). Click Finish.

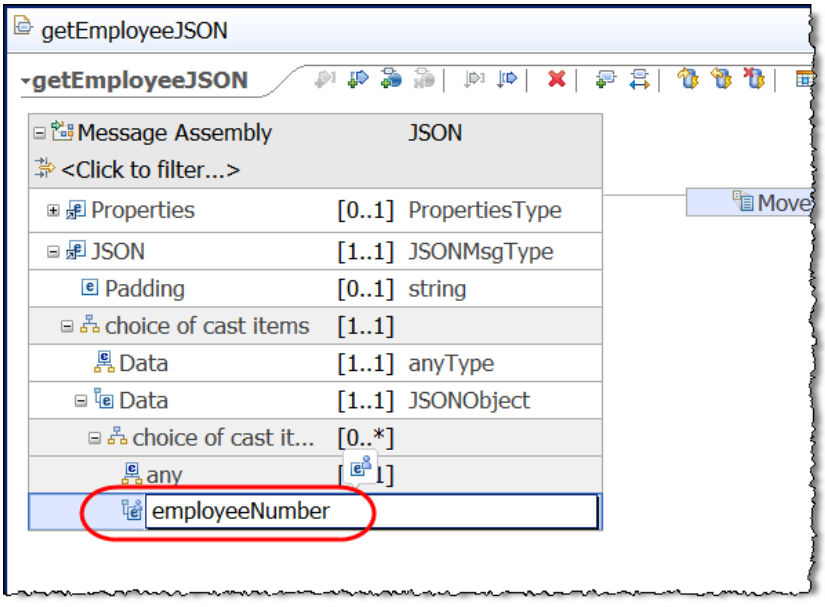


4. When the mapping editor opens, expand all levels of the JSON element (in the input Message Assembly).

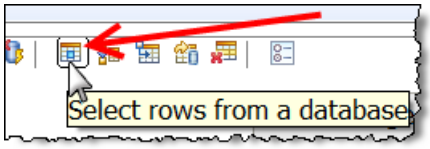
   Right Click on the final “any” element and select Add User-Defined:



5. Rename the element to employeeNumber:



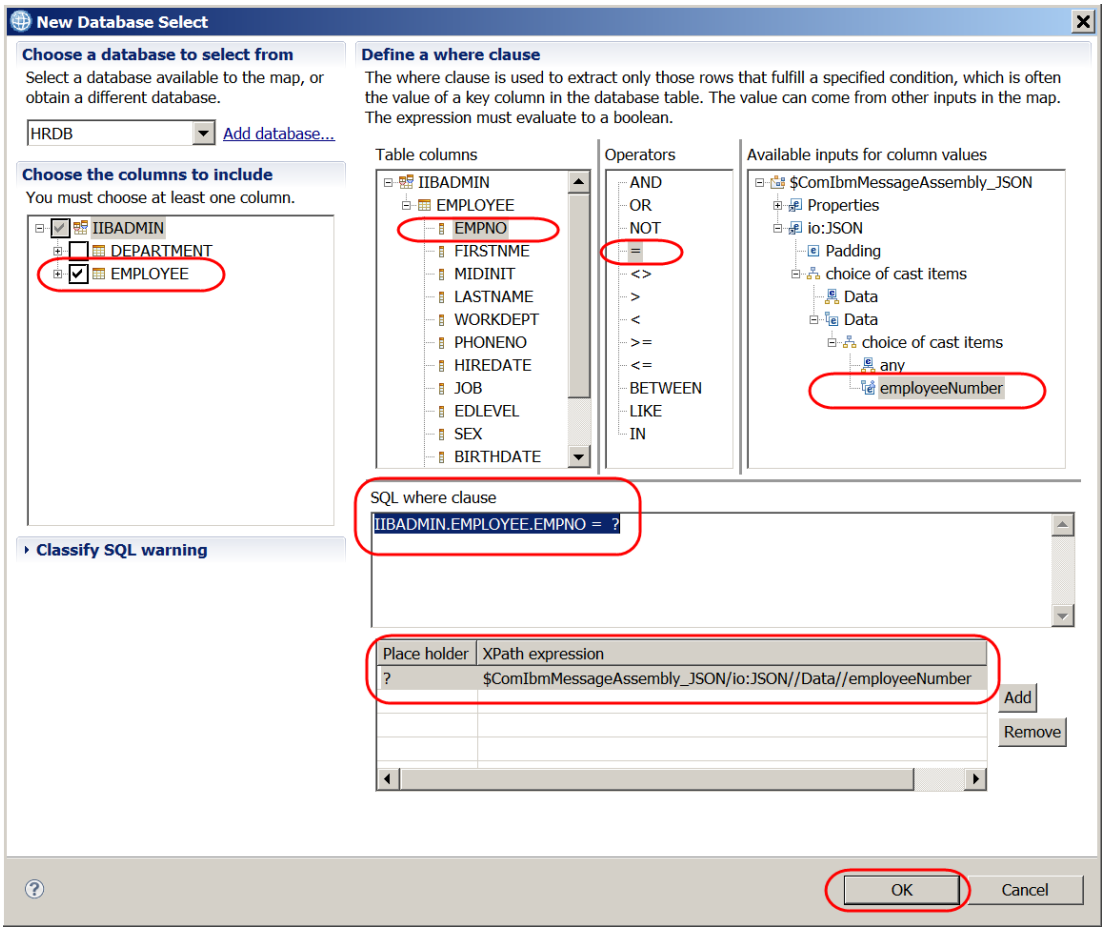
6. Click Select Rows from a Database:



7. In the “New Database Select” window,

a) Select **EMPLOYEE** in “Choose the columns to include”
b) Delete the **1=1** in the SQL where clause and leave your cursor in this box.
c) Double click on the IIBADMIN>EMPLOYEE>EMPNO (leave the cursor in the box once more)
d) Double click on the equals sign “**=**” under the list of operators (this will add an equal sign to the “SQL where clause” box.
e) In “available inputs for column values” expand JSON until you find **employeeNumber** (this should be the last element).
f) Double click on **employeeNumber**, this will add a question mark “?” into the where clause. The meaning of this question mark appears in the XPATH expression – this is the value of employeeNumber passed into the map.
g) Click **OK** when complete.

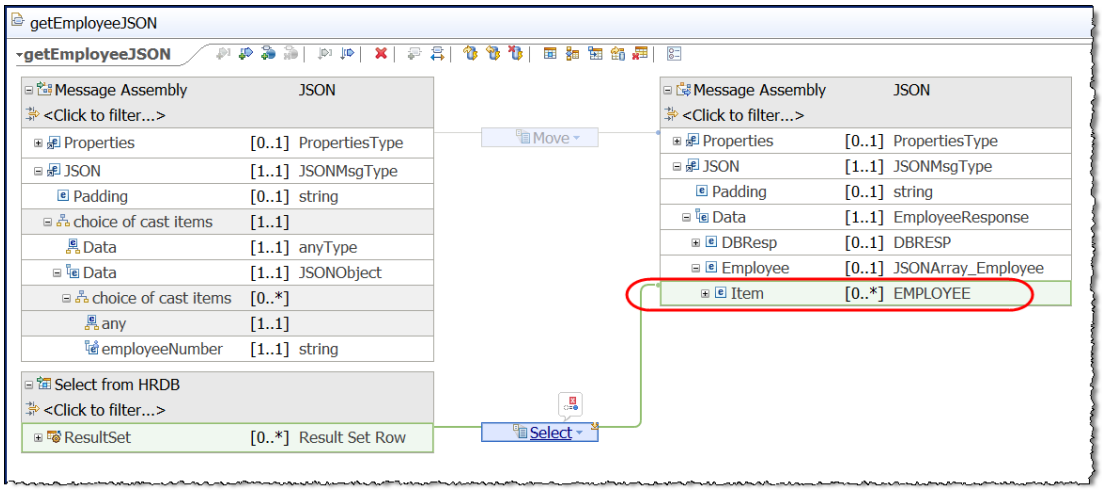
The SQL where clause will look like this when complete:



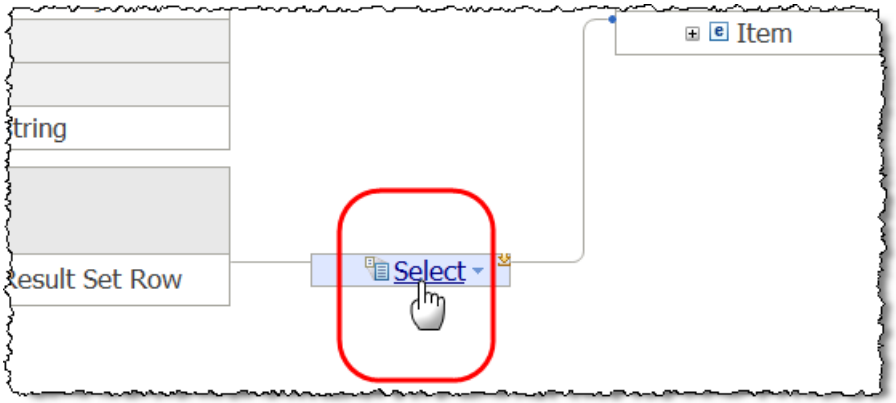
8. “Select from HRDB” will be created in the message map.

   On the Output Message Assembly expand JSON > Data > Employee

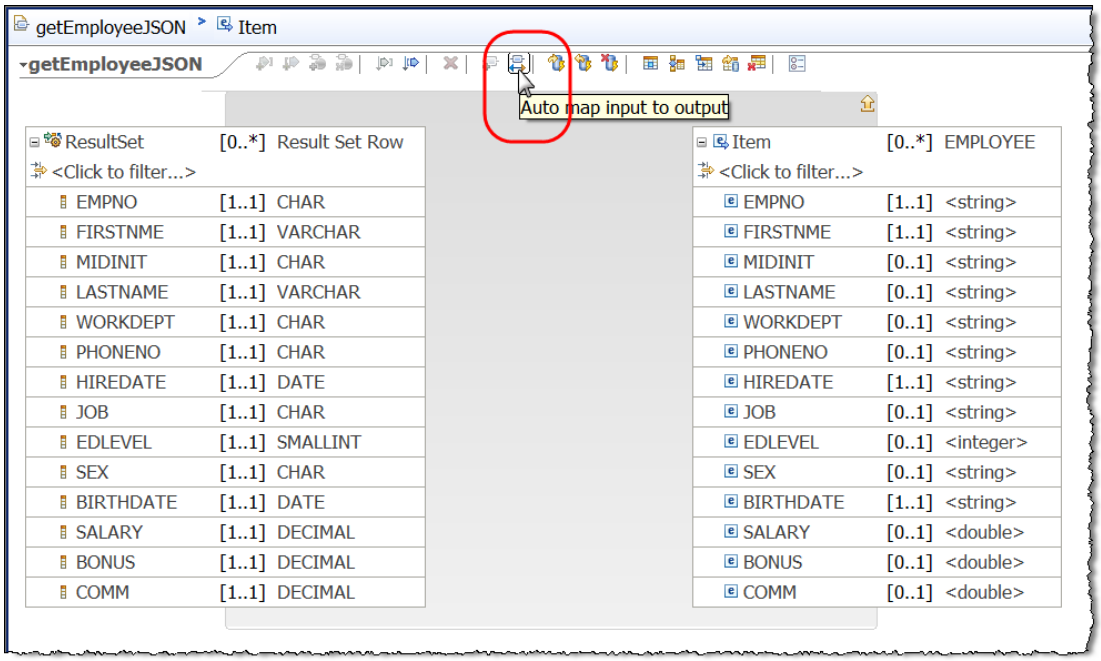
   Connect the **Select** transform to **Item** (of type **EMPLOYEE**):



9. Click the Select (text) to enter the nested map that will process the data passed back from the Select statement:

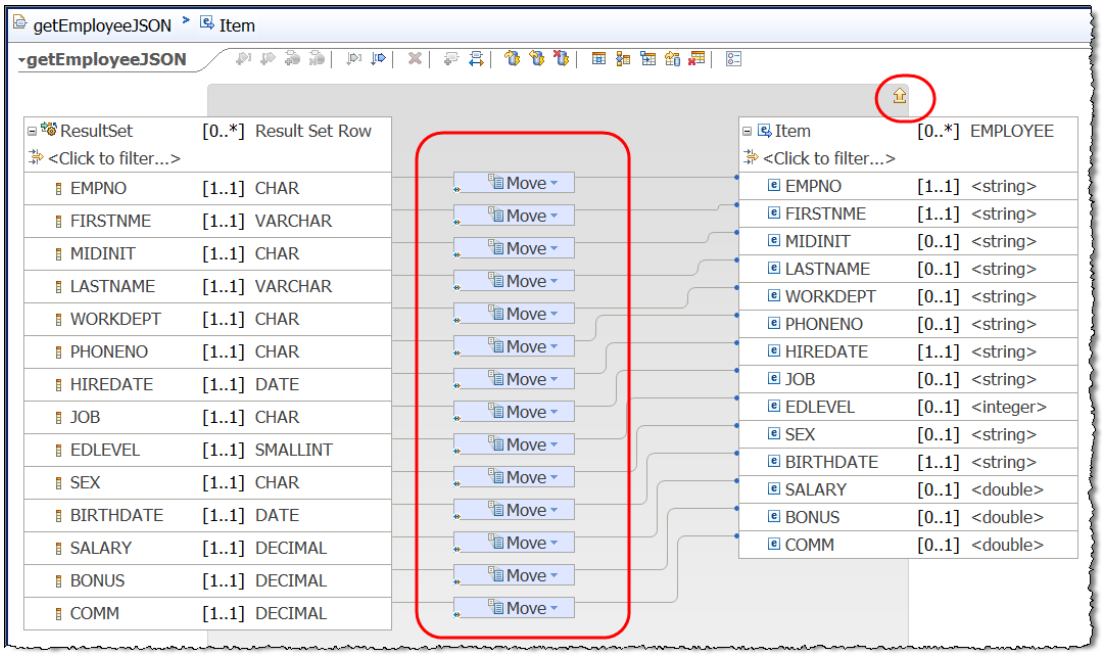


10. In the nested map, select the “Auto map” input and output icon and accept all the defaults to map like-named items:



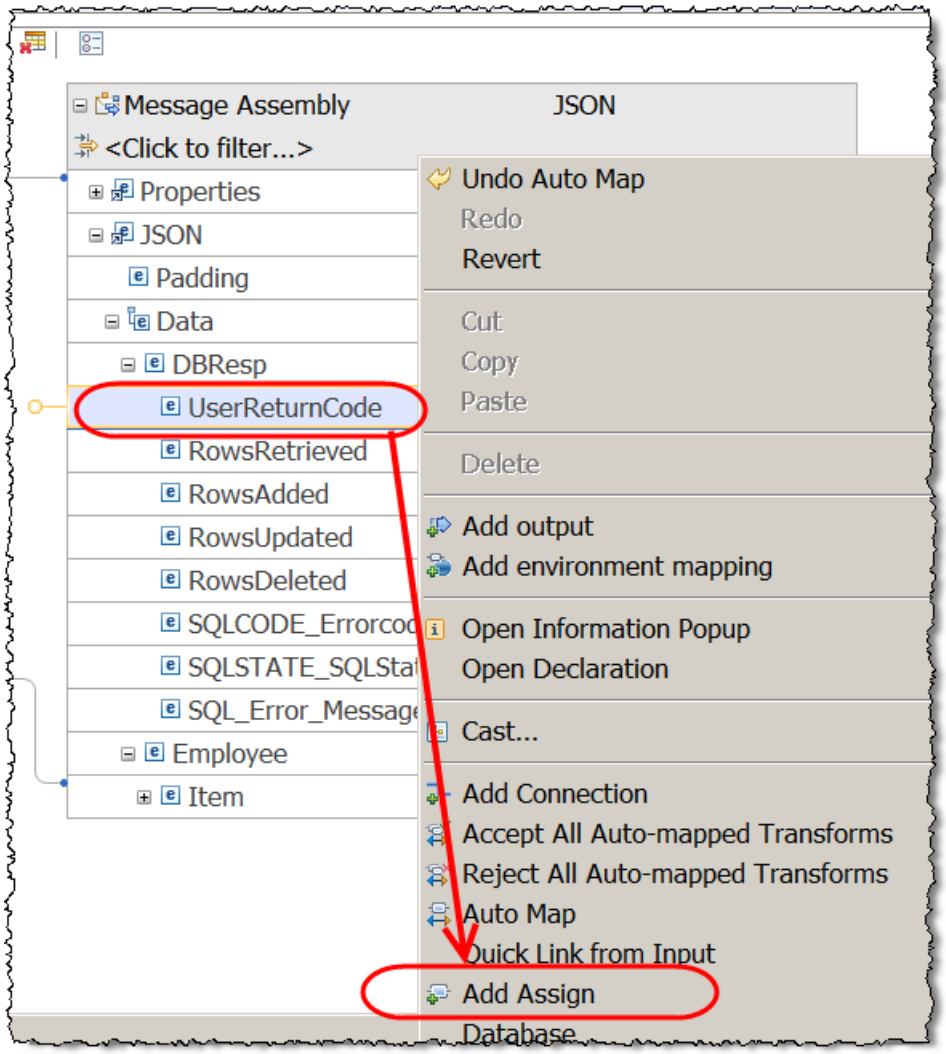
11. The nested map will look like this when complete.

Click the yellow Up arrow to return to the previous higher level of the map:



12. Expand DBResp on the output message assembly.

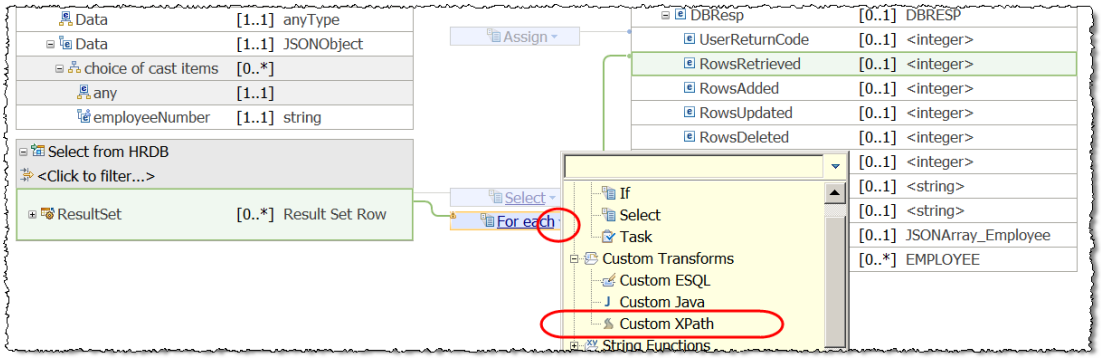
Right click on **UserReturnCode** and click **Add Assign**:



Leave the (default) value for the assign transform to 0 (zero)

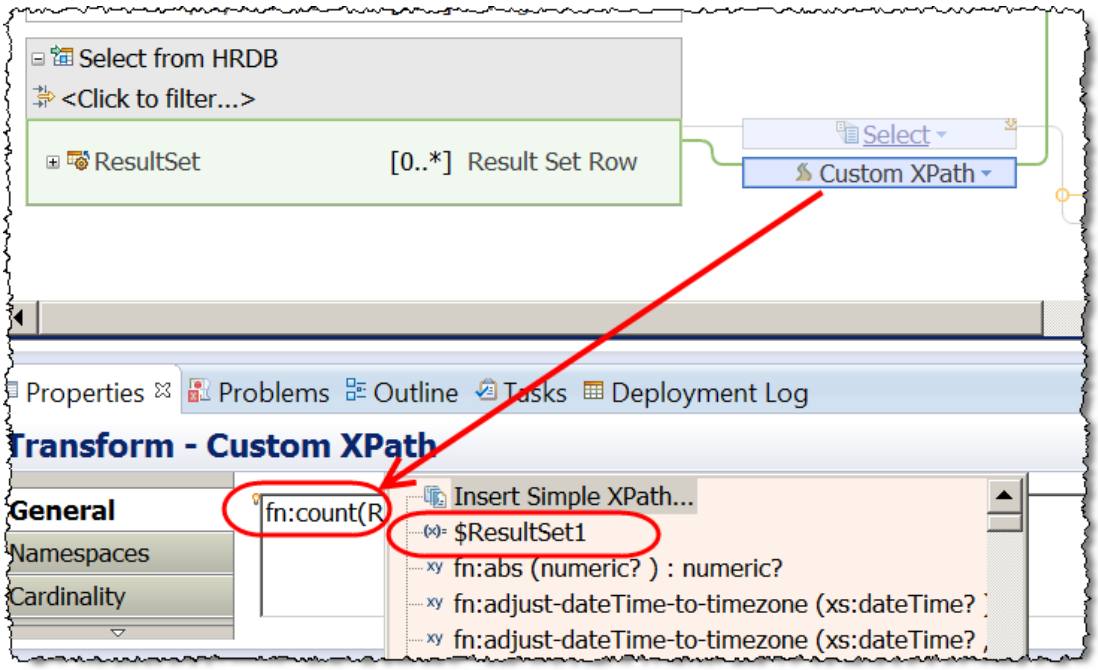
13. Connect ResultSet to DBResp.RowsRetrieved.

Change the For each transform to **Custom XPATH** (right click on the triangle to the right of the For Each transform text to see the options):

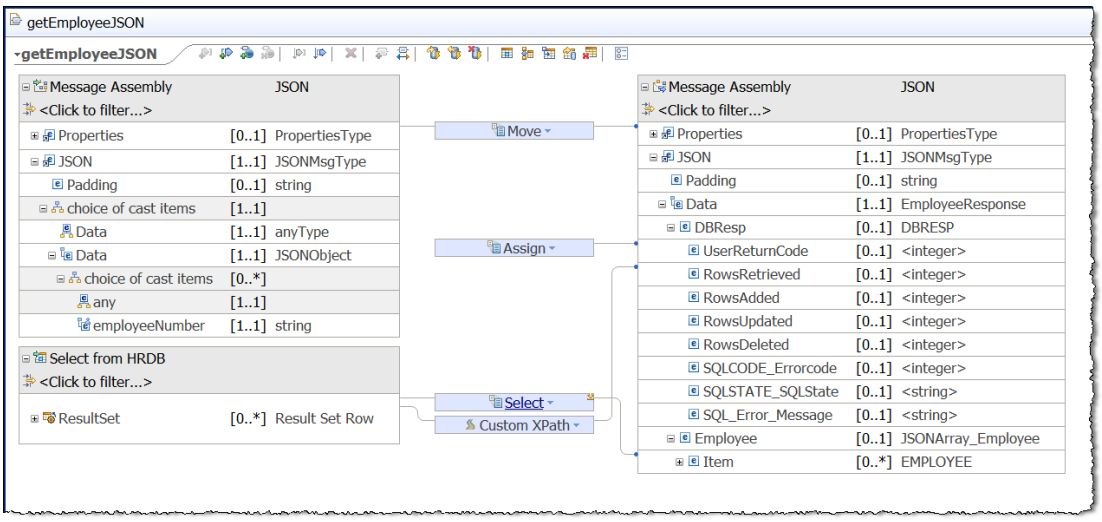


14. In the Custom XPATH properties type `fn:count( )`

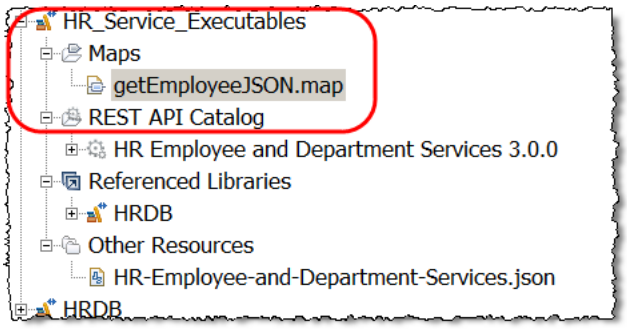
Place your cursor between the two brackets and type R then press Ctrl and the Space bar. Select the ResultSet pertinent to your environment – in the example here **$ResultSet1**:



15. The map will look like this when complete:



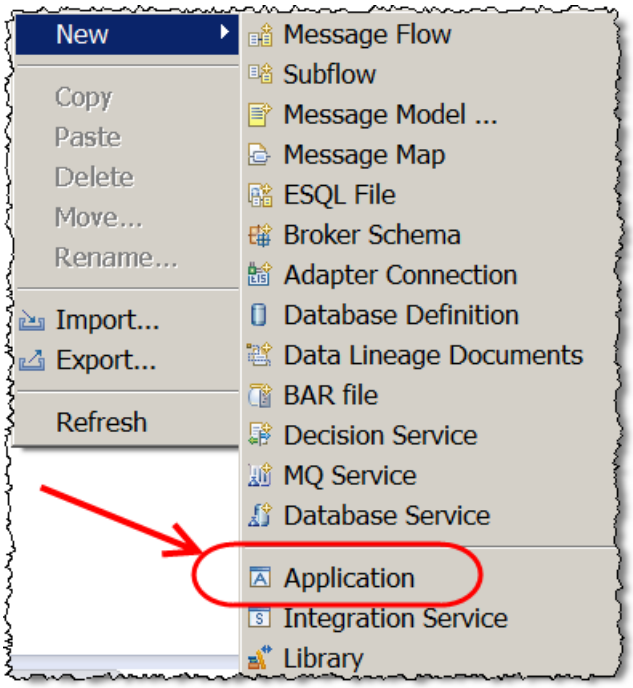
16. Save (Ctrl S) and close the map. Check to ensure the map has been saved in the HR_Service_Executables Shared Library:



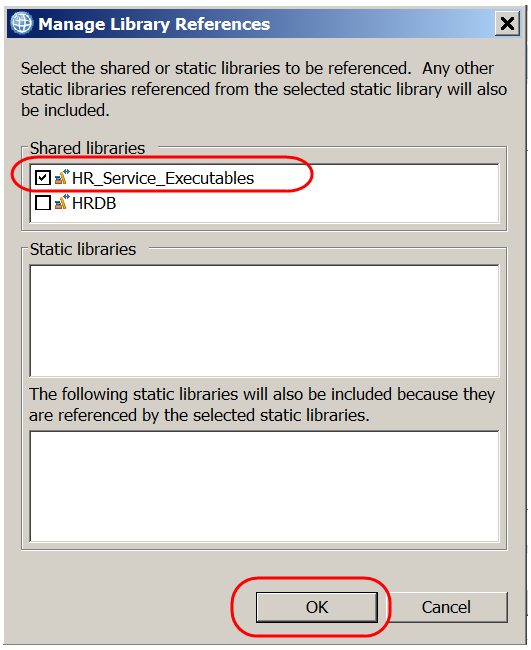
5. Create HR_Service_MQProvider Application

This application will be used as a provider of details from the EMPLOYEE table. The implementation (subflow in HR_Service) that is associated with Resource **/employees/{employeeNumber}/MQEndpoint** will write request data (an MQ message) to MQREQUEST_CLOUD (an MQ queue which exists locally). The request data will be processed by this application and its response will be written to MQRESPONSE_CLOUD (also a local queue). The HR_Service REST API will obtain this response data from MQRESPONSE_CLOUD and provide the details to the requestor.

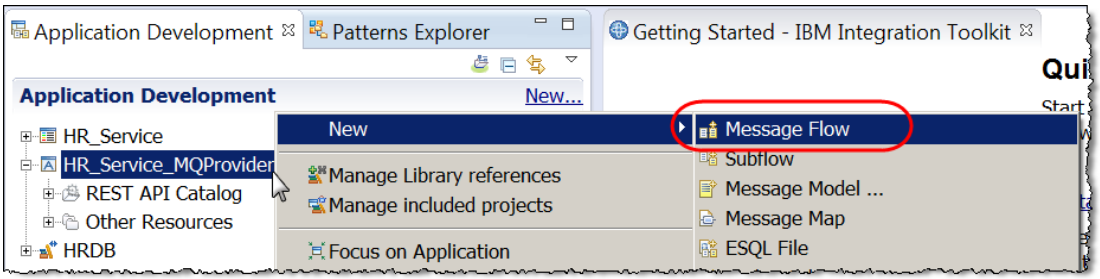
5.1 Create the Application to respond to HR_Service

1.	In the Integration Toolkit create a new Application. Call it **HR_Service_MQProvider**. (right click on the Application Development window and click New > Application): 

2. Right-click on the Application, select Manage Library References. In the Manage Library References window, select HR_Service_Executables then click OK:



3. In the Integration Toolkit, right click on **HR_Service_MQProvider** and select New > Message Flow.

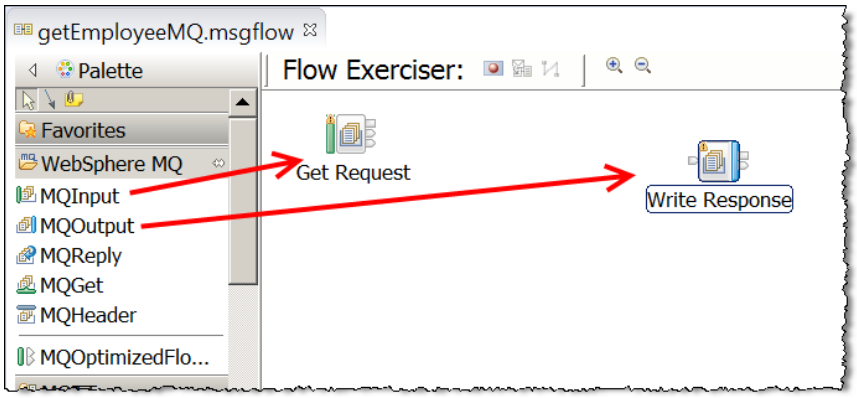


4. Call the message flow **getEmployeeMQ**

5. When the message flow editor opens, add the following:

   a) An MQ Input node (call it `Get Request`).
   Configure the queue name field ("Basic" tab) to be: `MQREQUEST_CLOUD`.
   Configure the (local) queue manager name ("*MQ Connection*" tab) to be: `IB10QMGR`

   b) An MQ Output node (call it `Write Response`)
   Configure the queue name field to be: `MQRESPONSE_CLOUD`.
   Configure the (local) queue manager name to be: `IB10QMGR`



6. The data being passed to the MQGet node (Get Request) is in JSON format so the MQGet node needs to be configured to handle JSON input.

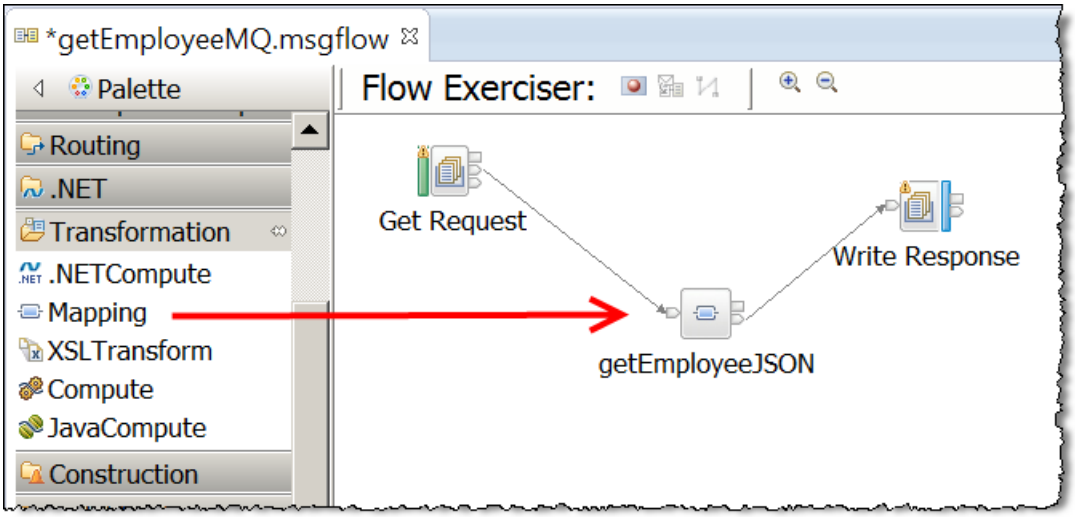
   Configure the Message Domain on the *"Input Message Parsing" tab*) to: `JSON`

5.2 Configure the Mapping node

1. Add a mapping node between the two MQ nodes.

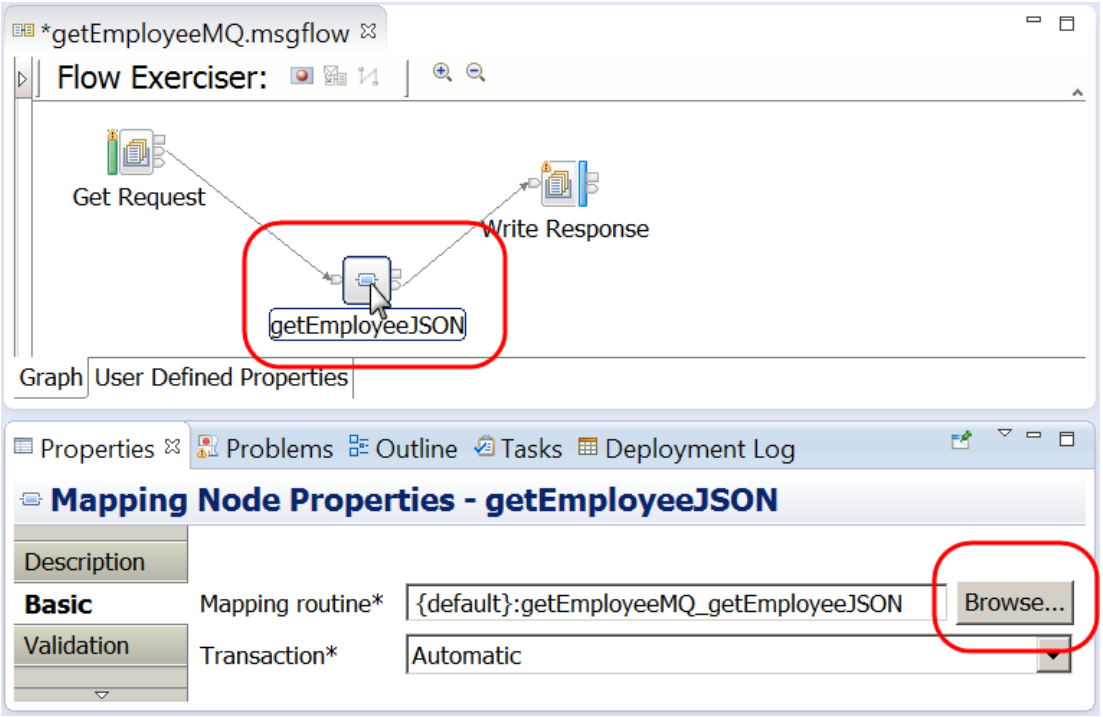
   Give the mapping node a label of getEmployeeJSON.

   Connect the mapping node to the Get Request (MQInput) and Write Response (MQOutput) nodes as shown:

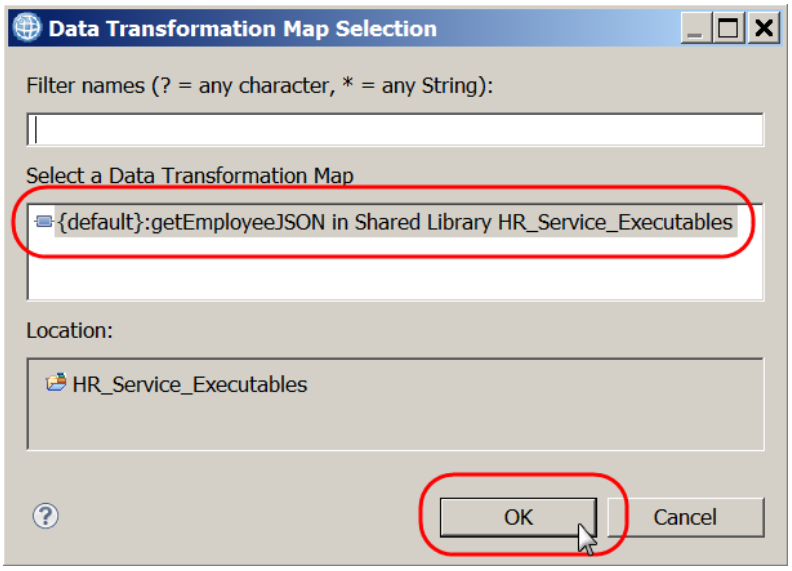


2. The map has already been created and stored in the HR_Service_Executables Shared Library, you will now configure this mapping node to reference that map.

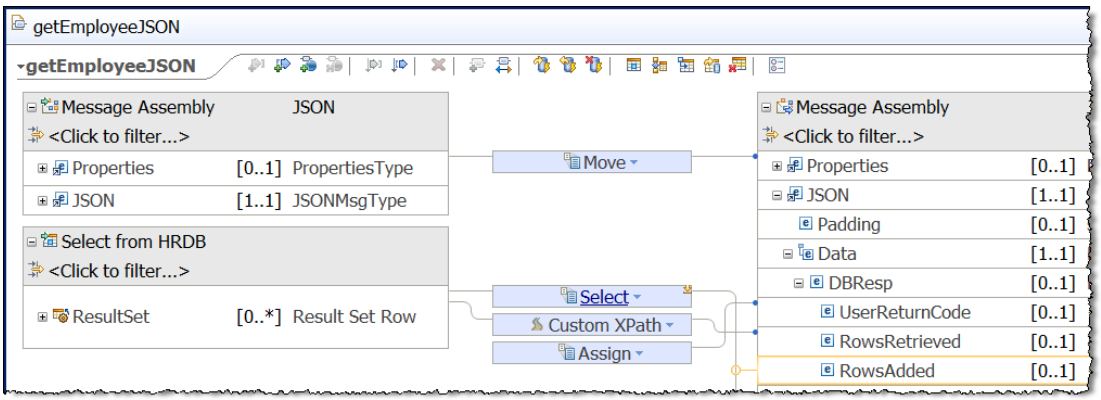
   Select the getEmployeeJSON map (single click), In the properties tab (Basic) click the Browse button:



3. Select the getEmployeeJSON map from the Shared Library and click OK:

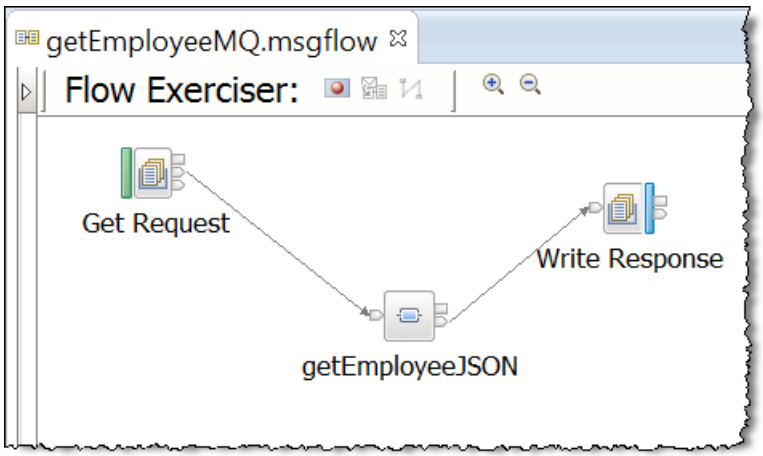


4. Double click on the map to make sure you have performed the correct configuration. The map will look like this:



Close the map editor.

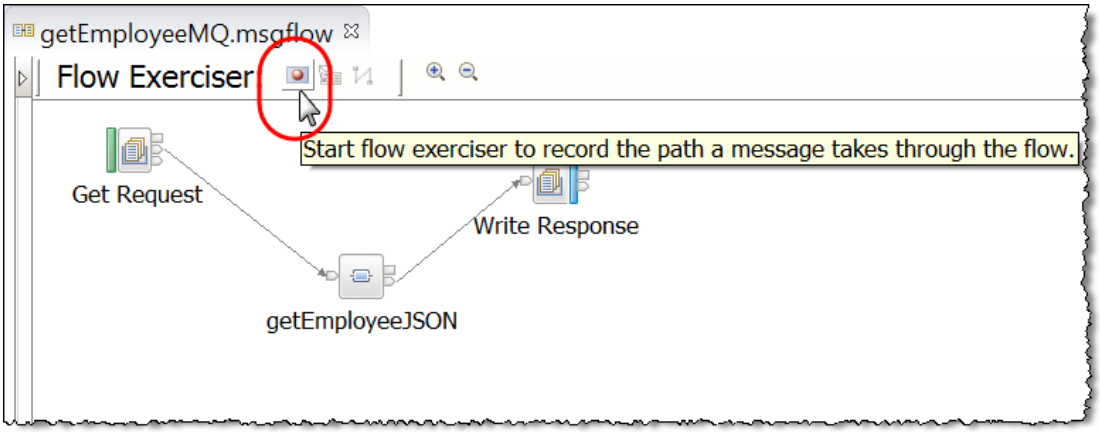
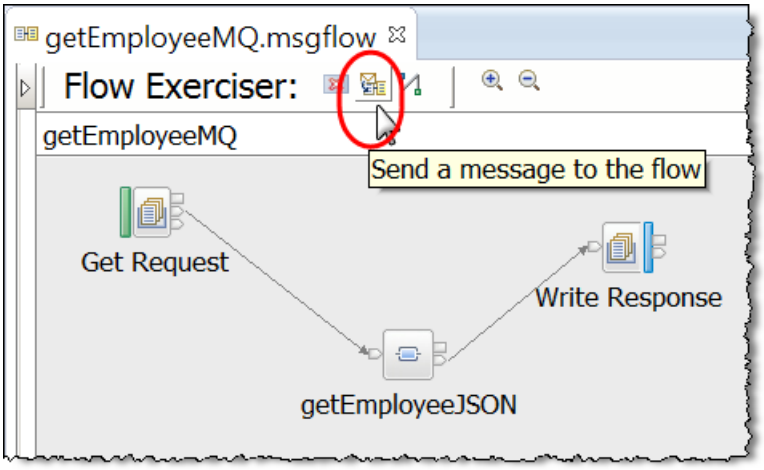
5. Save the getEmployeeMQ message flow, it will now look like this:



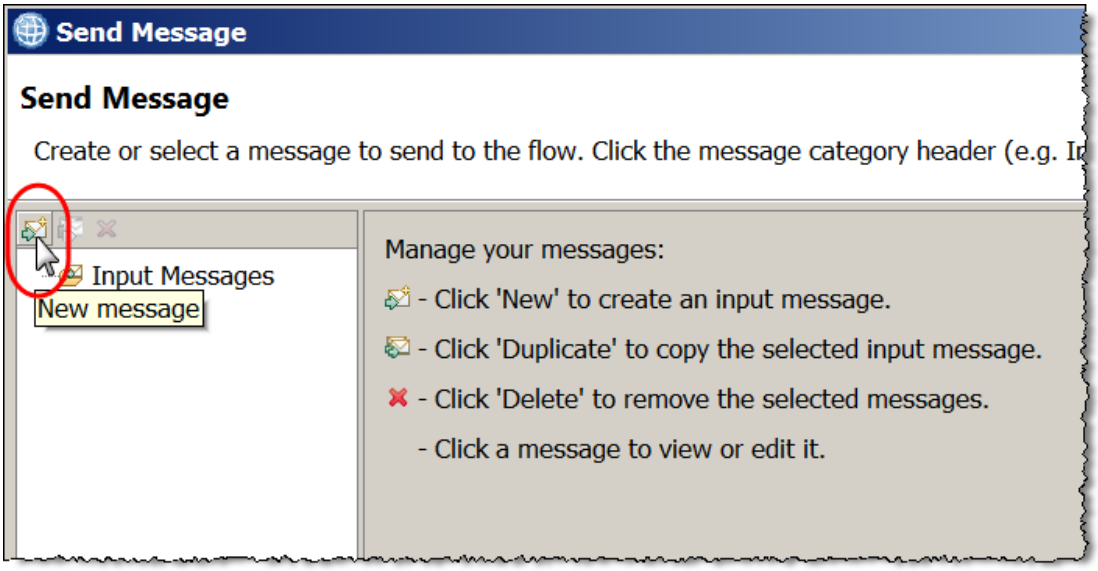
Leave the message flow editor open.

5.3 Test (locally) HR_Service_MQProvider

You will now test the application using the Flow Exerciser to ensure that HR_Service_MQProvider works correctly before using it with the HR_Service REST API.

1.	In the message flow editor start the Flow Exerciser (select the Record button):  This will automatically deploy the application. If you see messages that need responses respond to them appropriately.
2.	When the Flow Exerciser has started the message flow will have a grey background. Click the icon to send a message through the flow: 

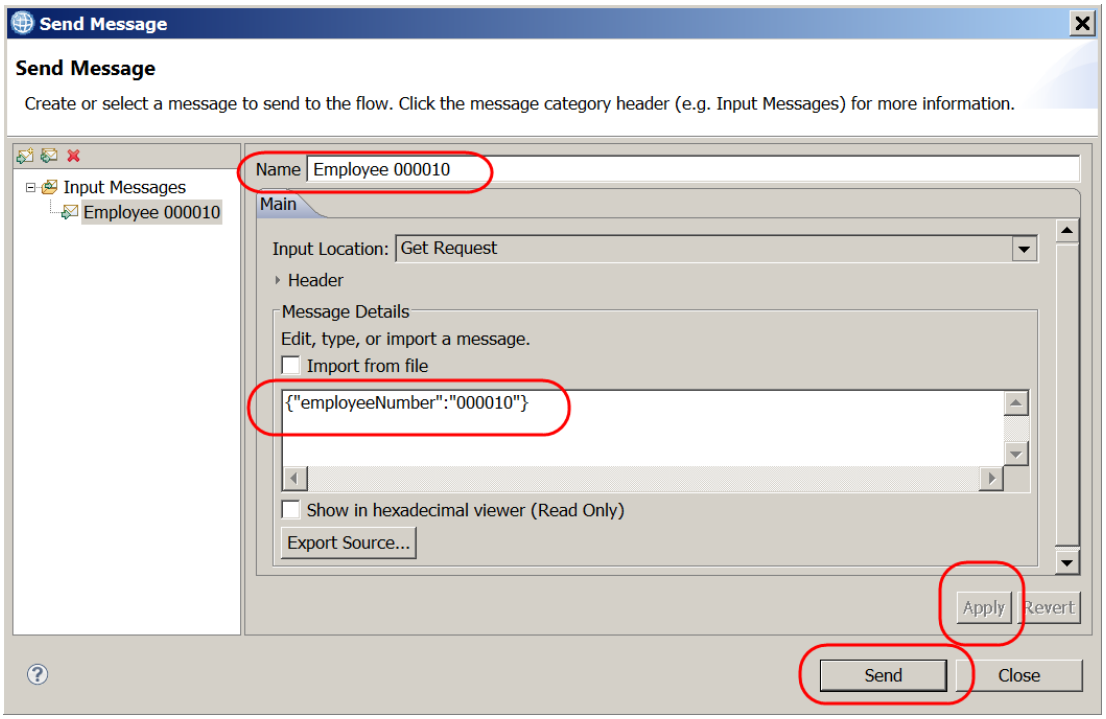
3. Select New Message:



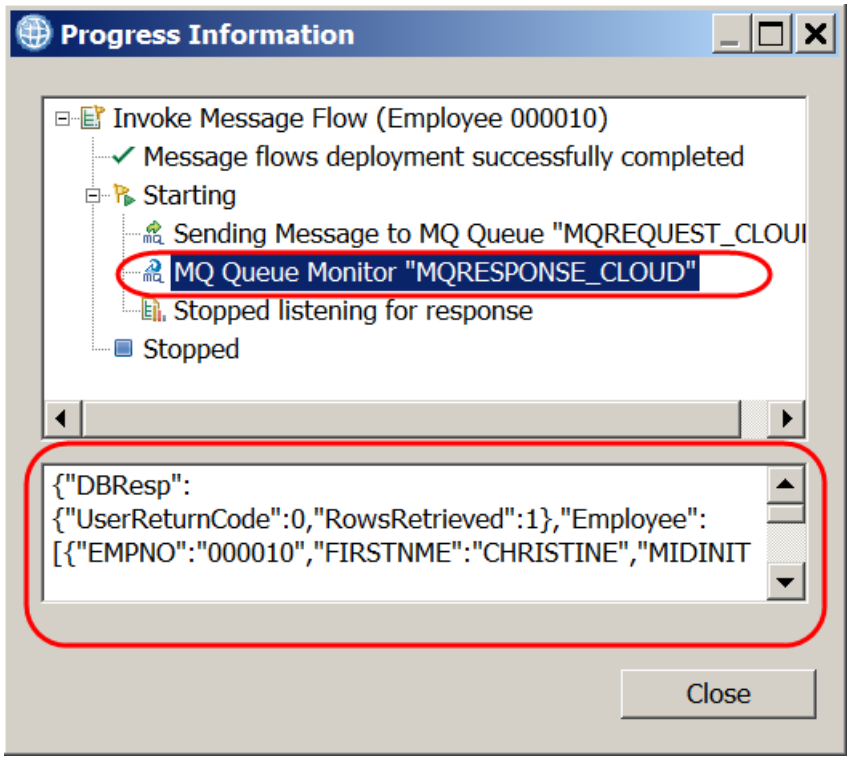
4. In the message details window, type the following

```
{"employeeNumber":"000010"}
```

Name the message Employee 000010 and click Apply, then Click Send:

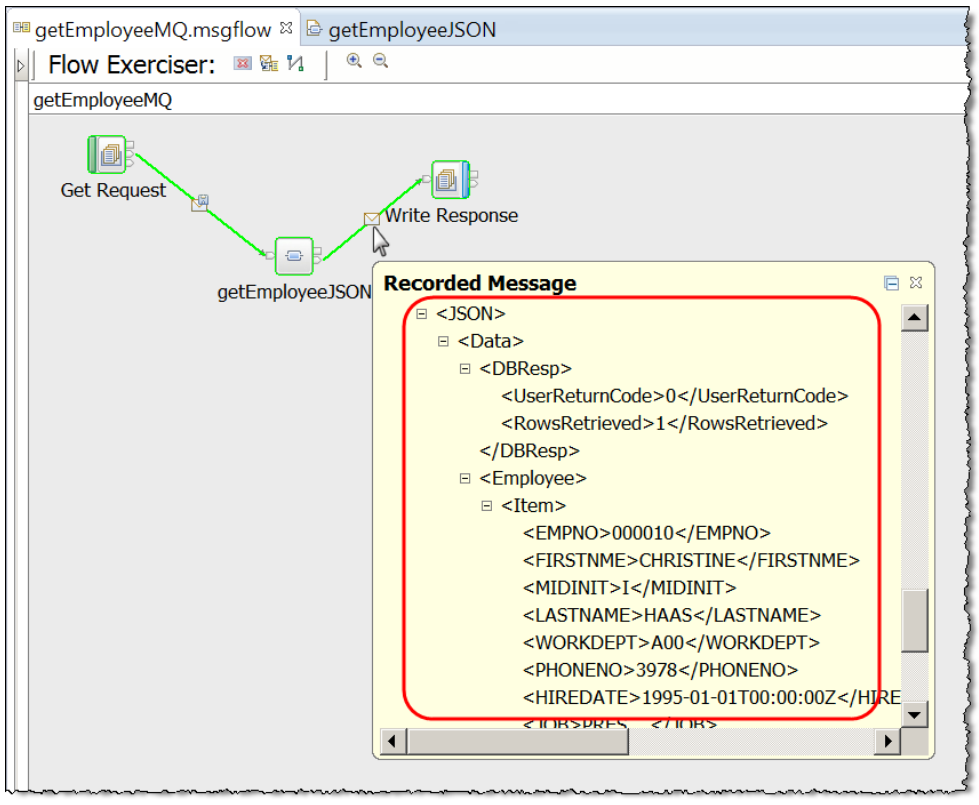


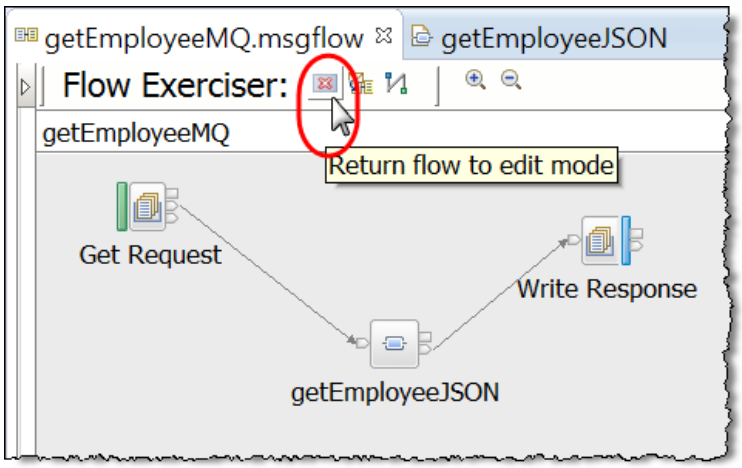
5. After a few seconds the Progress Information window will update and show the response has been processed. Click MQRESPONSE_CLOUD to see the JSON response to the database request (the details of employee 000010 – Christine Haas – will be shown in JSON format:



Close the Progress Information window when complete.

6. In the Flow Exerciser window the path the message took will be highlighted in green. Click the envelope icon to see the message details:



7.	Return the flow to Edit mode:  The MQ Provider application is now ready to be used with HR_Service.

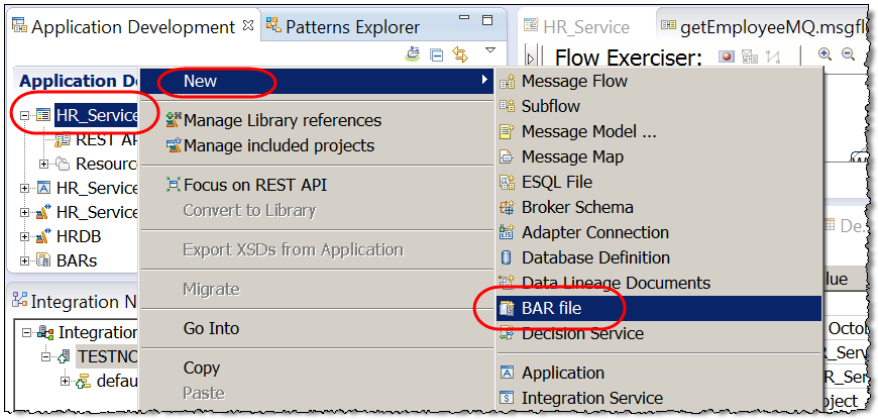
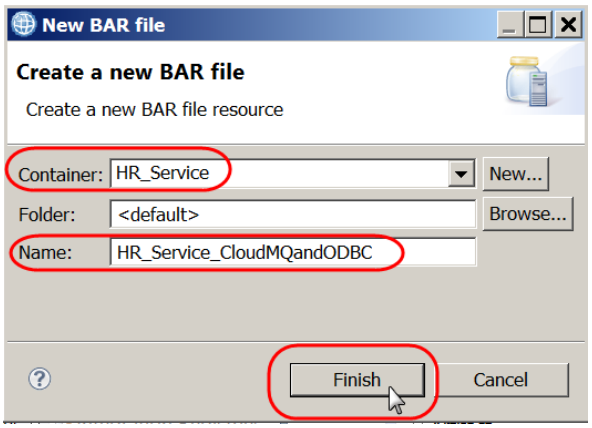
6. Run HR_Service Locally

You will now test that the implementation for **/employees/{employeeNumber}/MQEndpoint** is configured correctly when running **locally** on the same machine as **HR_Service_MQProvider**.

6.1 Prepare and deploy a bar file

When you test the REST API on IIB on Cloud you will need to upload a Broker Archive (bar) file to the cloud environment. In this next section you will create a Broker Archive file that you will use to:

a) test the REST API locally and
b) (after a rebuild) test the REST API running on IIB on Cloud.

1.	In the Integration Toolkit, right click on HR_Service and select New > BAR file: 
2.	Set the Container to HR_Service, name the file HR_Service_CloudMQandODBC and click Finish: 

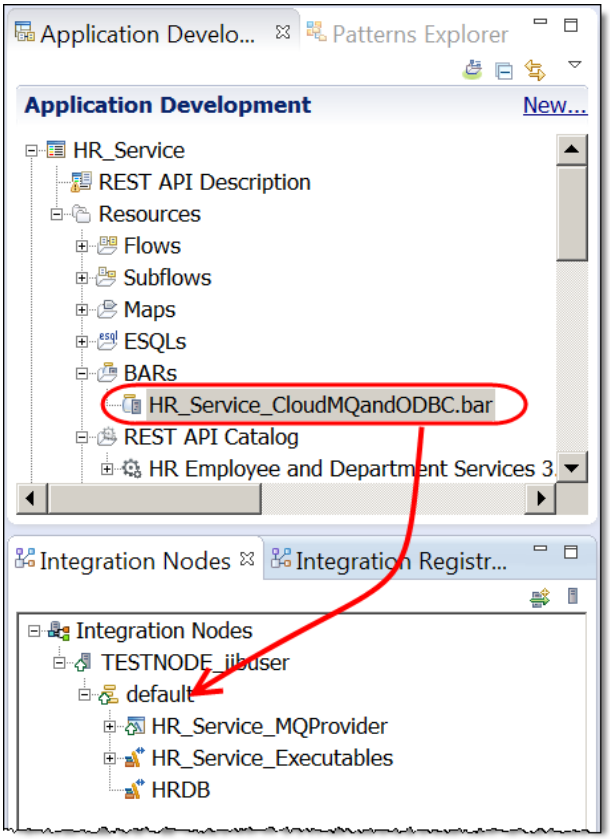
3. In the bar file editor, on the Prepare tab, select the HR_Service REST API and the two Shared Libraries: HR_Service_Executables and HRDB.

   You will use this bar file to test the REST API locally and on IIB on Cloud. When deploying the bar file on IIB on Cloud, if you do not include the Shared Libraries in the bar file, the deploy process will fail. Click the “Build and Save” button to save the bar file:

   The bar file will now be saved in the HR_Service REST API.

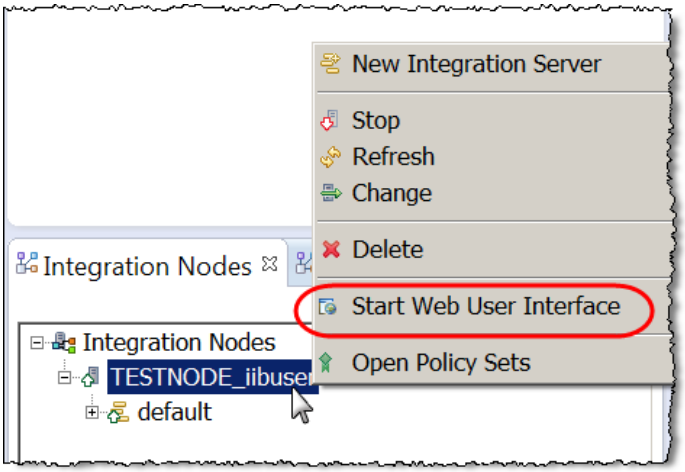
   Leave the Bar file editor open you will rebuild the bar file again later after you configure the MQ nodes to use an MQ Endpoint Policy.

4. Drag the bar file to the default Integration Server to deploy the bar file:



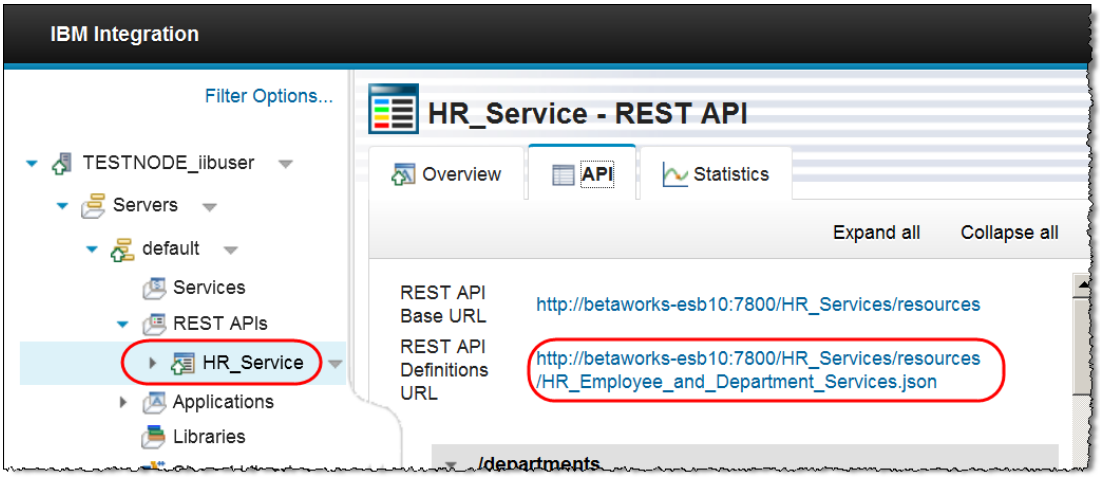
6.2 Test MQ Scenario using Swagger UI

1. Right click on TESTNODE_iibuser and select Start Web User Interface:

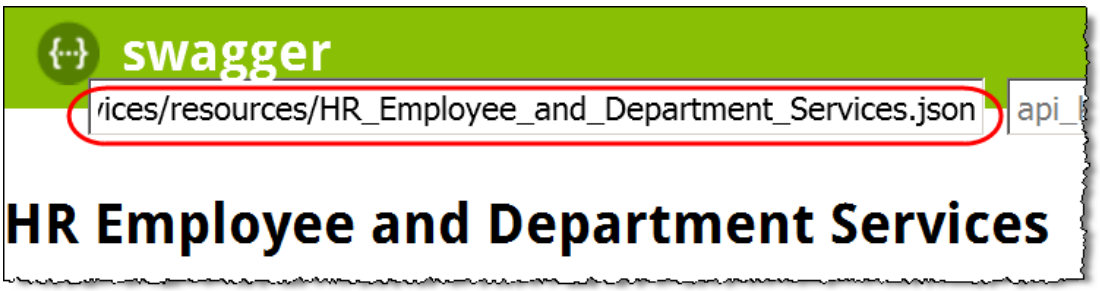


2. In the Web UI navigate to TESTNODE_iibuser > Servers> default> REST APIs and click on HR_Service.

   Copy the value of REST API Definitions URL:

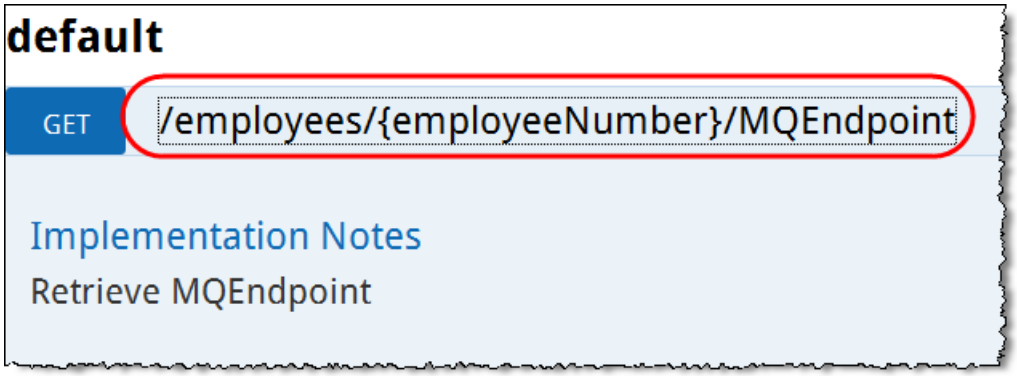


3. Open SwaggerUI (*if you are using the Workshop VM–select REST folder in Firefox*) and paste the URL of the REST API Definition that you just copied from the web UI into the URL field.



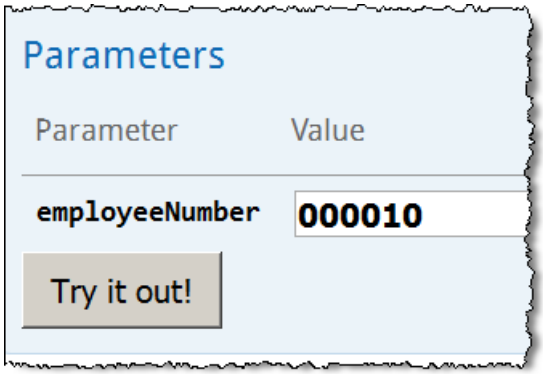
   Press enter to see the details of the REST API.

4. Expand default and select the GET for the MQEndpoint:



(*Note the resource is in "default" because at the time of writing this lab guide the IIB toolkit does not provide the ability to write "tags" in the swagger.json associated with the REST API*)

5. Locate the parameter employeeNumber and enter `000010` in the employeeNumber field and click the **Try it out!** button :



6. After a few seconds the REST API will respond with the details for employeeNumber 000010:

In the next section of this Lab guide you will run the REST API HR_Service on IIB on Cloud and access the same local MQ and DB resources.

7. Running HR_Service on IIB on Cloud

In this part of the lab you will use the IIB on Cloud user interface to deploy HR_Service to IIB on Cloud. You will then run the resources

/employees/{employeeNumber}/MQEndpoint
and
/employees/{employeeNumber}/CloudODBC

to demonstrate how MQ and DB2 resources installed locally on site can be accessed from the IIB on Cloud Managed Service.

If you require access to IIB on Cloud follow this link to find details on how to register:

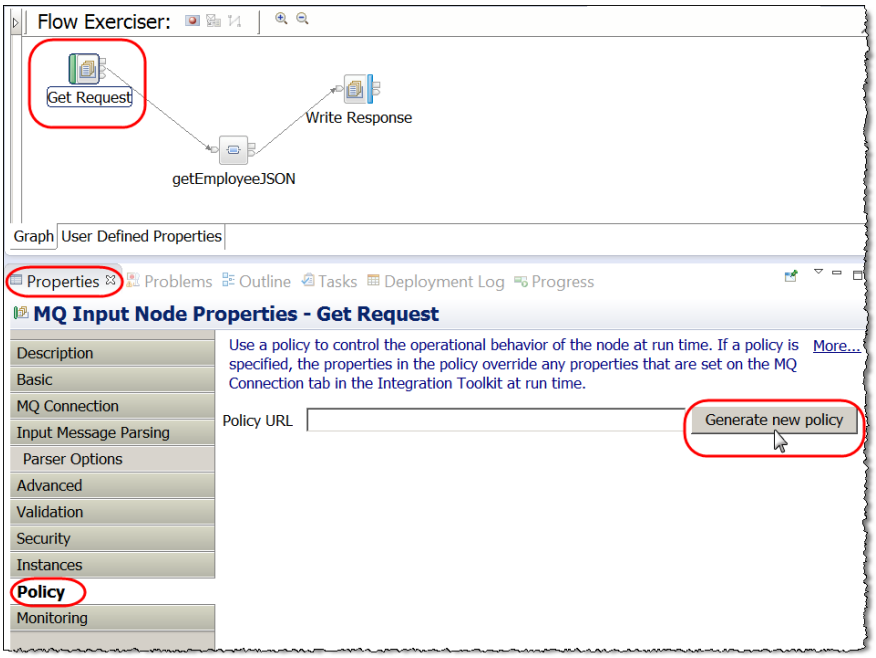
https://www.ibm.com/ibmid/basic_register/register_generic.html?a=IBMSystems&ctx=IIB&cc=us&lc=en&catalogName=Master&partNumber=IIB_Trial&quantity=1&trial=yes&S_TACT=C4310CSW

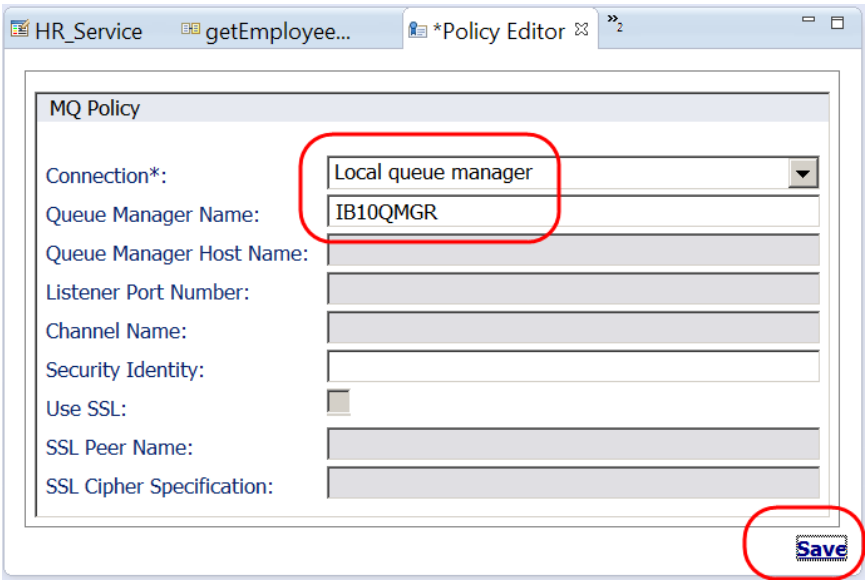
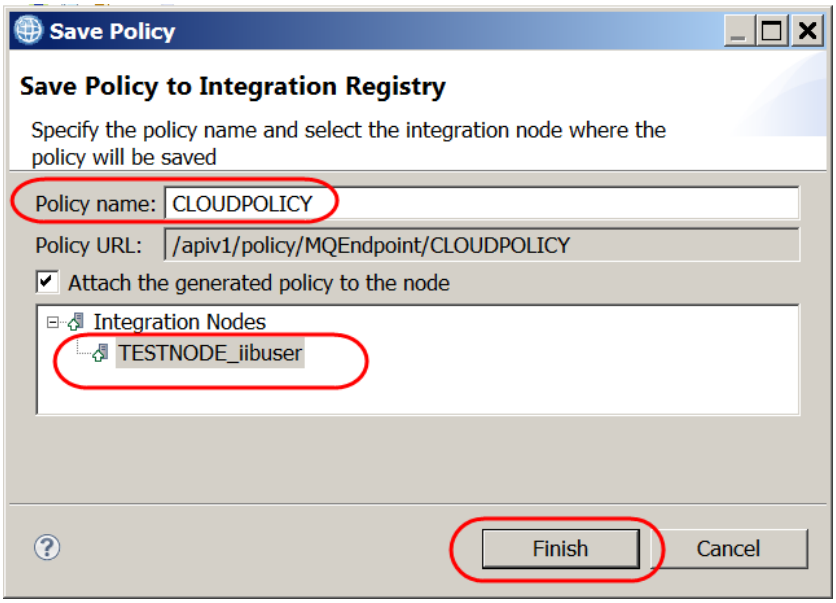
The remaining part of this lab guide requires access to an active IIB on Cloud account.

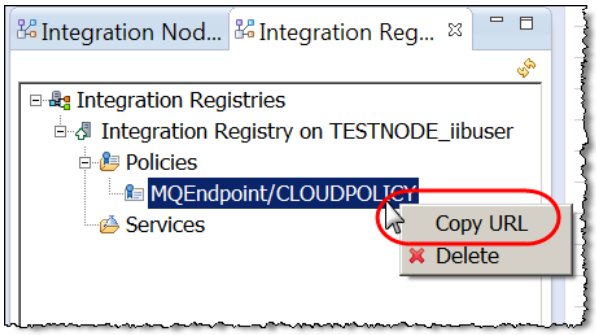
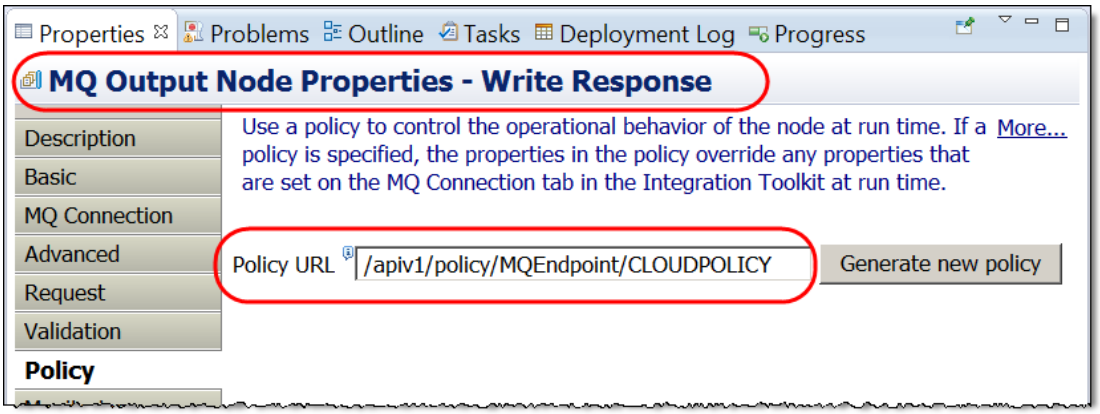
7.1 Configure MQ Endpoint Policy

In the previous sections you configured and tested the HR_Service locally using the Flow Exerciser. The names of the MQ queues and queue manager were explicitly configured on the MQ nodes.

In this next section you will reconfigure the MQ nodes to use an MQ Endpoint Policy. MQ Endpoint policies must be used when running MQ nodes on IIB on Cloud (they cannot be used when testing locally using the Flow Exerciser).

1.	In **getEmployeeMQ** message flow (*in HR_Service_MQProvider*).
2.	Select the Get Request (MQInput node) and click Generate new policy (Properties/Policy): 

3.	Configure the MQ Policy as follows, then click Save: 
4.	Call the policy “CLOUDPOLICY” and attach it to TESTNODE_iibuser. Click Finish: 
5.	Click OK to dismiss the Policy Saved message.

6.	The Policy is now saved in the Integration Registry on TESTNODE_iibuser. *Although it is saved there, IIB on Cloud will not use it locally, however the bar file that you deploy to IIB on Cloud needs to be configured to use and MQ Endpoint Policy.* In the Integration Registry window, right click on the policy and select Copy URL: 
7.	In the **getEmployeeMQ** message flow, paste the url into the **Policy URL** field (Policy tab) for the MQ Output Node called Write Response: 
8.	Save the Message flow (ctrl s)

7.2 Rebuild the bar file

The bar file that you will deploy to IIB on Cloud needs to have the MQ Endpoint Policy configuration. In this next section you will rebuild the bar file ready to deploy on IIB on Cloud.

1. In the bar file editor (you should have the bar file editor open already from the previous bar file prepare), make sure the following are selected:

   (REST APIs) **HR_Service**
   (Shared Libraries) **HR_Service_Executables** and **HRDB**

   Click the “Build and Save” button to rebuild and save the bar file:

   The bar file with the MQ Endpoint configuration will now be saved in the HR_Service REST API.

7.3 Deploy HR_Service to IIB on Cloud

In this next section you will deploy this updated bar file to IIB on Cloud.

1. If you are using the BetaWorks IIB Workshop VMware image, open the IIB on Cloud link in the Firefox window (IIB on Cloud in the Cloud folder).

   Enter the details of your ‘IBM Id’/password authorised for IIB on Cloud and click ‘**Sign In**’.

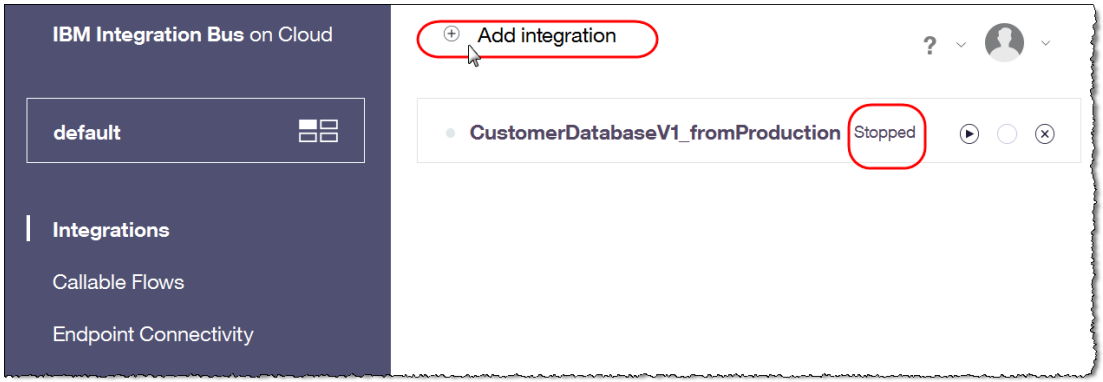
   (*If you are using your own system sign on to IIB on Cloud using this URL:* https://ibm-cloud-ui.ibmintegrationbus.ibmcloud.com/)

2. You now have your IIB on Cloud default integration space opened.

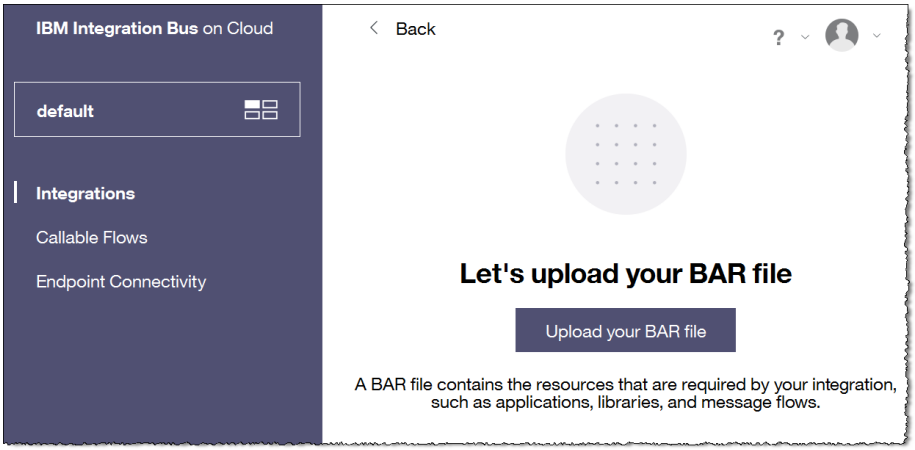
   If you have done the introduction lab for IIB on Cloud from this series you will have the sample integration CustomerDatabaseV1 shown in this view.

   The trial version for IIB on Cloud allows only one integration to run at one time. Therefore, if you have previous Integrations running, please make sure that they are stopped.

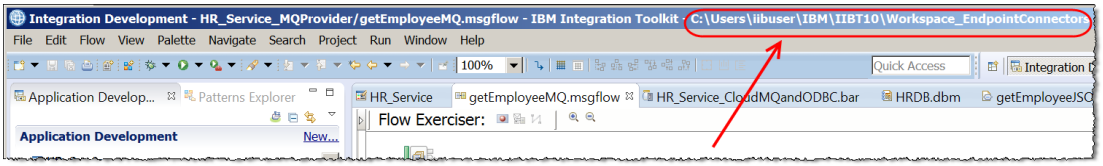
   You will now upload the REST API integration that you created in earlier in this series. Click '**Add Integration**'.

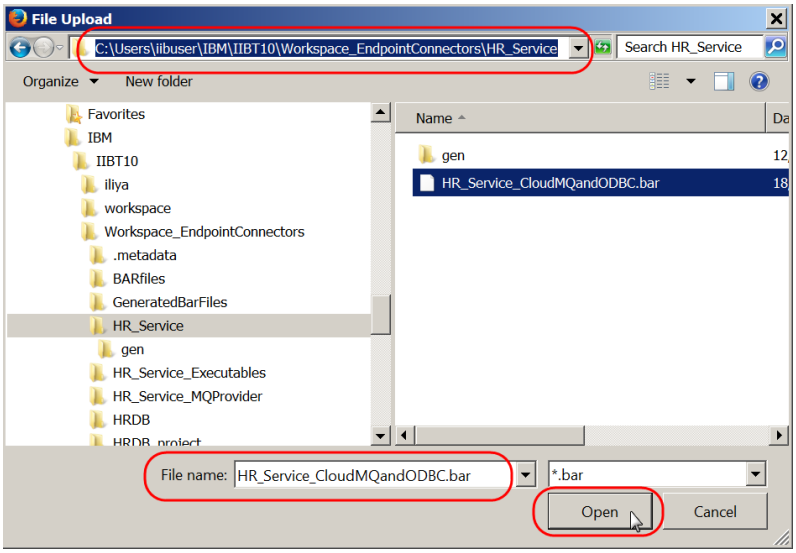
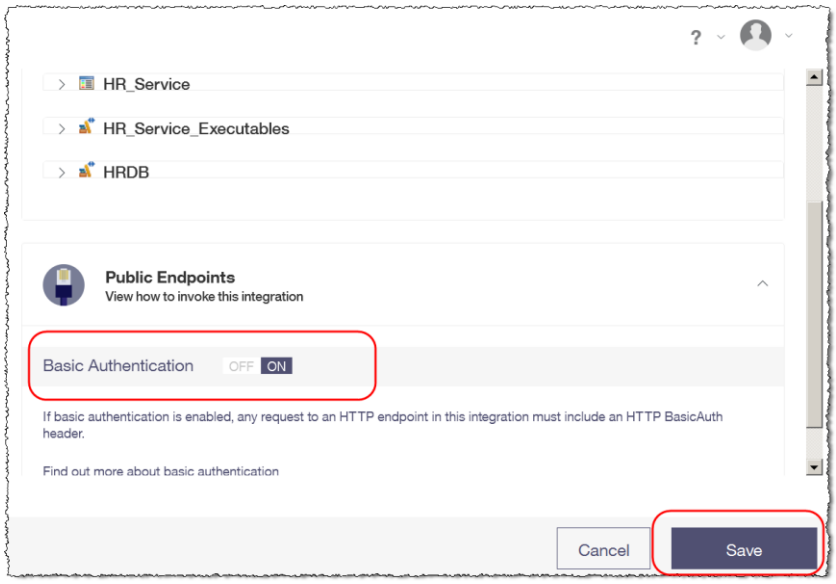


3. Click '**Upload your BAR file**'.



4. In the Integration Toolkit, make a note of the file system path of your workspace (it will appear in the header of your toolkit window:



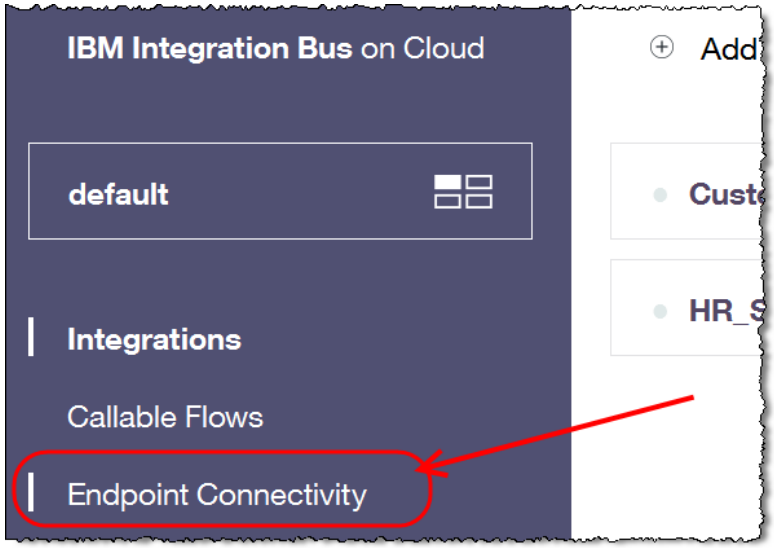
5.	In the File upload (browser) window, navigate to the HR_Service folder in your workspace (where you saved the bar file that you tested in the previous section), select the bar file and click Open: 
6.	IIB on Cloud will validate the bar file and then show details about the REST API.
7.	As the REST API is not running, you can prepare a limited set of features. By default Basic Authentication is set to ON. Switch this to OFF (click OFF), and click Save: 
8.	The Integration will then show as '**Preparing**'. This may take a few minutes. Click the "Refresh Listing" at the bottom of the screen to update the page with the latest status of the Integration. The status will change to Stopped:  Whilst the Integration is preparing, you can proceed with the rest of the lab.

7.4 Configure IIB on Cloud to connect to local MQ and DB2

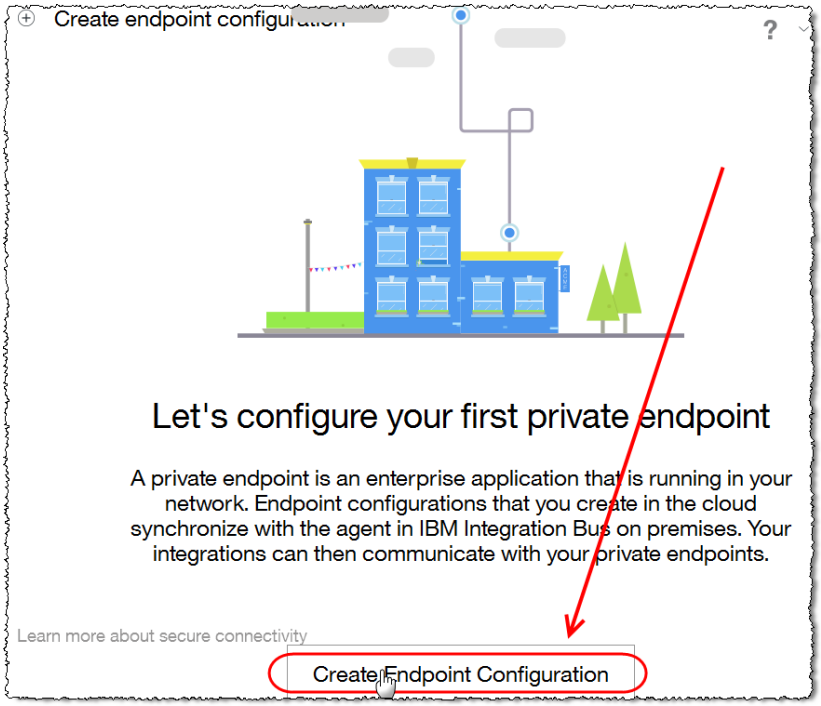
In this part of the lab, you will configure how HR_Service will access your local MQ and DB2 environment using Endpoint Connectivity. The resources you will test in HR_Service will access local MQ queues and access your local HRDB DB2 database directly from IIB on Cloud. It will access these local resources using the IIB Secure Connector.

7.4.1 Define Endpoints

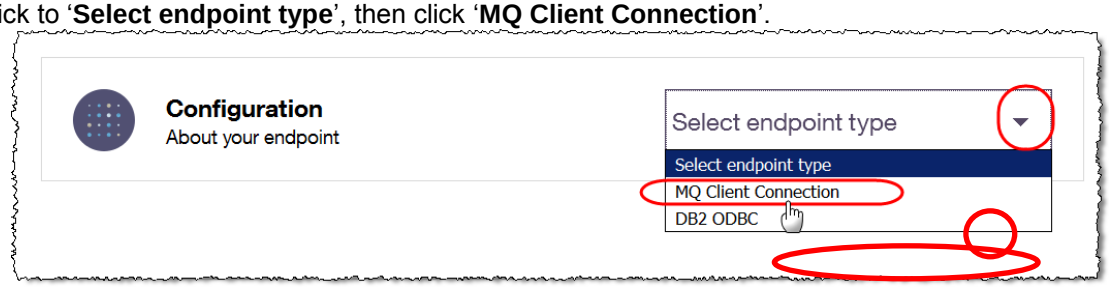
1. Click **Endpoint Connectivity**:



2. If this is the first time you have set up an Endpoint configuration, you will see the following, Click '**Create Endpoint Configuration**'.



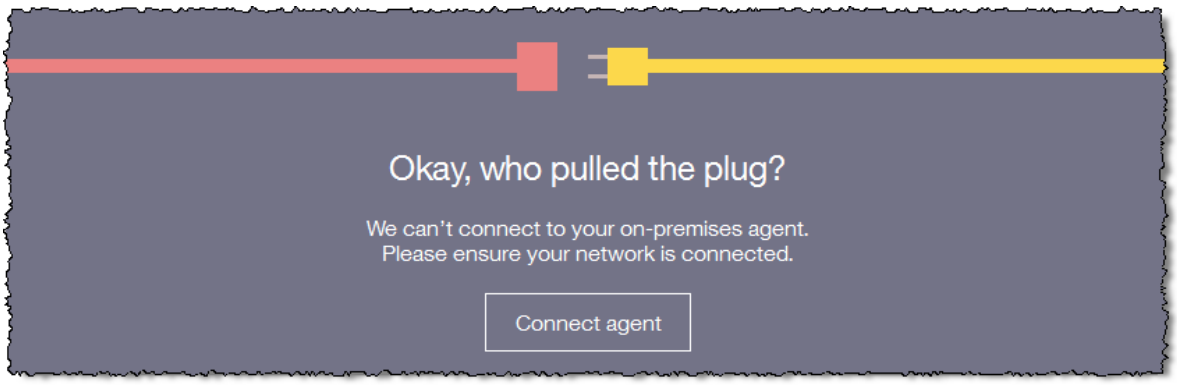
3. Click to '**Select endpoint type**', then click '**MQ Client Connection**'.

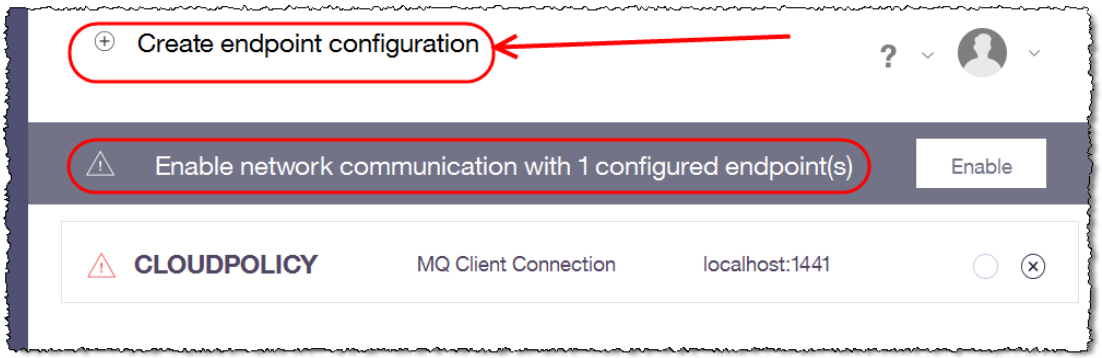


4. Configure the MQ Endpoint as follows, click Save when complete:

- MQ policy name – **`CLOUDPOLICY`**
  (this is the MQ Endpoint policy that is specified on the MQ nodes in the getMQEndpoints subflow in HR_Service)
- Host - **localhost**
- Port - **1441**
- Queue manager – **IB10QMGR**
- Channel name – **SYSTEM.DEF.SVRCONN**

5. Once the configuration has been saved you will see a screen, listing the Endpoint you just created. If you see a message saying "Okay, who pulled the plug" the IIB on Cloud system thinks that you already have an IIB Switch agent configured on your machine. For the purposes of the next section you can ignore this message.

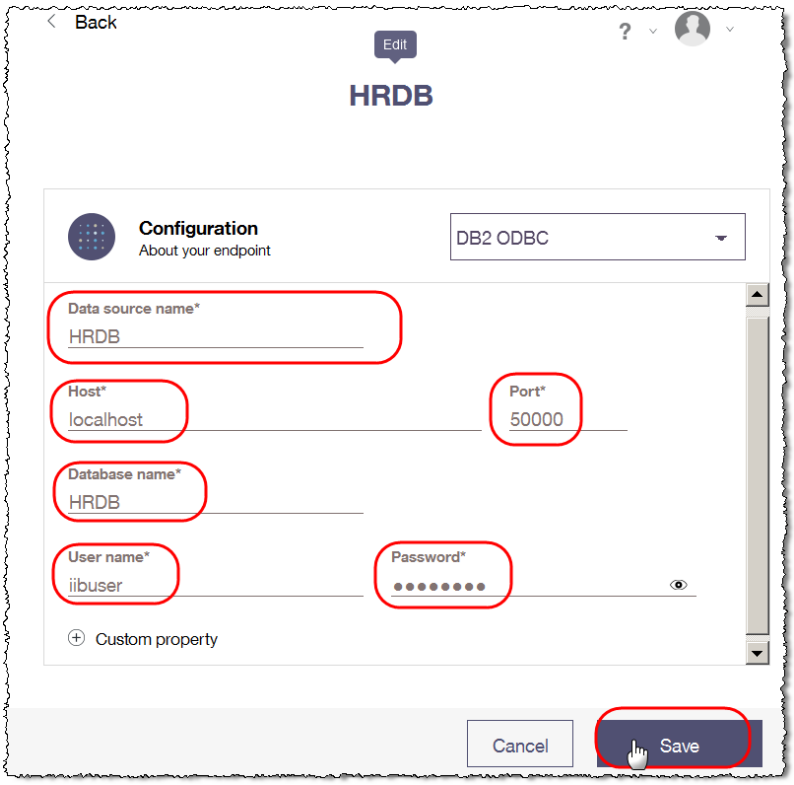


6.	If you have previously had an IIB switch Agent configured on using your IIB on Cloud ID, you will see the following screen. Ignore the message to “Enable network communication with 1 configured endpoint (s) ”. For this lab guide scenario, you will need to create one more configuration – for DB2.  Click ‘**Create endpoint configuration**’.
7.	From the drop-down menu select ‘**DB2 ODBC**’. 
8.	Similar to the MQ endpoint set up, the ‘**DB2 ODBC**’ requires information about the connection to the DB2 database in your environment.

9. Complete the fields, entering the details as below (corresponding to your DB2 installation on the VM):

   - Data source name - `HRDB`
   - Host - `localhost`
   - Port - `50000`
   - Database – `HRDB`
   - User name – `iibuser`
   - Password – `passw0rd`

   Once complete, click the '**Save**' button:



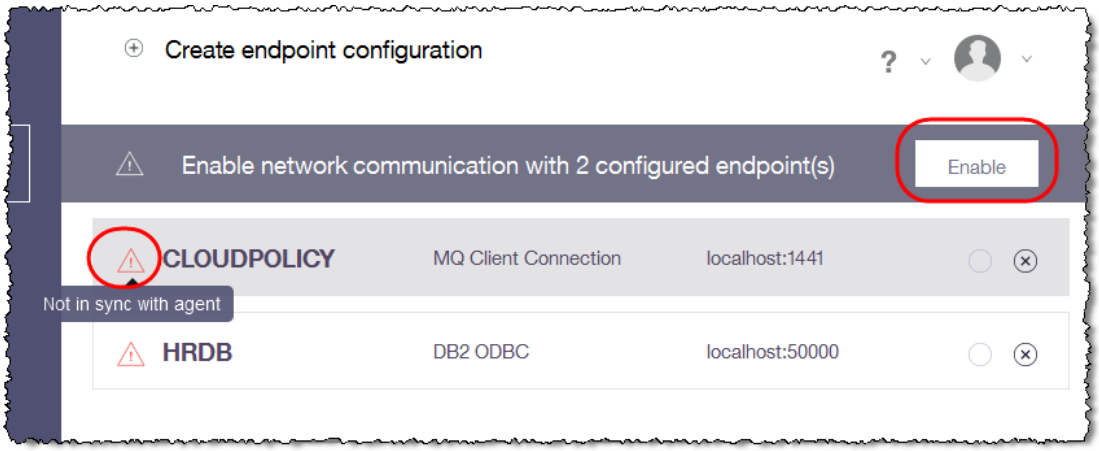
10. A list of the two Endpoints you have just configured will be shown.

    Note the red triangle warning icon on both endpoints, hovering over this icon shows the message "Not in sync with agent".

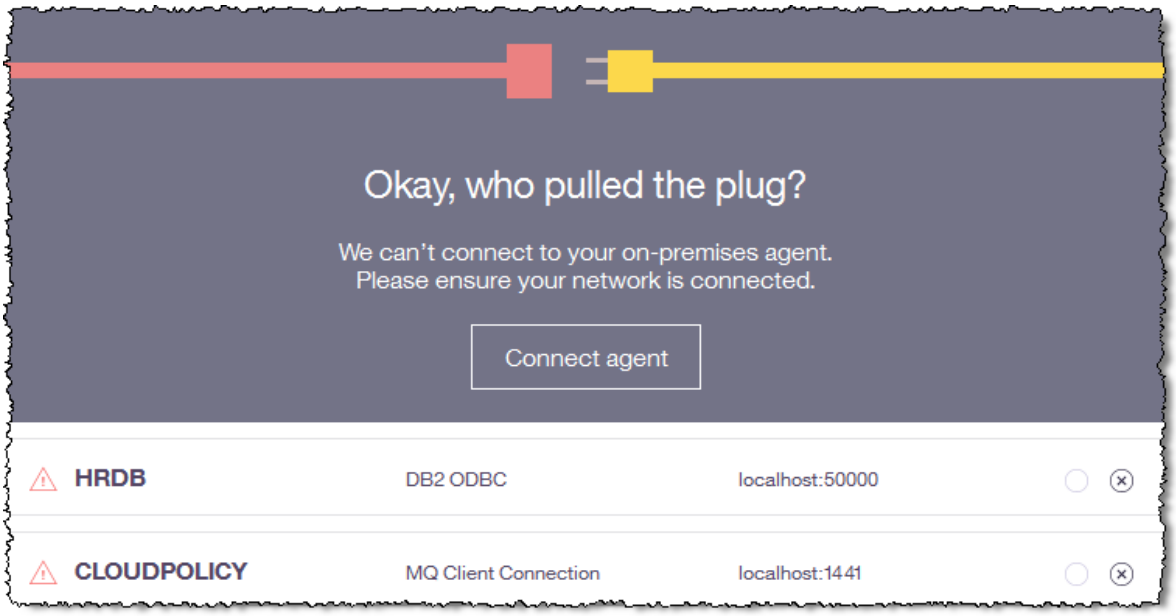
    Both endpoints require an IIB Secure Connector (an "agentp") to be configured before they can successfully connect to your local MQ and DB2 environments. The list of tasks is slightly different dependent on whether you have previously configured an IIB Secure Connector. Choose only ONE of the next two sections

7.4.2 Determine IIB Secure Connector configuration

1. If you have not previously configured an agent on IIB on Cloud, click on '**Enable**' and follow the rest of this chapter:



If you have previously configured an IIB Secure Connector, you will see the following "Okay, who pulled the plug?" message. Follow the tasks in "**Set up a previously configured IIB Secure Connector**":



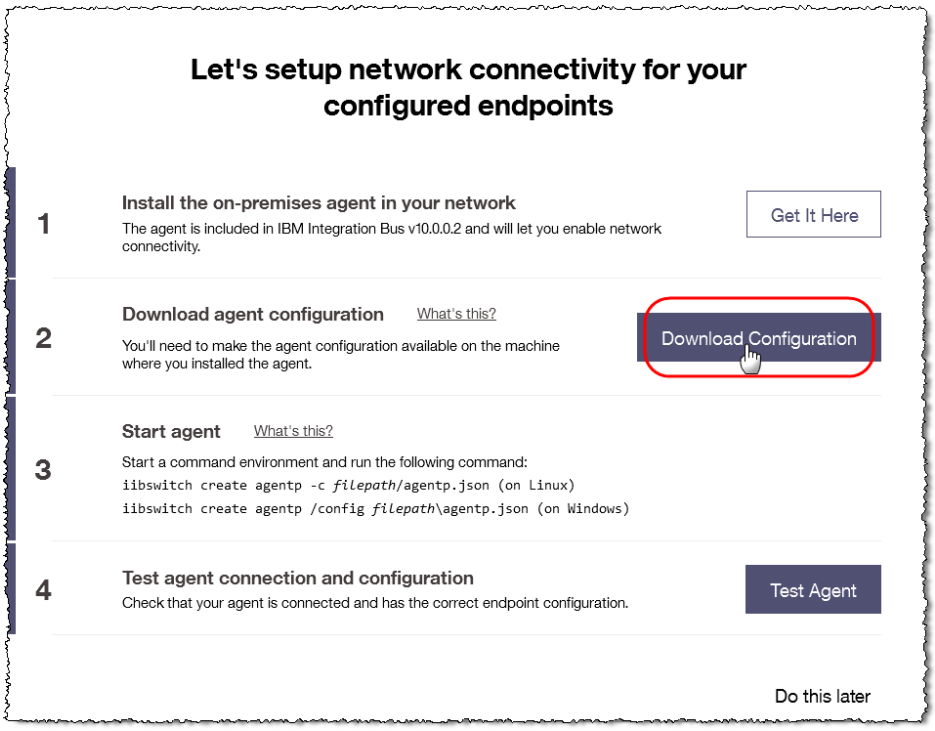
7.4.3 Option1: Set up a NEW IIB Secure Connector

Follow this section <ONLY> if you are configuring an IIB Secure Connector for the first time in your IIB on Cloud ID.

Once you click on 'Enable' (previous step) you will see a pop up, which shows the steps required for setting up the local agent that will act as the local (on premise) end of the Secure Connector.

The Secure Connector is shipped with IIB v10.0.0.2 (and higher). Configuration of the agent is now required.

1. Select '**Download Configuration**' (number 2 in the list):



2. When prompted save the **agentp.json** file in a folder called `c:\IIBonC\`

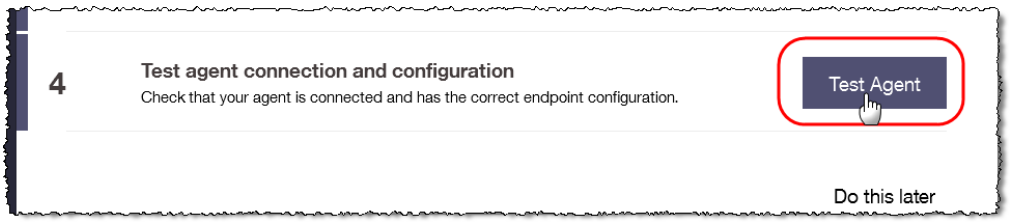
3. In an Integration Console, type the following command:

```
iibswitch create agentp /config C:\IIBonC\agentp.json
```

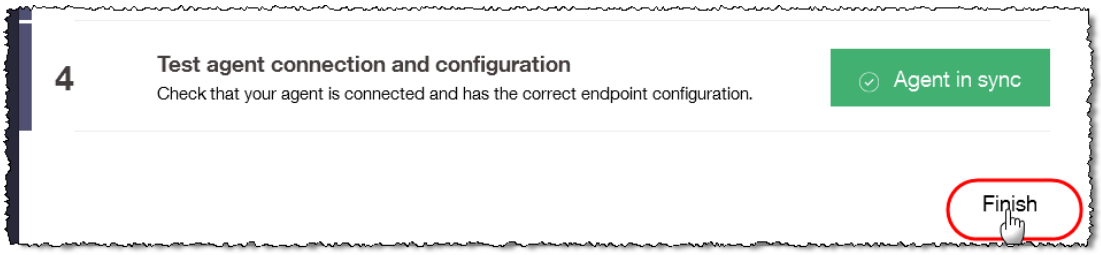
You will see the following response when the agent has been successfully created and started:

```
Creating iibswitch component 'agentp', please wait...
iibswitch created and started.
```

4. Back in the browser where you are signed on to IIB on Cloud, click '**`Test Agent`**' (step 4) to establish connectivity with the local agent from IIB on Cloud:



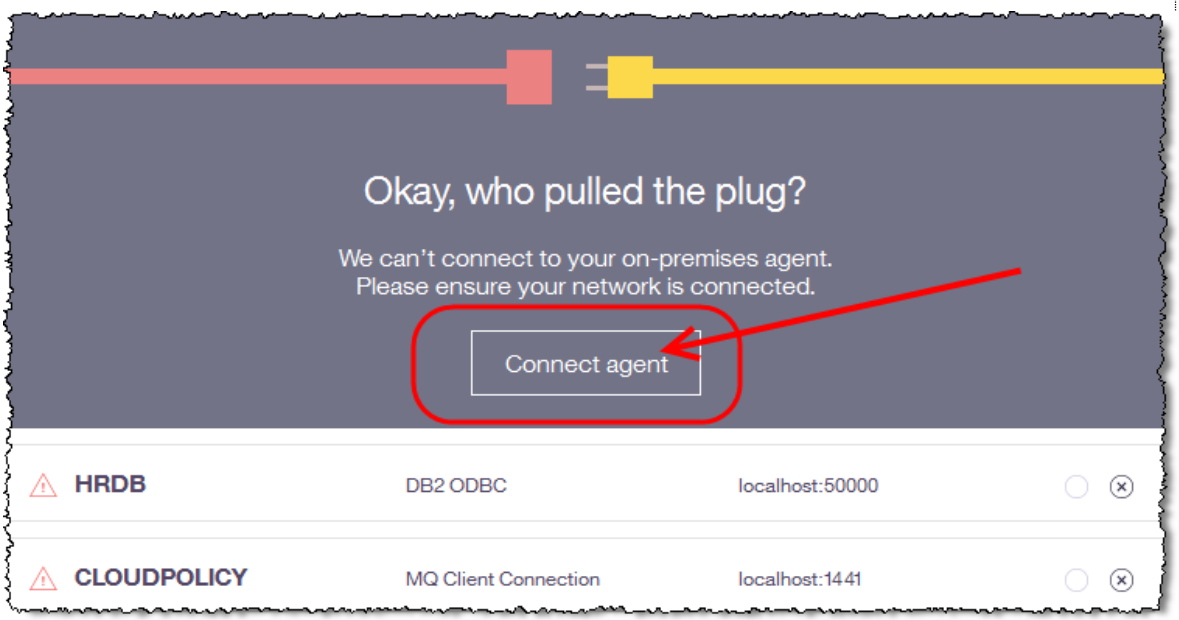
5. When the agent has successfully connected, you will see the message '**Agent in sync**' in a green box, Click Finish:



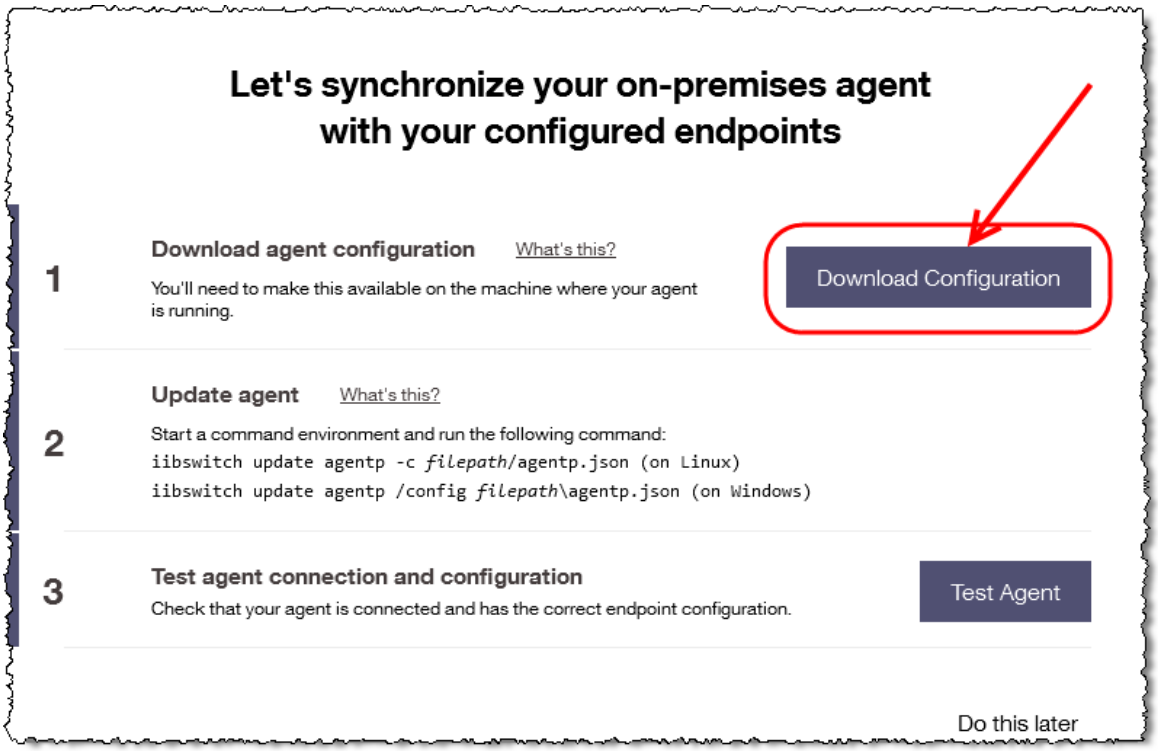
7.4.4 Option2: Set up a previously configured IIB Secure Connector

Follow this section <ONLY> if you have a previously configured IIB Secure Connector in your IIB on Cloud ID. If you have configured a NEW IIB Secure Connector (Option1 above), go to the Next section.

1. Click Connect agent:



2. For an existing configuration 3 tasks appear, select '**Download Configuration**' (number 1 in the list):



3. When prompted save the **agentp.json** file in a folder called `c:\IIBonC\`

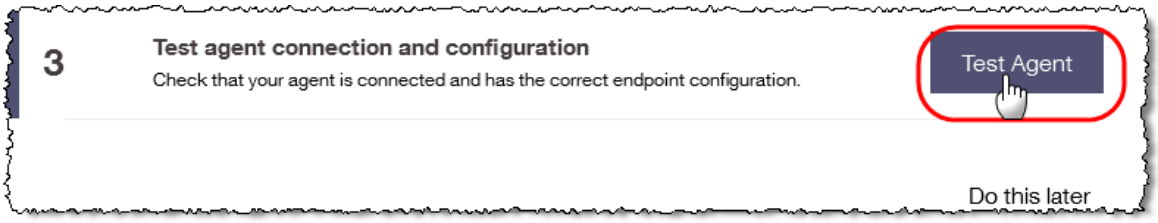
4. In an Integration Console, type the following command (this is option 2 in the list of tasks) :

```
iibswitch update agentp /config C:\IIBonC\agentp.json
```

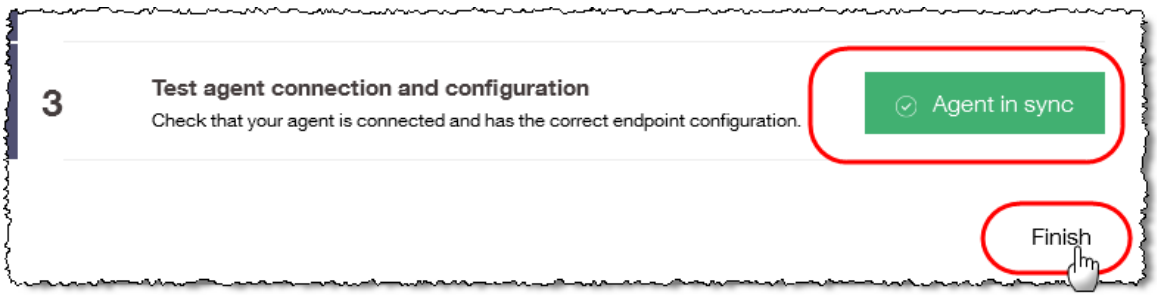
You will see the following response when the agent has been successfully created and started:

```
Updating iibswitch component 'agentp', please wait...
iibswitch updated.
```

5. Back in the browser where you are signed on to IIB on Cloud, click '**`Test Agent`**' (step 3) to establish connectivity with the local agent from IIB on Cloud:



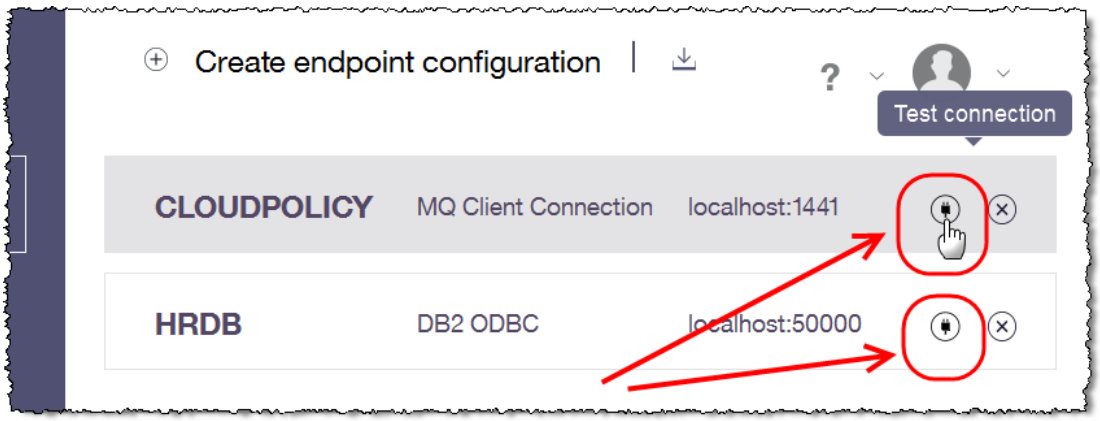
6. When the agent has successfully connected, you will see the message '**Agent in sync**' in a green box, Click Finish:



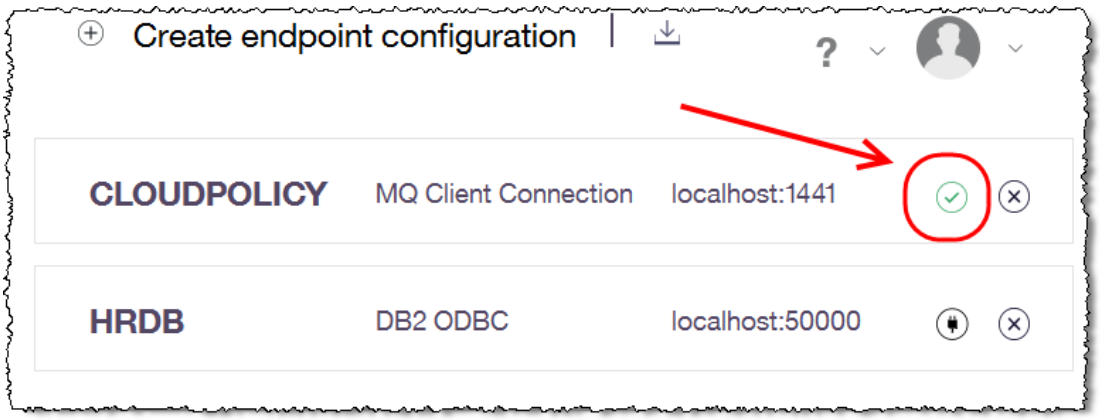
7.4.5 Confirm MQ and DB2 Endpoint configuration

At this point you will have either created (or updated) the IIB Secure Connector that IIB on Cloud will use to communicate with your local system. In the next section you will test connectivity from IIB on Cloud to your local MQ and DB2 Systems.

1. In the list of Endpoints you will see the two Endpoints – MQ and DB2. On the right, click on the "Test Connection" icon for each Endpoint (*the small 'plug'-like icon*).



2. After a few seconds the Plug icon will turn to a green tick mark (confirming that IIBoC can communicate with MQ and DB2 On-Premise). The green tick mark will revert back to a plug sign after a few seconds:



8. Test HR_Service running on IIB on Cloud

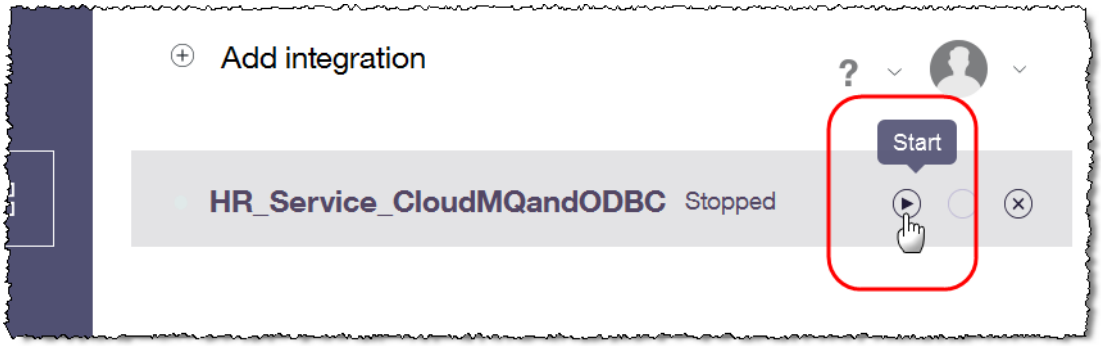
In this part of the lab you will test your IIB on Cloud integration. HR_Service will connect to your local MQ and DB2 systems using the Endpoint configurations you have just defined. Two HR_Service resource URLs will be used to obtain information from the local HRDB database running on your machine.

1. **/employees/{employeeNumber}/MQEndpoint**
   Testing the MQ Endpoint connection from IIB on Cloud to your local system, the request will be written to your local MQ environment where the HR_Service_MQProvider application will process it and write a response to a local queue. When written HR_Service will receive the response and pass back to the application calling the REST API.

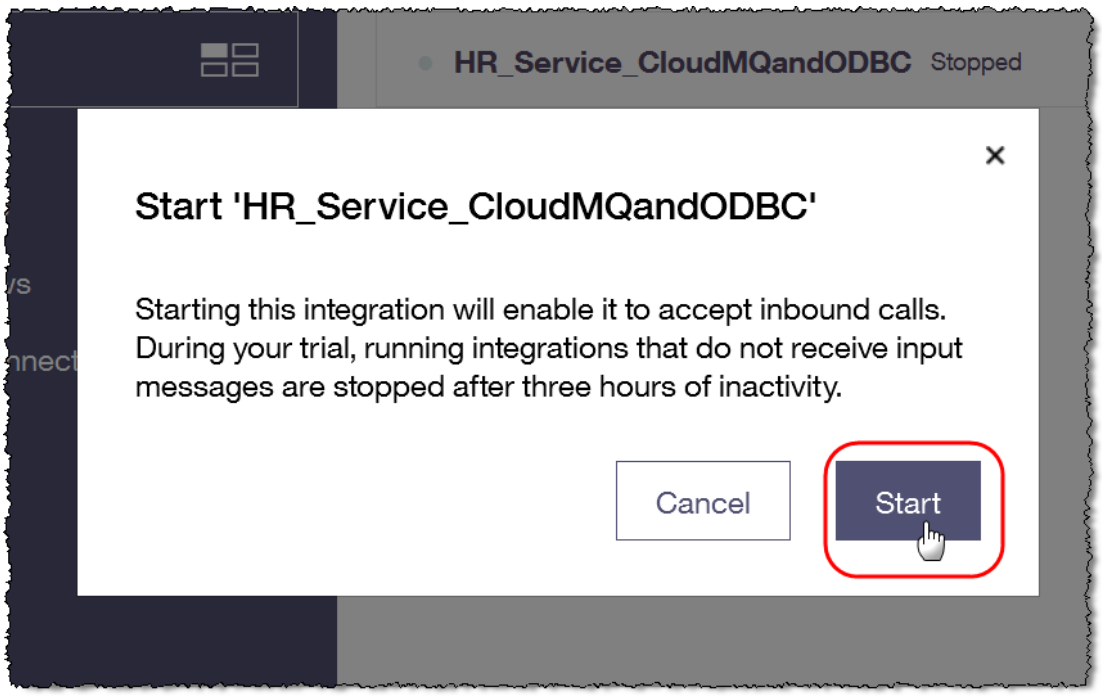
2. **/employees/{employeeNumber}/cloudODBC**
   This HR_Service resource will use the ODBC endpoint definition to access HRDB directly from IIB on Cloud (without accessing any local MQ environments).

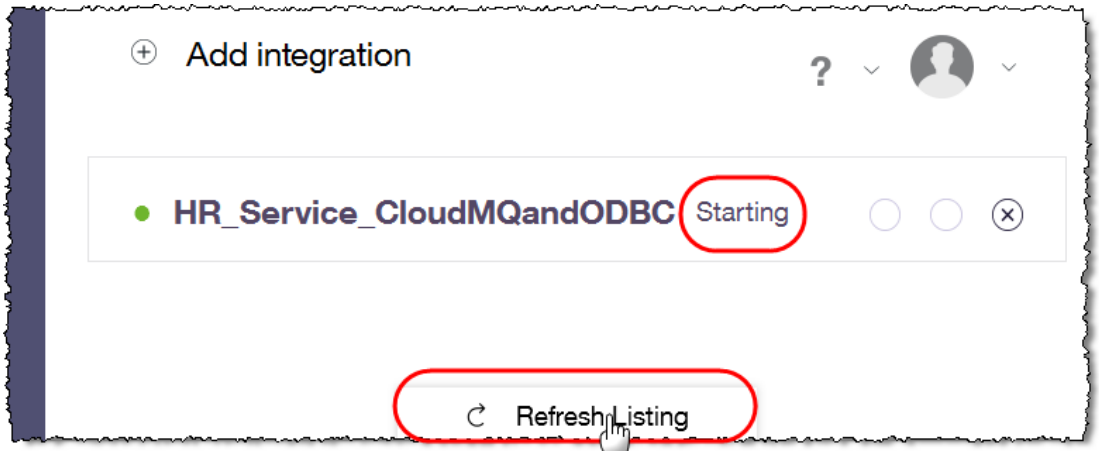
8.1 Start your IIB on Cloud integration

1. In the IIB on Cloud Web User Interface, click **Integrations**, then start **HR_Service_CloudMQandODBC:**

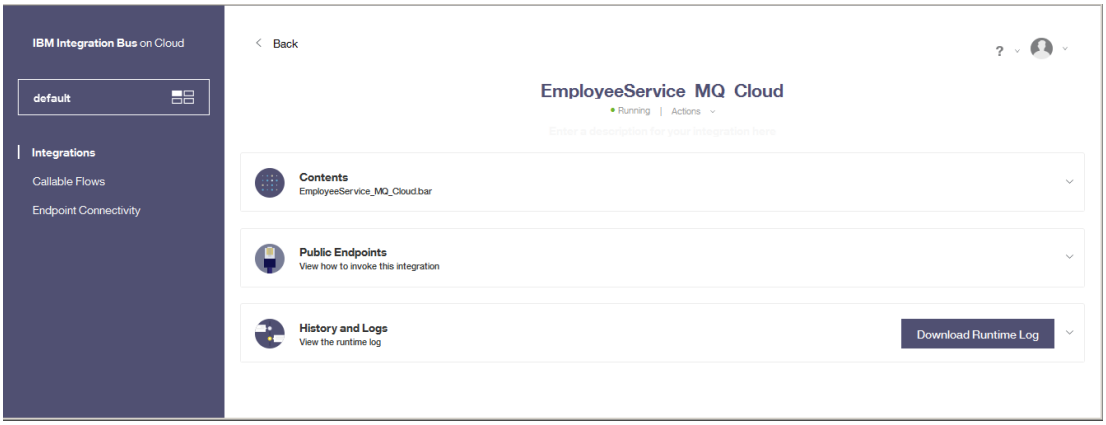


2. Click Start:



3.	The web User Interface will show that the Integration is starting. It may take a minute or two for the integration to start completely and you can check its status by click on '**Refresh Listing**'. 
4.	When it has started, the User Interface will be updated, showing the Integration '**Running**'. 

8.2 Test the MQEndpoint integration

1.	Click on **EmployeeService_MQ_Cloud** integration.  Here, you can see the integration **Content**, its **Public Endpoints** and **History and Logs**. Click '**Public endpoints**'.

2. When the service is running click HR_Service_CloudMQandODBC.

   Select Public Endpoints, then locate the Service URL for the MQEndpoint resource, Click the "Show full URL link:

3. Copy the full URL (Ctrl c)

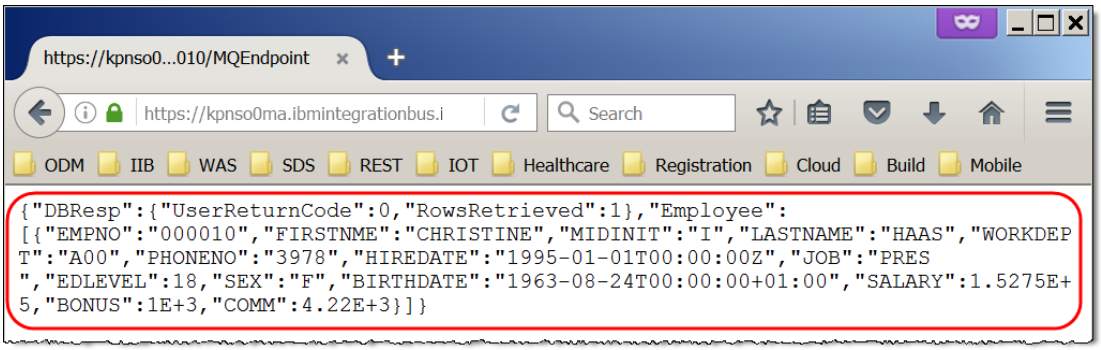
   In this example here is the URL:

   Keep the browser open at this section you will need it later.

4. Copy the url into a separate browser window. Replace **{employeeNumber}** in the URL with the text `000010` for example:

   Press enter.

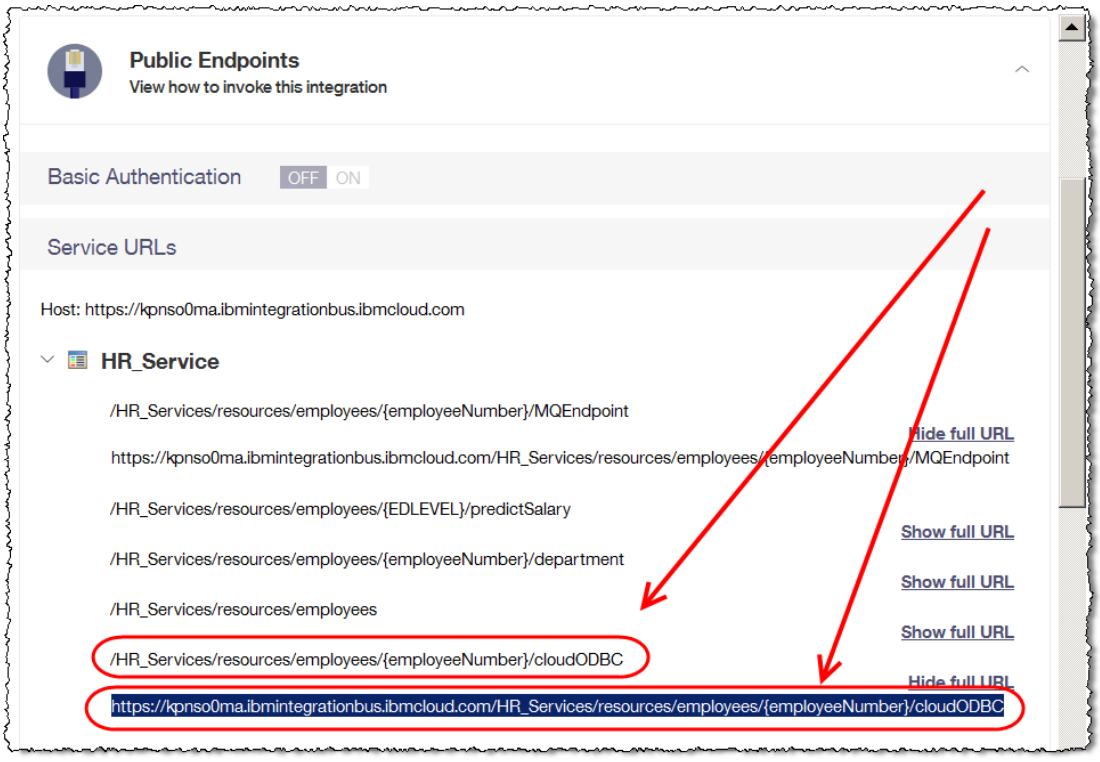
5. After a few seconds the REST API will respond with the details for user 000010 from the HRDB database:



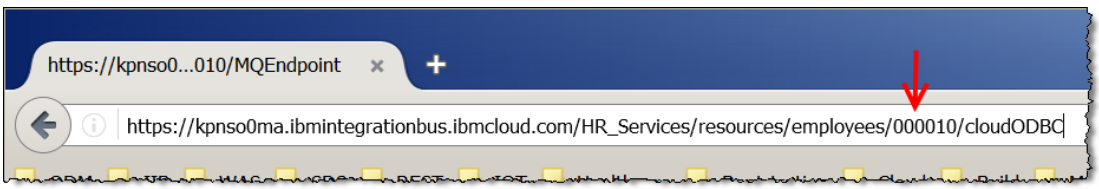
8.3 Test the DB2 (ODBC) Endpoint

1. In the IIB on Cloud browser window, locate the URL for the cloudODBC resource (*it will be in the same list as the MQEndpoint that you copied in the previous section*)

   You will already be in “Public Endpoints”, locate the Service URL for the resource ending in cloudODBC, Click the “Show full URL” link and copy the URL:

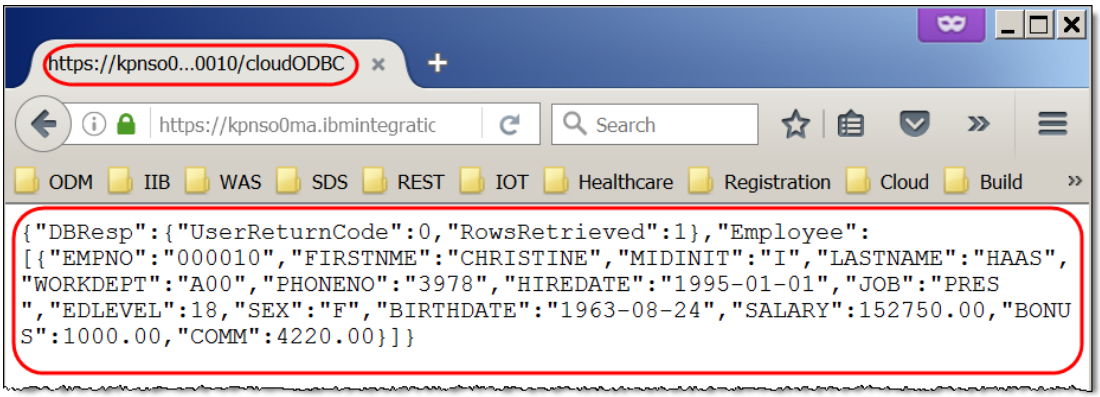


2. As before, copy the url into a separate browser window. Replace **{employeeNumber}** in the URL with the text `000010` for example:



   Press enter.

3. After a few seconds the REST API will respond with the details for user 000010 from the HRDB database located on your machine:



Appendix

Recreating the HRDB database and tables

The HRDB database, and the EMPLOYEE and DEPARTMENT tables have already been created on the supplied VMWare image. If you wish to recreate your own instance of this database, the commands
`1_Create_HRDB_database.cmd` and
`2_Create_HRDB_Tables.cmd` are provided for this. If used in conjunction with the VM image, these commands must be run under the user "iibadmin". Appropriate database permissions are included in the scripts to GRANT access to the user iibuser.

END OF LAB GUIDE