IBM®

βetaWorks

# IBM Integration Bus

# Developing a REST API
# *(Without an existing swagger.json)*

Featuring:

The REST API tools for IIB
Implementing REST Post and Get operations
REST operation parameter aware mapping
Conditional if, then, else transforms in a Mapping node
Testing REST with SwaggerUI

**September 2016**
Hands-on lab built at product
Version 10.0.0.6

# 1. Introduction and Preparation

## 1.1 Introduction

In this lab you will create a new REST API <without> using a previously created Swagger document. Integration Toolkit features now available in IIB V10 fix pack 6 provide a new feature to create a REST API from scratch. A Swagger document is still used when creating a REST API; however the file will be automatically created (and maintained) by the IIB Toolkit when using these new features.

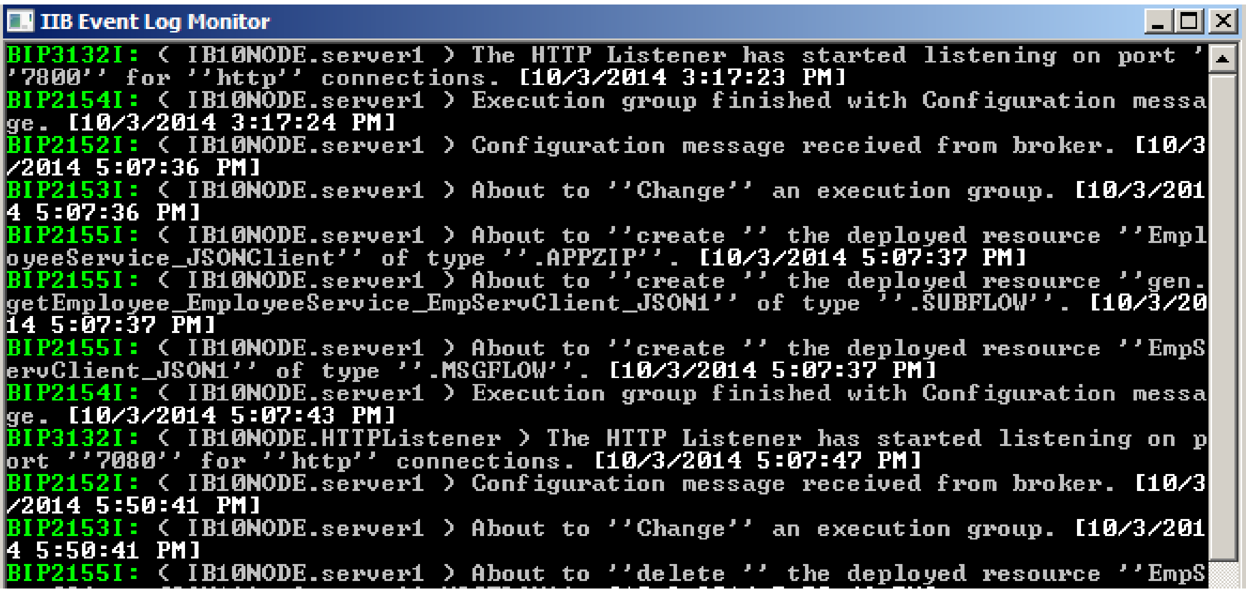
The lab guide will provide a REST API to insert entries into the DEPARTMENT database by implementing a POST operation in the REST API.

An optional Appendix to the guide will also demonstrate implementing a GET operation to retrieve entries from the DEPARTMENT table.

## 1.2 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.



```
BIP3132I: ( IB10NODE.server1 ) The HTTP Listener has started listening on port '
'7800'' for ''http'' connections. [10/3/2014 3:17:23 PM]
BIP2154I: ( IB10NODE.server1 ) Execution group finished with Configuration messa
ge. [10/3/2014 3:17:24 PM]
BIP2152I: ( IB10NODE.server1 ) Configuration message received from broker. [10/3
/2014 5:07:36 PM]
BIP2153I: ( IB10NODE.server1 ) About to ''Change'' an execution group. [10/3/201
4 5:07:36 PM]
BIP2155I: ( IB10NODE.server1 ) About to ''create '' the deployed resource ''Empl
oyeeService_JSONClient'' of type ''.APPZIP''. [10/3/2014 5:07:37 PM]
BIP2155I: ( IB10NODE.server1 ) About to ''create '' the deployed resource ''gen.
getEmployee_EmployeeService_EmpServClient_JSON1'' of type ''.SUBFLOW''. [10/3/20
14 5:07:37 PM]
BIP2155I: ( IB10NODE.server1 ) About to ''create '' the deployed resource ''EmpS
ervClient_JSON1'' of type ''.MSGFLOW''. [10/3/2014 5:07:37 PM]
BIP2154I: ( IB10NODE.server1 ) Execution group finished with Configuration messa
ge. [10/3/2014 5:07:43 PM]
BIP3132I: ( IB10NODE.HTTPListener ) The HTTP Listener has started listening on p
ort ''7080'' for ''http'' connections. [10/3/2014 5:07:47 PM]
BIP2152I: ( IB10NODE.server1 ) Configuration message received from broker. [10/3
/2014 5:50:41 PM]
BIP2153I: ( IB10NODE.server1 ) About to ''Change'' an execution group. [10/3/201
4 5:50:41 PM]
BIP2155I: ( IB10NODE.server1 ) About to ''delete '' the deployed resource ''EmpS
```

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

# 1.3 Configure TESTNODE_iibuser for REST APIs

**Login to Windows as the user "iibuser", password = "passw0rd".** (You may already be logged in).

**Start the IIB Toolkit from the Start menu.**

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

_1. Ensure that TESTNODE_iibuser is started.

_2. 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
```

_3. Restart the IIB node

## 1.4 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.

## 1.5 Import HRDB resources project

This lab guide uses a shared library called HRDB which contains the database definitions for access to the HRDB database. You will now import this project into your workspace.

_1. In the Integration Toolkit, create a new workspace called `RESTAPI_NoSwagger`. (File> Switch workspace> other).

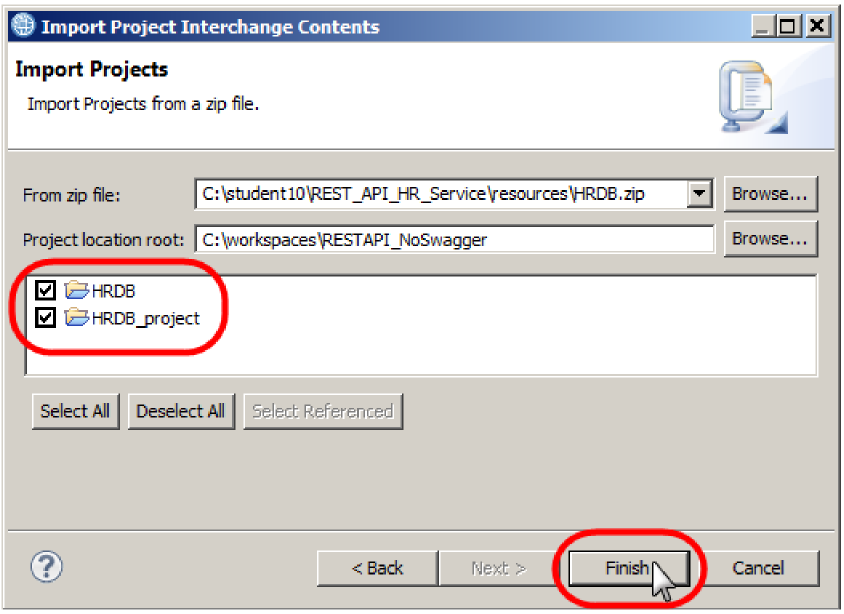
The Eclipse workbench will restart and present the new workspace, ready for you to use.

_2. Right click on the background of the Application Development window and select Import.

_3. Select Project Interchange and click Next.

_4. Navigate to “`C:\student10\REST_API_HR_Service\resources`” and select “`HRDB.zip`”, click the Open button.

_5. Select `HRDB` and `HRDB_Project` and click Finish:



Import Project Interchange Contents

**Import Projects**

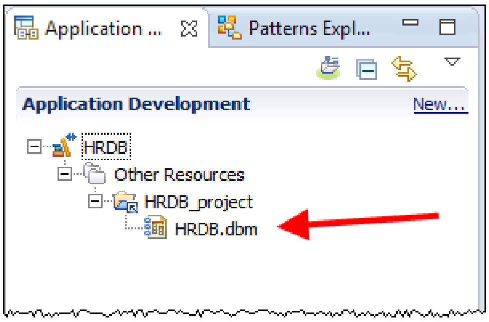
Import Projects from a zip file.

From zip file: C:\student10\REST_API_HR_Service\resources\HRDB.zip

Project location root: C:\workspaces\RESTAPI_NoSwagger

☑ HRDB
☑ HRDB_project

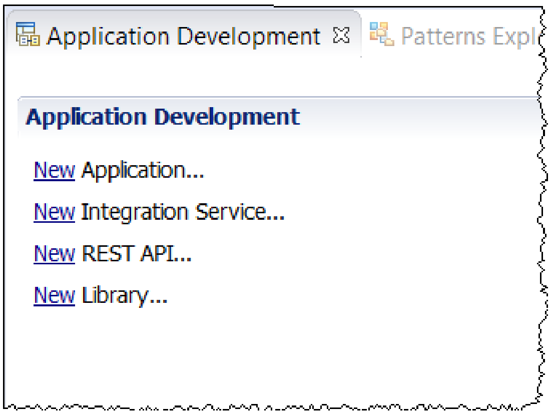
_6. This will import a Shared folder called “HRDB”. The Shared Library contains the HRDB database definition in a project. These files were created in another lab guide in this series of lab guides:



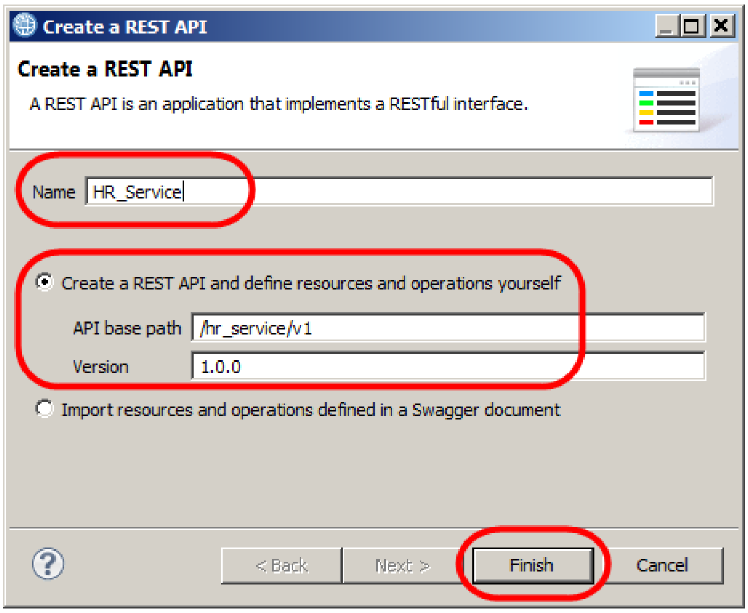
# 2. Create the REST API

In this section you will create a new REST API without using an existing Swagger.json file. The REST API will provide retrieve operations to obtain data from the HRDB DEPARTMENT table. The data will be obtained from the table using the IIB mapping node.

_1. When the Application Development window, click “New REST API ….”



_2. Call the REST API “`HR_Service`” and click Finish:



Create a REST API

A REST API is an application that implements a RESTful interface.

Name: HR_Service

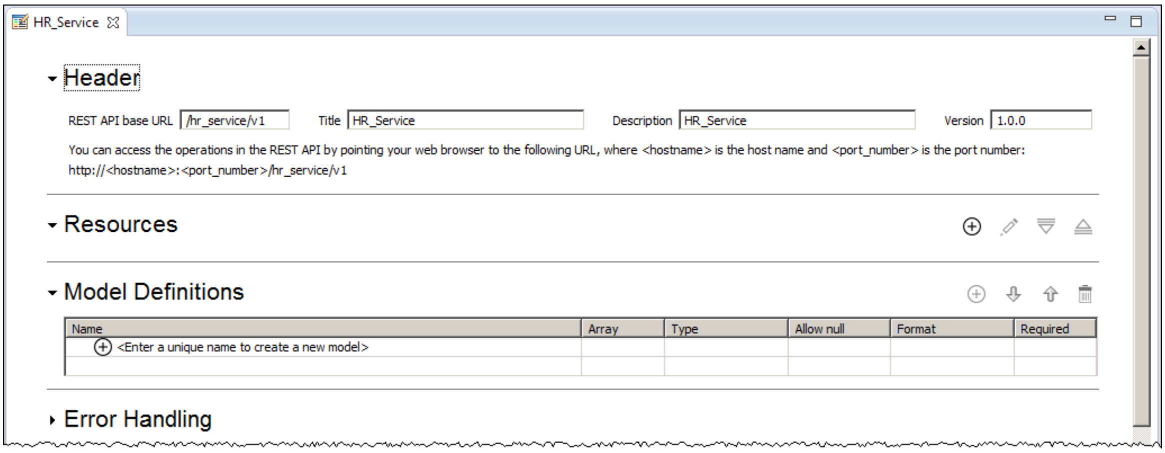
● Create a REST API and define resources and operations yourself

API base path: /hr_service/v1

Version: 1.0.0

○ Import resources and operations defined in a Swagger document

_3. The REST API development configuration will open:



**Header**

REST API base URL /hr_service/v1 Title HR_Service Description HR_Service Version 1.0.0

You can access the operations in the REST API by pointing your web browser to the following URL, where <hostname> is the host name and <port_number> is the port number:
http://<hostname>:<port_number>/hr_service/v1

**Resources**

**Model Definitions**

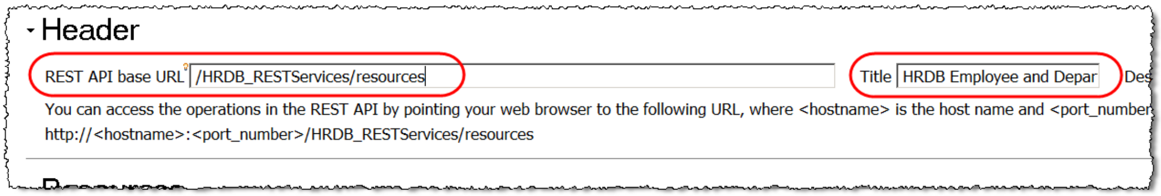
| Name | Array | Type | Allow null | Format | Required |
|---|---|---|---|---|---|
| <Enter a unique name to create a new model> | | | | | |

**Error Handling**

_4. In the Header section, modify:

a) The field “REST API base URL” to read: “`/HRDB_RESTServices/resources`”

b) The Title to read: “`HR Employee and Department Services`”.

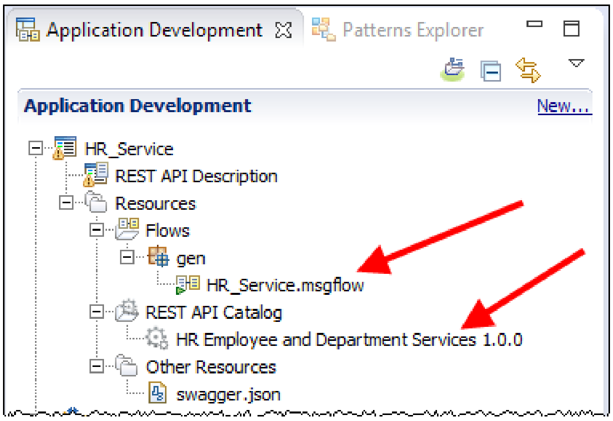


**Header**

REST API base URL /HRDB_RESTServices/resources Title HRDB Employee and Depar Des

You can access the operations in the REST API by pointing your web browser to the following URL, where <hostname> is the host name and <port_number
http://<hostname>:<port_number>/HRDB_RESTServices/resources

_5. Note creating the REST API has automatically created a message flow and a REST API Catalog (*this is a reference to the swagger.json file that is also automatically created by the toolkit*):



## 2.1 Create Model Definitions

We will use the **Model Definitions** section of the REST API editor to model the JSON data that will be used for the Request and Response Schema type in the createDepartment operation that you will create later. You will create objects that will be used to model:

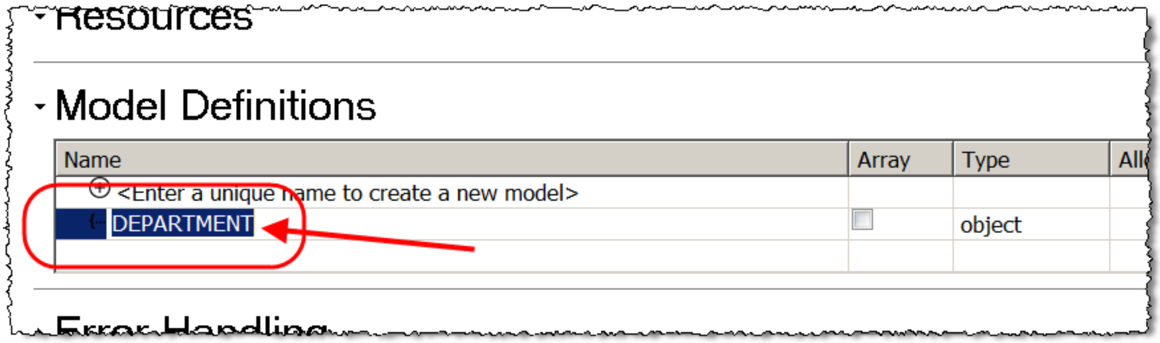
a) Data from the DEPARTMENT table
b) Response data from Database operations
c) A consolidation of the above.

Once these objects have been created you will have the required definitions to enable you to begin creating Resources with Operations in your REST API.

### 2.1.1 DEPARTMENT

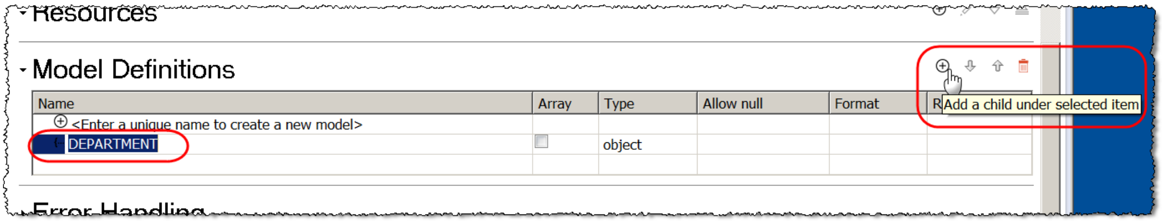
In the REST API Configuration window, configure a new model called “DEPARTMENT”. The model will describe the layout of the DEPARTMENT table.

_1. Click “***Enter a unique name to create a new model***” and replace this text with `DEPARTMENT` and press enter:



| Name | Array | Type | All |
|---|---|---|---|
| <Enter a unique name to create a new model> | | | |
| DEPARTMENT | □ | object | |

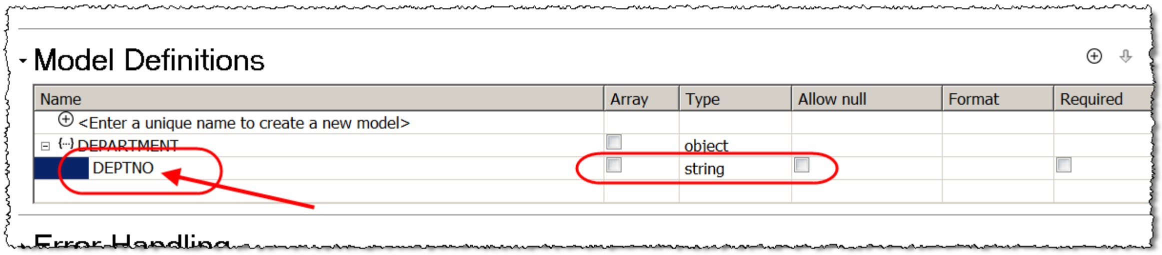
_2. Now click the text “`DEPARTMENT`” to highlight it and click the “Add a child under selected item” button:



| Name | Array | Type | Allow null | Format | R |
|---|---|---|---|---|---|
| <Enter a unique name to create a new model> | | | | | |
| DEPARTMENT | □ | object | | | |

_3. A child element called “element1” will be created.

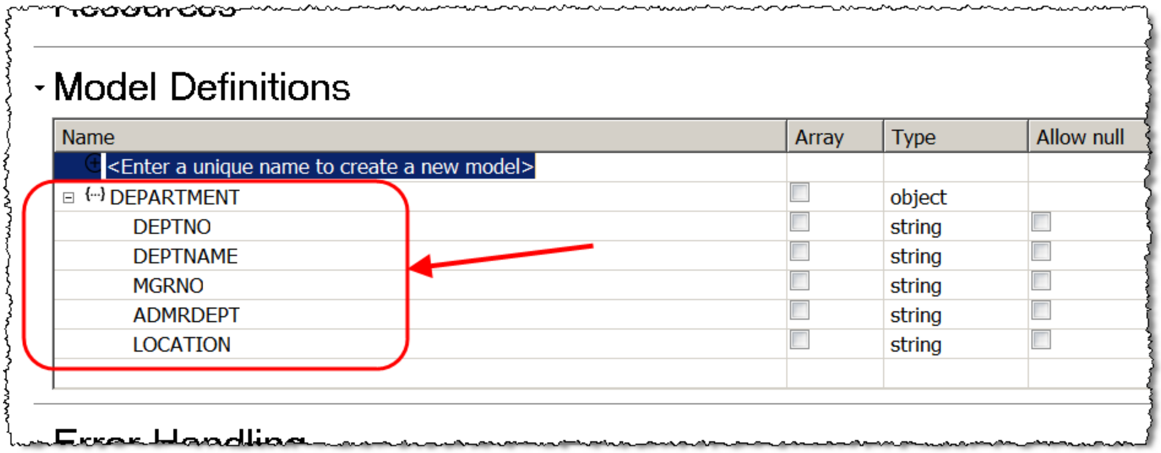
Overtype “element1” to rename rename it to “`DEPTNO`”. Since this field is a character, keep the Type value as “string”:



| Name | Array | Type | Allow null | Format | Required |
|---|---|---|---|---|---|
| <Enter a unique name to create a new model> | | | | | |
| DEPARTMENT | □ | object | | | |
| DEPTNO | □ | string | □ | | □ |

_4. Highlight "`DEPARTMENT`" again and repeat the previous steps to add the following children (in this order): `DEPTNAME, MGRNO, ADMRDEPT; LOCATION` (all with string types).

When complete, the Model Definitions table will look like this:



Model Definitions

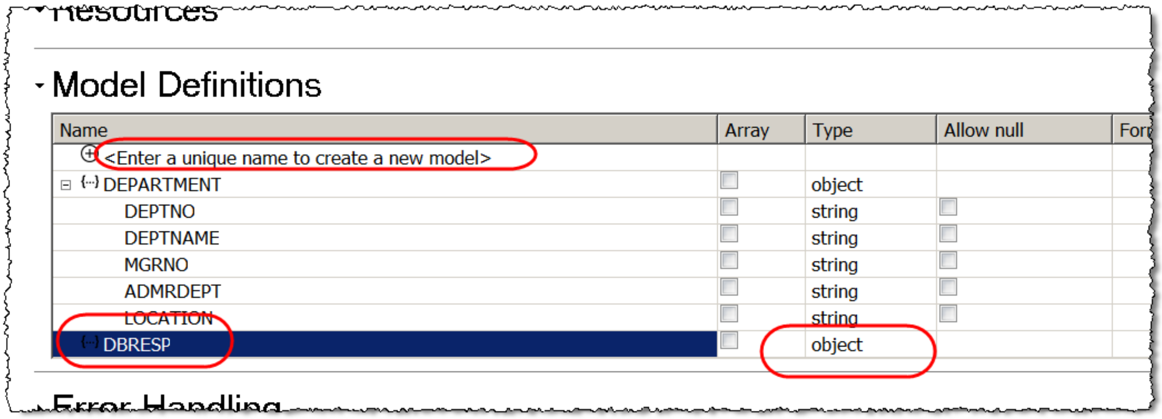
| Name | Array | Type | Allow null |
|---|---|---|---|
| <Enter a unique name to create a new model> | | | |
| DEPARTMENT | ☐ | object | |
| DEPTNO | ☐ | string | ☐ |
| DEPTNAME | ☐ | string | ☐ |
| MGRNO | ☐ | string | ☐ |
| ADMRDEPT | ☐ | string | ☐ |
| LOCATION | ☐ | string | ☐ |

## 2.1.2 DBRESP

The DBRESP model will be used by the REST API to return database status information for example number of rows returned and any database error codes to help with problem determination.

_1. In the REST API Configuration window, configure a new model called "DBRESP".

Click "***Enter a unique name to create a new model***", replace this text with `DBRESP`, and press enter:



Model Definitions

| Name | Array | Type | Allow null | For |
|---|---|---|---|---|
| <Enter a unique name to create a new model> | | | | |
| DEPARTMENT | ☐ | object | | |
| DEPTNO | ☐ | string | ☐ | |
| DEPTNAME | ☐ | string | ☐ | |
| MGRNO | ☐ | string | ☐ | |
| ADMRDEPT | ☐ | string | ☐ | |
| LOCATION | ☐ | string | ☐ | |
| DBRESP | ☐ | object | | |

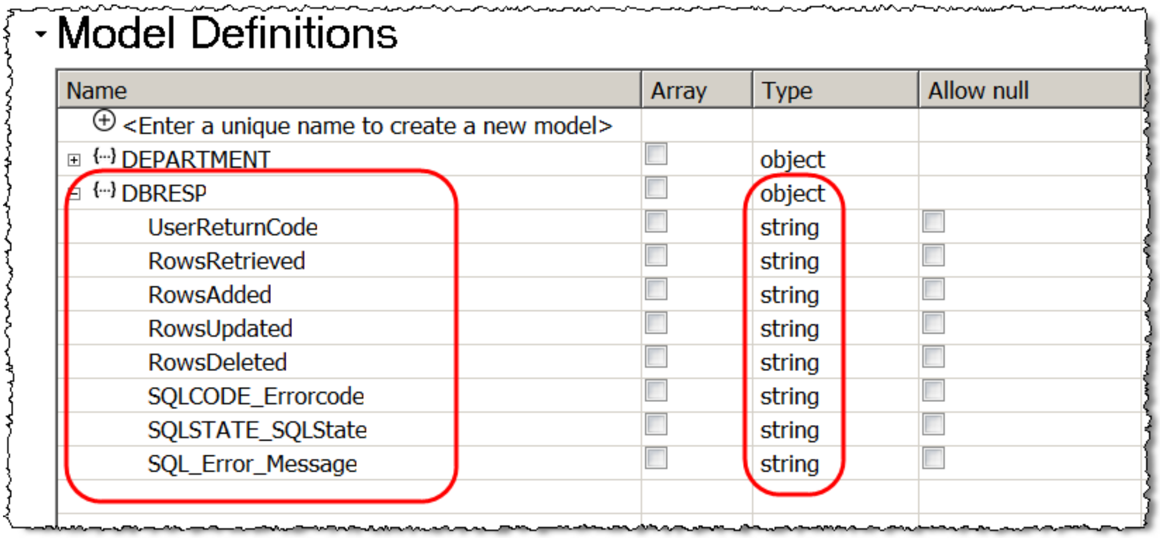
Note: DBRESP should be created as an Object (type column) at the same level as DEPARTMENT.

_2. Use the “**Add a Child under selected element**” button as you did earlier to add the following children to DBRESP:

a) UserReturnCode
b) RowsRetrieved
c) RowsAdded
d) RowsUpdated
e) RowsDeleted
f) SQLCODE_Errorcode
g) SQLSTATE_SQLState
h) SQL_Error_Message

You will add the elements with a (default) Type definition of String.

The Model definition table will look like this when you have completed this step:



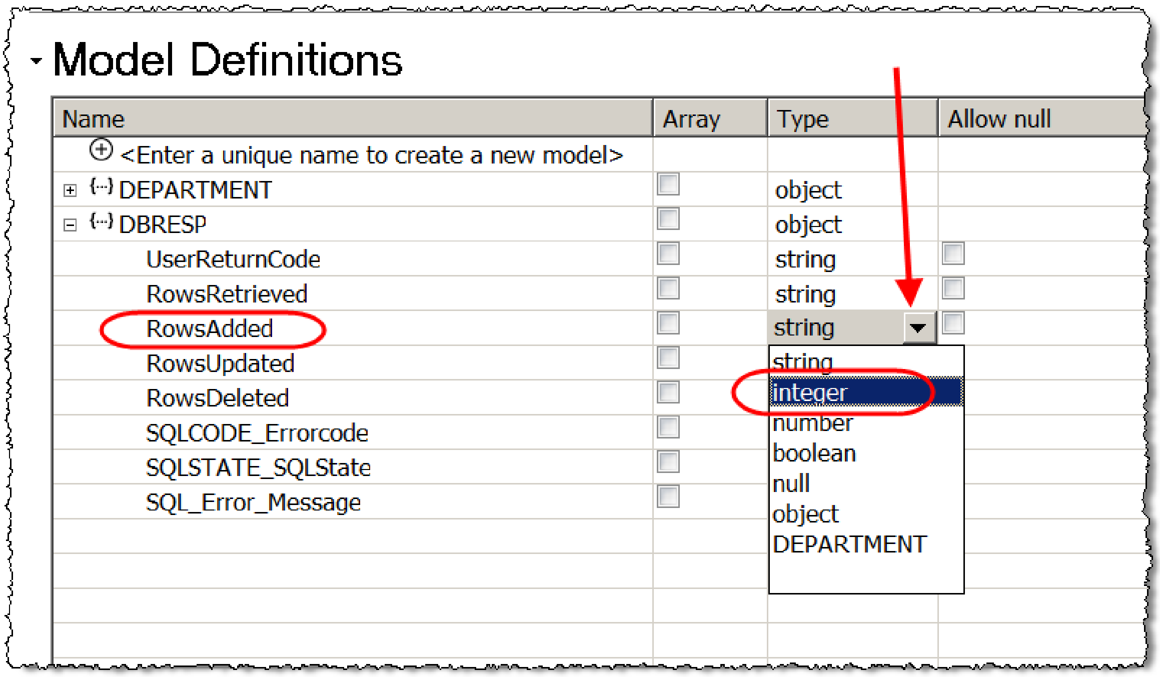
## Model Definitions

| Name | Array | Type | Allow null |
|---|---|---|---|
| ⊕ <Enter a unique name to create a new model> | | | |
| ⊞ {...} DEPARTMENT | □ | object | |
| ⊟ {...} DBRESP | □ | object | |
| UserReturnCode | □ | string | □ |
| RowsRetrieved | □ | string | □ |
| RowsAdded | □ | string | □ |
| RowsUpdated | □ | string | □ |
| RowsDeleted | □ | string | □ |
| SQLCODE_Errorcode | □ | string | □ |
| SQLSTATE_SQLState | □ | string | □ |
| SQL_Error_Message | □ | string | □ |

_3. Change the Type to **Integer** for the following elements:

a) UserReturnCode
b) RowsRetrieved
c) RowsAdded
d) RowsUpdated
e) RowsDeleted

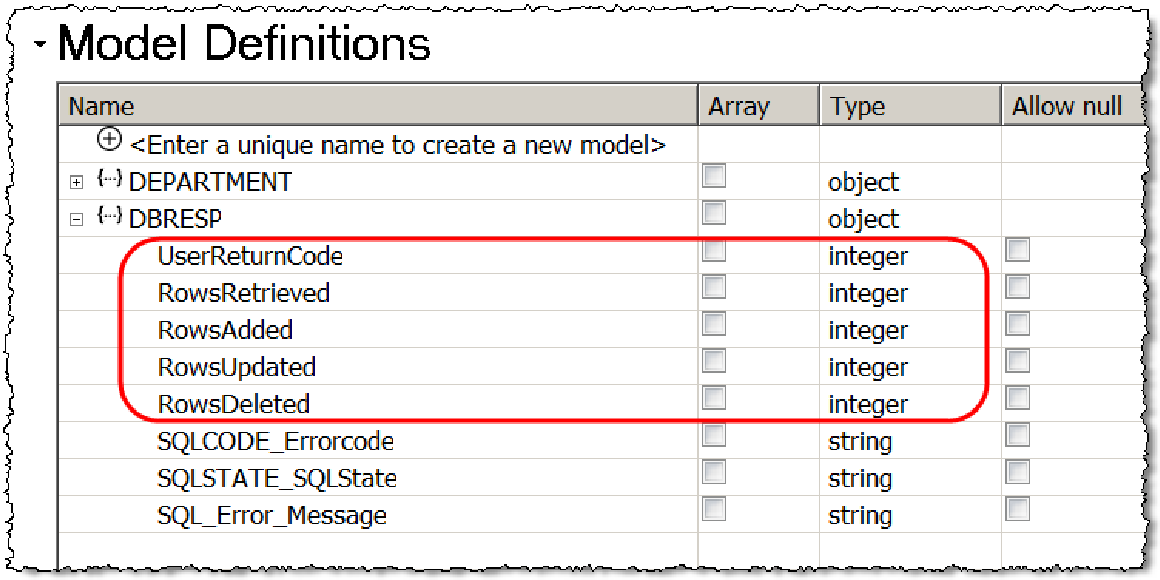
(*click on the default string definition in the "Type" column to change the type*), for example: the following shows how to change RowsAdded to integer:



Model Definitions

| Name | Array | Type | Allow null |
|---|---|---|---|
| ⊕ <Enter a unique name to create a new model> | | | |
| ⊞ {…} DEPARTMENT | □ | object | |
| ⊟ {…} DBRESP | □ | object | |
| UserReturnCode | □ | string | □ |
| RowsRetrieved | □ | string | □ |
| RowsAdded | □ | string ▼ | □ |
| RowsUpdated | □ | | |
| RowsDeleted | □ | | |
| SQLCODE_Errorcode | □ | | |
| SQLSTATE_SQLState | □ | | |
| SQL_Error_Message | □ | | |

_4. The Model definition table will look like this when you have completed this step:



Model Definitions

| Name | Array | Type | Allow null |
|---|---|---|---|
| ⊕ <Enter a unique name to create a new model> | | | |
| ⊞ {…} DEPARTMENT | □ | object | |
| ⊟ {…} DBRESP | □ | object | |
| UserReturnCode | □ | integer | □ |
| RowsRetrieved | □ | integer | □ |
| RowsAdded | □ | integer | □ |
| RowsUpdated | □ | integer | □ |
| RowsDeleted | □ | integer | □ |
| SQLCODE_Errorcode | □ | string | □ |
| SQLSTATE_SQLState | □ | string | □ |
| SQL_Error_Message | □ | string | □ |

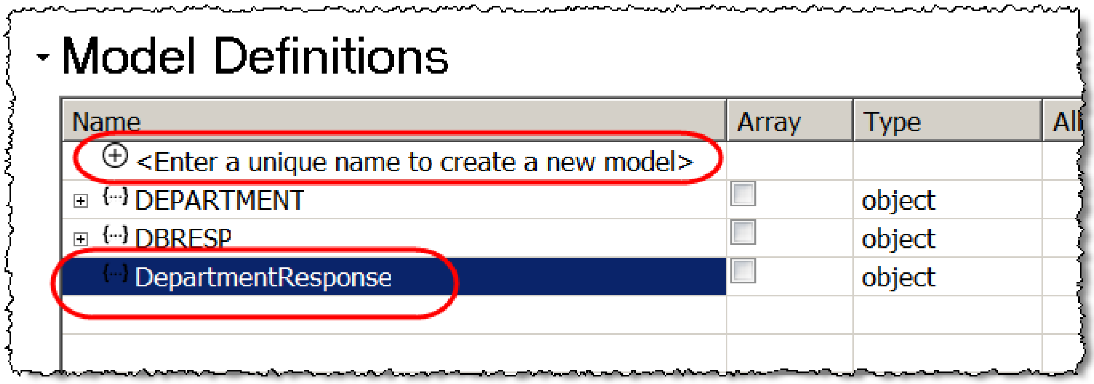
### 2.1.3 DepartmentResponse

The **DepartmentResponse** model will be used by the REST API to return both Department information (from the DEPARTMENT Table) and database status information together as a single object. The object will be defined using the DEPARTMENT and DBRESP objects that you created earlier.

_1. Minimise the DPRESP object by clicking on the "-" sign next to its name.

_2. Create a new object called "**DepartmentResponse**".

Click "***Enter a unique name to create a new model***", replace this text with `DepartmentResponse`, and press enter:



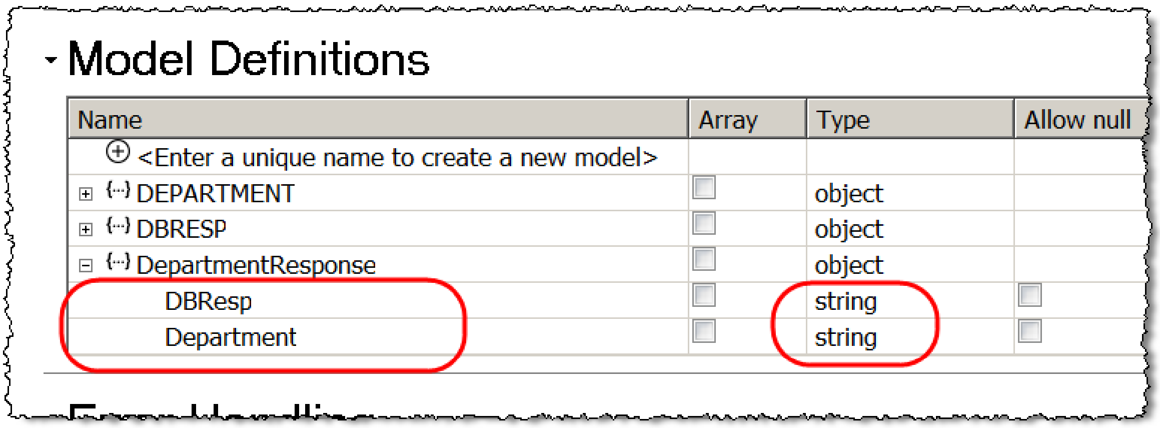
Model Definitions

| Name | Array | Type | Al |
|---|---|---|---|
| ⊕ <Enter a unique name to create a new model> | | | |
| ⊞ {...} DEPARTMENT | □ | object | |
| ⊞ {...} DBRESP | □ | object | |
| {...} DepartmentResponse | □ | object | |

Note: DepartmentResponse should be created as an Object (type column) at the same levels as DEPARTMENT and DPRESP.

_3. Using the "**Add a child under selected item**" button, add the following elements:

a) DBResp
b) Department



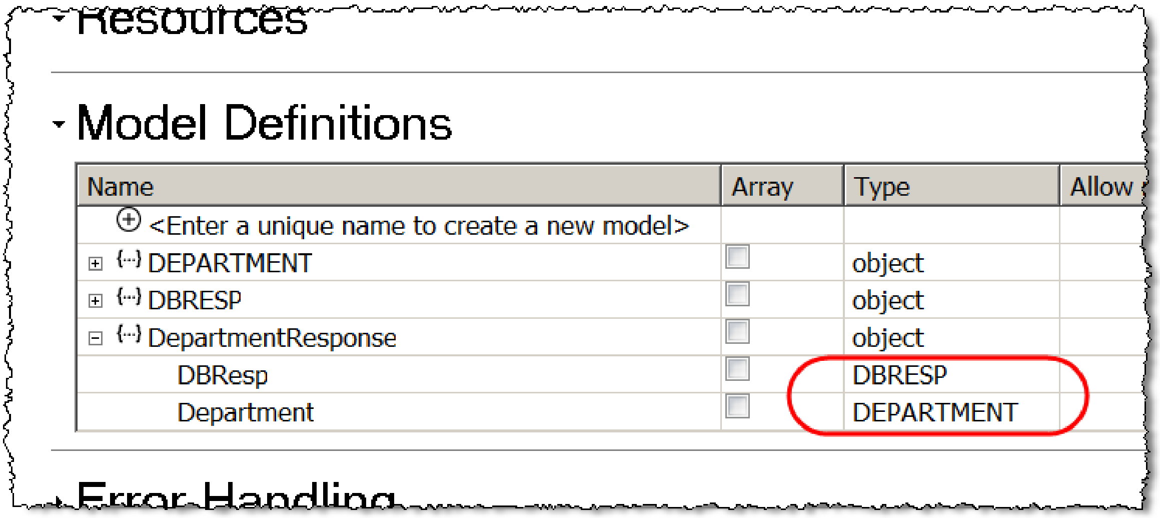
Model Definitions

| Name | Array | Type | Allow null |
|---|---|---|---|
| ⊕ <Enter a unique name to create a new model> | | | |
| ⊞ {...} DEPARTMENT | □ | object | |
| ⊞ {...} DBRESP | □ | object | |
| ⊟ {...} DepartmentResponse | □ | object | |
| DBResp | □ | string | □ |
| Department | □ | string | □ |

__4. Click on the string definition for each element and change the type to the followng:

a) DBResp – Type= **DBRESP**
b) Department – Type= **DEPARTMENT**

The table will look like this when you have completed this step:



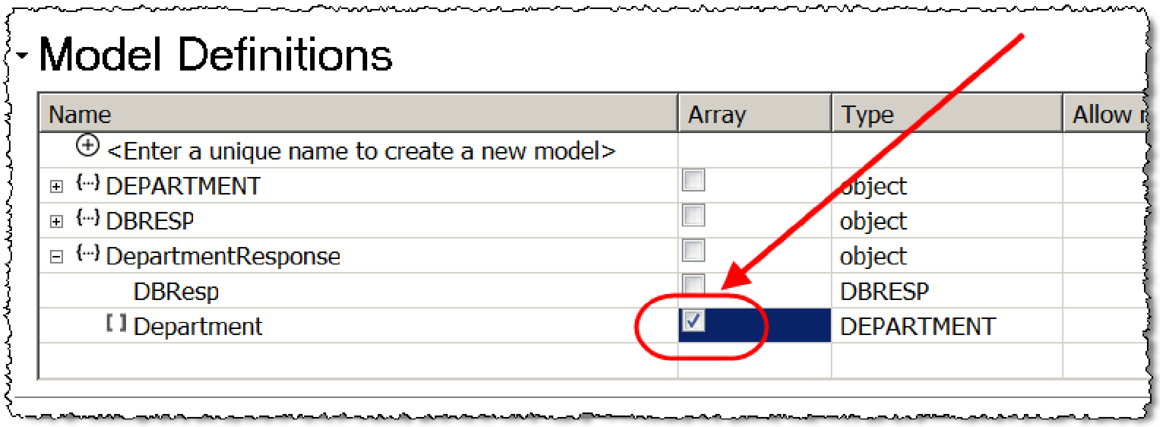
Resources

Model Definitions

| Name | Array | Type | Allow |
|---|---|---|---|
| ⊕ <Enter a unique name to create a new model> | | | |
| ⊞ {…} DEPARTMENT | □ | object | |
| ⊞ {…} DBRESP | □ | object | |
| ⊟ {…} DepartmentResponse | □ | object | |
| DBResp | □ | DBRESP | |
| Department | □ | DEPARTMENT | |

Error Handling

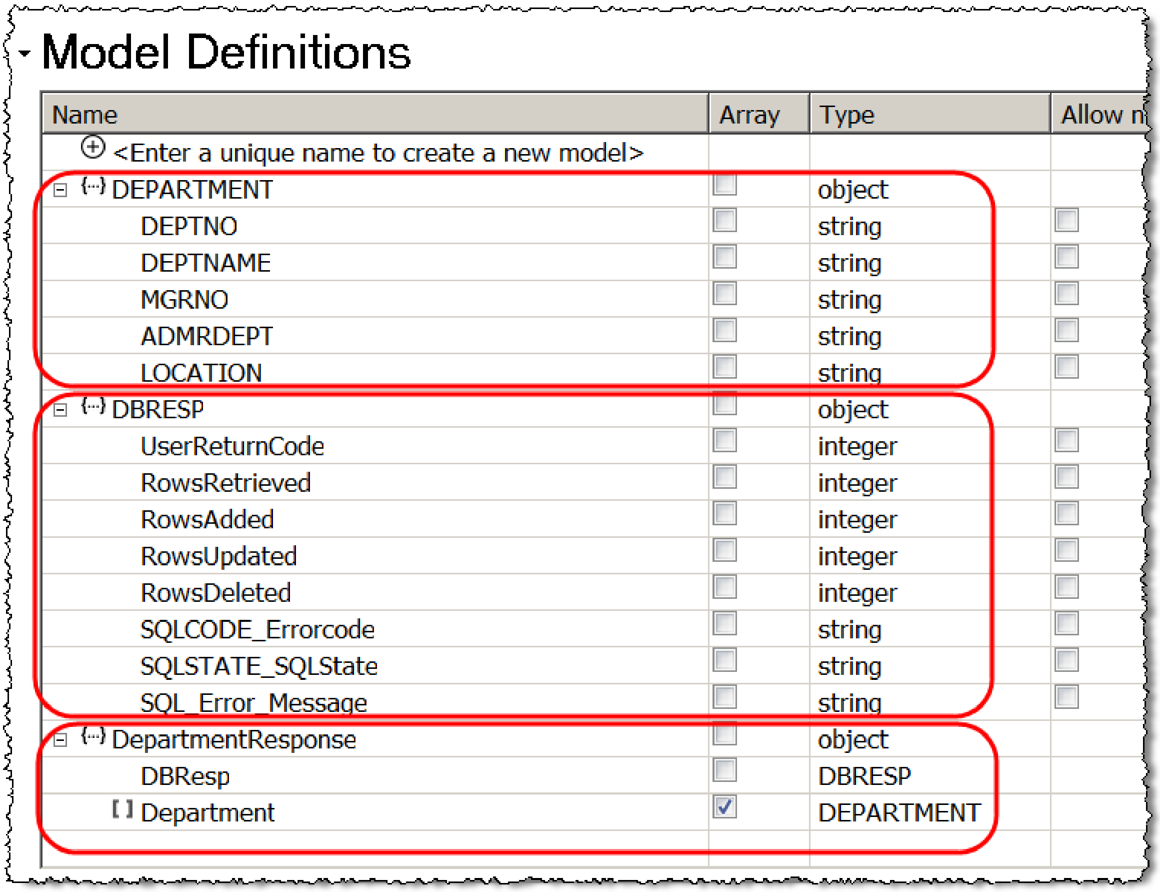
__5. The implementation logic that you will (optionally) configure in the appendix of this guide enables multiple occurrences of the DEPARTMENT element to be returned for example when a partial key or no key is specified in the get request. For this reason we need the DEPARTMENT element in DepartmentResponse to be defined as a JSON Array. Click the array tick box for DEPARTMENT in DepartmentResponse:



Model Definitions

| Name | Array | Type | Allow r |
|---|---|---|---|
| ⊕ <Enter a unique name to create a new model> | | | |
| ⊞ {…} DEPARTMENT | □ | object | |
| ⊞ {…} DBRESP | □ | object | |
| ⊟ {…} DepartmentResponse | □ | object | |
| DBResp | □ | DBRESP | |
| [ ] Department | ☑ | DEPARTMENT | |

_6. Expand the DEPARTMENT, DBRESP and DepartmentResponse objects and verify that your Model Definitions look like this:



## Model Definitions

| Name | Array | Type | Allow n |
|---|---|---|---|
| ⊕ <Enter a unique name to create a new model> | | | |
| ⊟ {…} DEPARTMENT | □ | object | |
| DEPTNO | □ | string | □ |
| DEPTNAME | □ | string | □ |
| MGRNO | □ | string | □ |
| ADMRDEPT | □ | string | □ |
| LOCATION | □ | string | □ |
| ⊟ {…} DBRESP | □ | object | |
| UserReturnCode | □ | integer | □ |
| RowsRetrieved | □ | integer | □ |
| RowsAdded | □ | integer | □ |
| RowsUpdated | □ | integer | □ |
| RowsDeleted | □ | integer | □ |
| SQLCODE_Errorcode | □ | string | □ |
| SQLSTATE_SQLState | □ | string | □ |
| SQL_Error_Message | □ | string | □ |
| ⊟ {…} DepartmentResponse | □ | object | |
| DBResp | □ | DBRESP | |
| [ ] Department | ☑ | DEPARTMENT | |

_7. Save the REST API definition (Ctrl S).

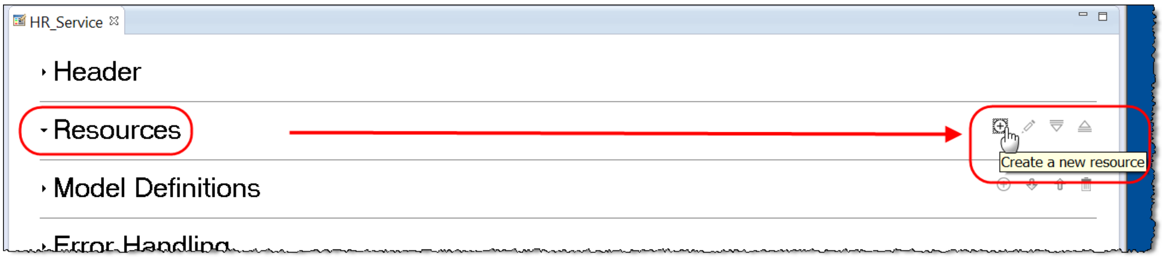
## 2.2 Create Resources

This REST API will obtain details from the DEPARTMENT table. The resource we are dealing with are departments defined in the DEPARTMENT table. In this section you will create a resource called "/departments".

The "/departments" resource will have two operations:

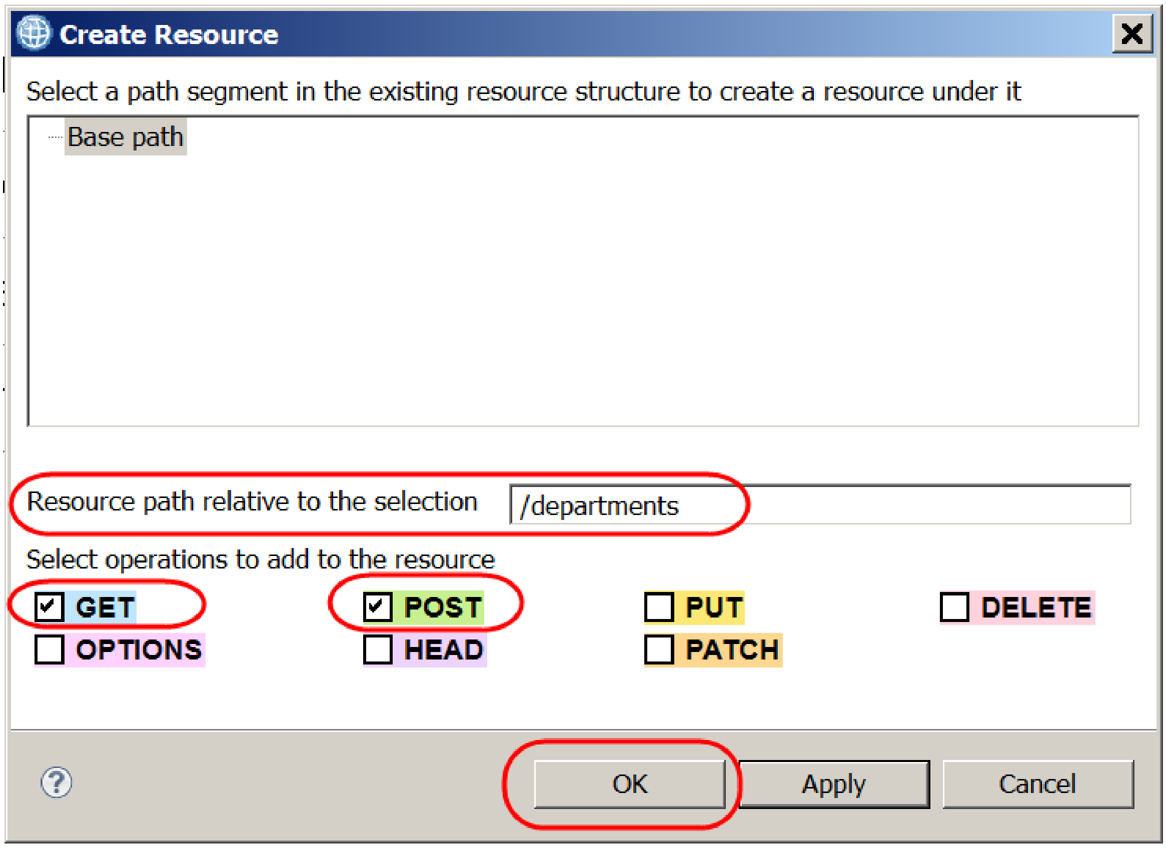
1. A "**POST**" operation called "`createDepartment`" which will be used to add entries to the DEPARTMENT table. The request body of the operation will be used as input.

2. A "**GET**" operation called "`getDepartments`". As input this operation will have an (input) query parameter called "`departmentKey`". If the value of this parameter is blank, ALL records in the DEPARTMENT table will be returned. If the value of the input parameter is not blank this value will be used to perform a keyed read on the DEPARTMENT table.

| | |
|---|---|
| _1. | In the Resources section, click Resources then Click "Create a new resource":<br><br> HR_Service<br>Header<br>Resources<br>Model Definitions<br>Error Handling<br>Create a new resource<br>The Create Resource window will open. |

_2. In the Create Resource window, type `/departments` in the “Resource path relative to the selection” field.

Select “GET” and “POST” and click OK:



**Create Resource**

Select a path segment in the existing resource structure to create a resource under it

Base path

Resource path relative to the selection: /departments

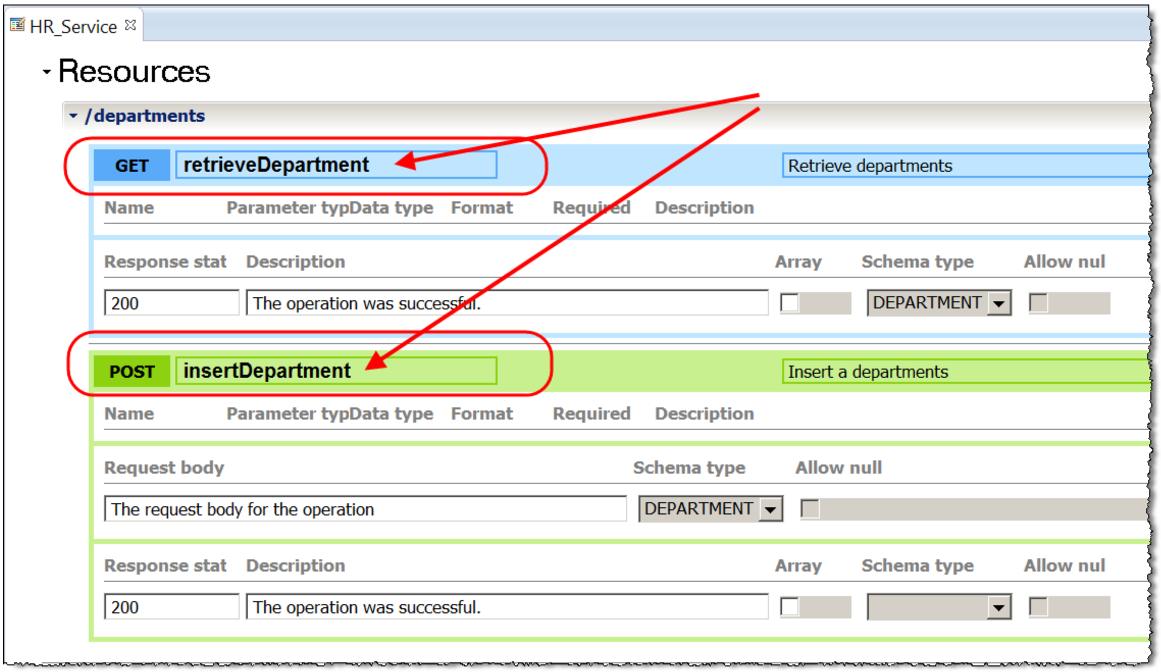
Select operations to add to the resource

☑ GET ☑ POST □ PUT □ DELETE

□ OPTIONS □ HEAD □ PATCH

_3. This will create a GET operation (in blue) and a POST operation (in Green).

Rename these operations `retrieveDepartment` and `insertDepartment` respectively:
(*If you receive a message to confirm saving the REST API, click OK to dismiss it*)



HR_Service

Resources

/departments

GET retrieveDepartment — Retrieve departments

| Name | Parameter typ | Data type | Format | Required | Description |
|---|---|---|---|---|---|

| Response stat | Description | Array | Schema type | Allow nul |
|---|---|---|---|---|
| 200 | The operation was successful. | □ | DEPARTMENT | □ |

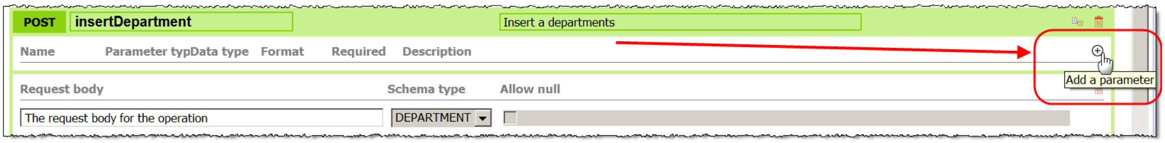
POST insertDepartment — Insert a departments

| Name | Parameter typ | Data type | Format | Required | Description |
|---|---|---|---|---|---|

| Request body | Schema type | Allow null |
|---|---|---|
| The request body for the operation | DEPARTMENT | □ |

| Response stat | Description | Array | Schema type | Allow nul |
|---|---|---|---|---|
| 200 | The operation was successful. | □ |  | □ |

_4. In the insertDepartment operation click the “Add a Parameter” icon:

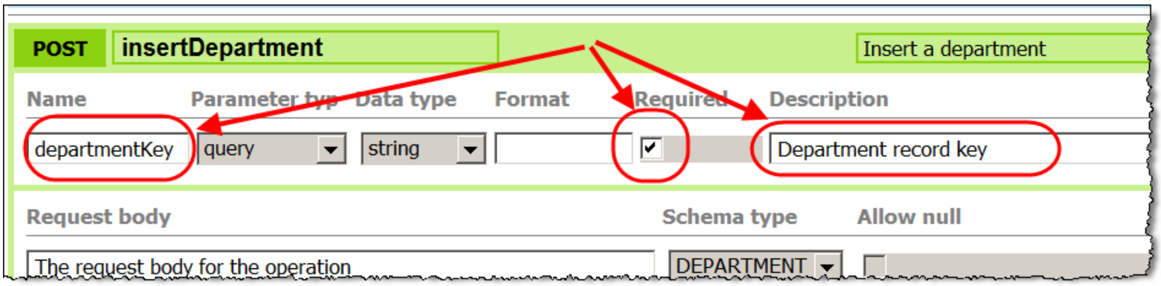


POST insertDepartment — Insert a departments

| Name | Parameter typ | Data type | Format | Required | Description |
|---|---|---|---|---|---|

| Request body | Schema type | Allow null |
|---|---|---|
| The request body for the operation | DEPARTMENT | □ |

_5. Call the new parameter `departmentKey`, give it a description and select “Required”:

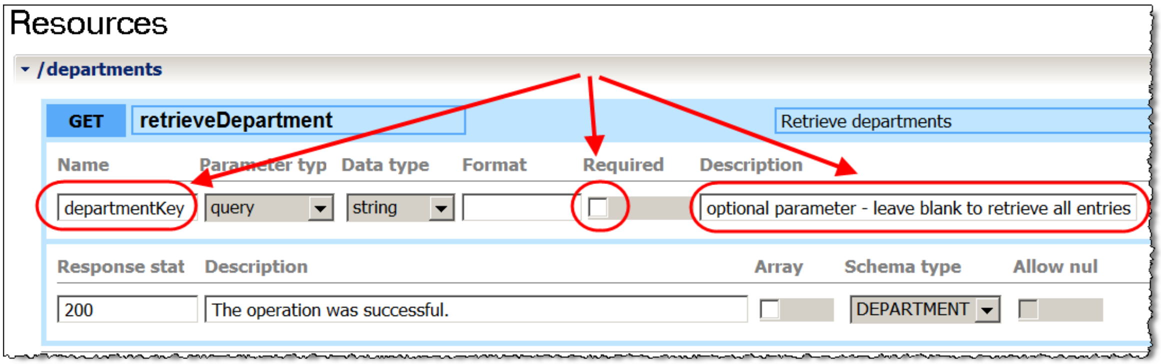


POST insertDepartment — Insert a department

| Name | Parameter typ | Data type | Format | Required | Description |
|---|---|---|---|---|---|
| departmentKey | query | string | | ☑ | Department record key |

| Request body | Schema type | Allow null |
|---|---|---|
| The request body for the operation | DEPARTMENT | □ |

_6. Repeat the above process to add (the same named) *optional* parameter to the retrieveDept (GET) operation:



Resources

/departments

GET retrieveDepartment — Retrieve departments

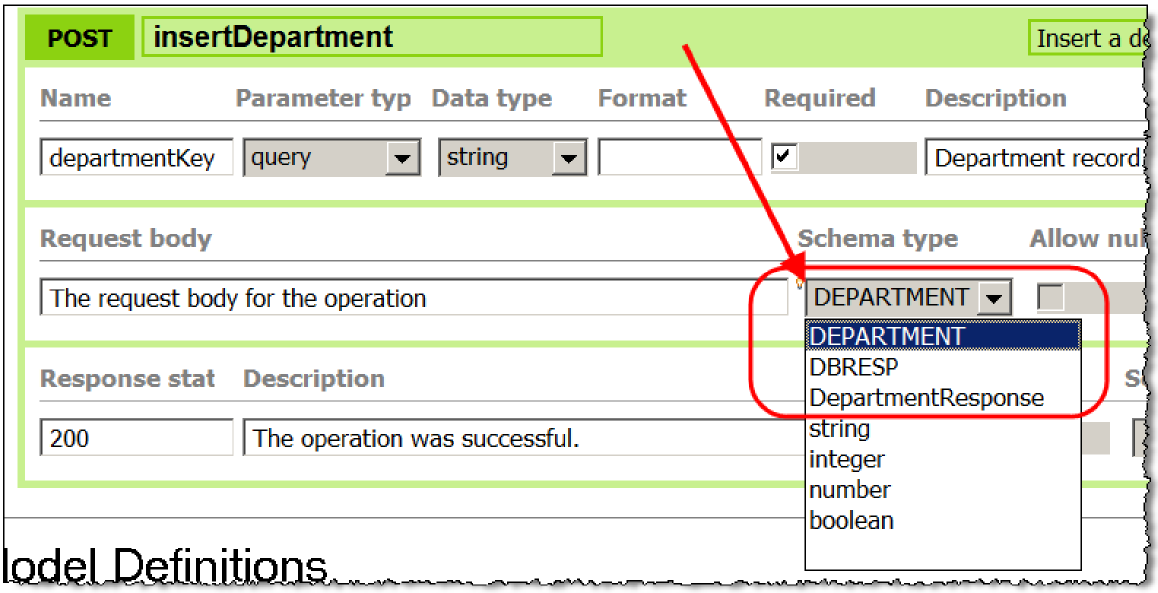
| Name | Parameter typ | Data type | Format | Required | Description |
|---|---|---|---|---|---|
| departmentKey | query | string | | □ | optional parameter - leave blank to retrieve all entries |

| Response stat | Description | Array | Schema type | Allow nul |
|---|---|---|---|---|
| 200 | The operation was successful. | □ | DEPARTMENT | □ |

## 2.3 Implement the insertDepartment (POST) operation

With the /departments resource defined with two operations, implementation of the operation is performed by creating an IBM Integration Bus subflow. In this next section you will implement the logic that will be performed when the insertDepartment operation is called.

_1. In the insertDepartment (POST) operation note that the **Schema type** definition in the **Request body** section has defaulted to the DEPARTMENT schema that you defined in the previous section, the other schemas can also be selected using the drop down (leave the DEPARTMENT schema selected):



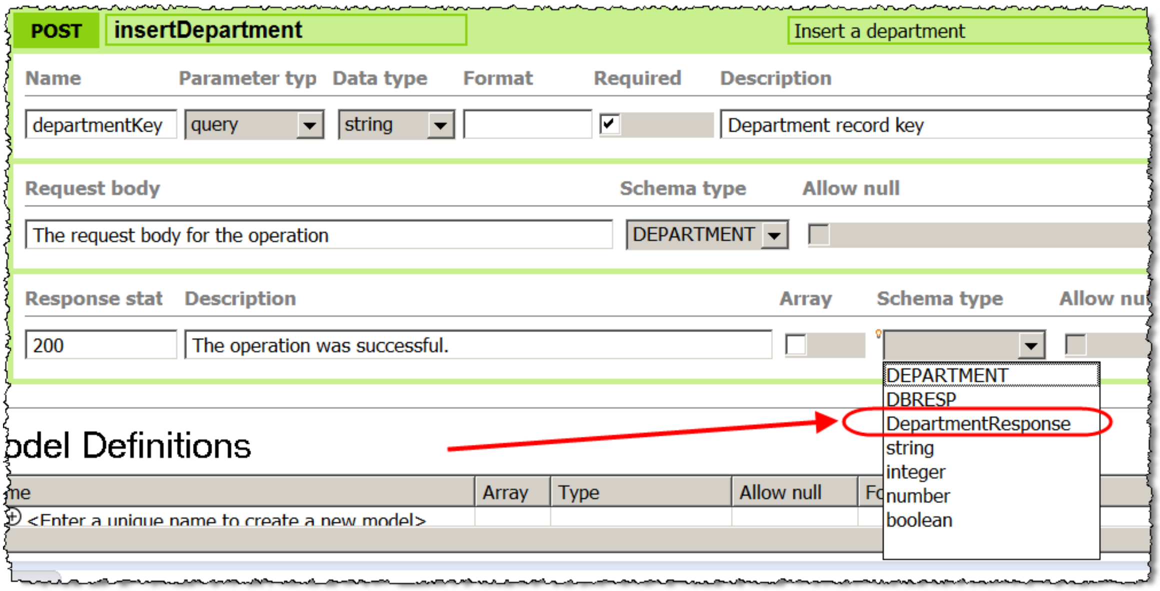
POST insertDepartment Insert a de

| Name | Parameter typ | Data type | Format | Required | Description |
|---|---|---|---|---|---|
| departmentKey | query | string | | ☑ | Department record |

| Request body | Schema type | Allow nul |
|---|---|---|
| The request body for the operation | DEPARTMENT | ☐ |

| Response stat | Description |
|---|---|
| 200 | The operation was successful. |

_2. In the Response status section for the POST command change the Schema type to DepartmentResponse; this will ensure that the Output of the map that you create in the next section to insert the entry into the DEPARTMENT table has the correct output Message Assembly:



POST insertDepartment Insert a department

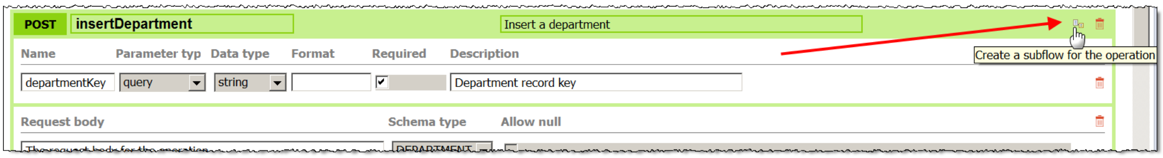
| Name | Parameter typ | Data type | Format | Required | Description |
|---|---|---|---|---|---|
| departmentKey | query | string | | ☑ | Department record key |

| Request body | Schema type | Allow null |
|---|---|---|
| The request body for the operation | DEPARTMENT | ☐ |

| Response stat | Description | Array | Schema type | Allow nu |
|---|---|---|---|---|
| 200 | The operation was successful. | ☐ | | ☐ |

| me | Array | Type | Allow null | Fo |
|---|---|---|---|---|
| <Enter a unique name to create a new model> | | | | |

_3. In the insertDepartment (POST) operation, click “Create a subflow for the operation”:



POST insertDepartment

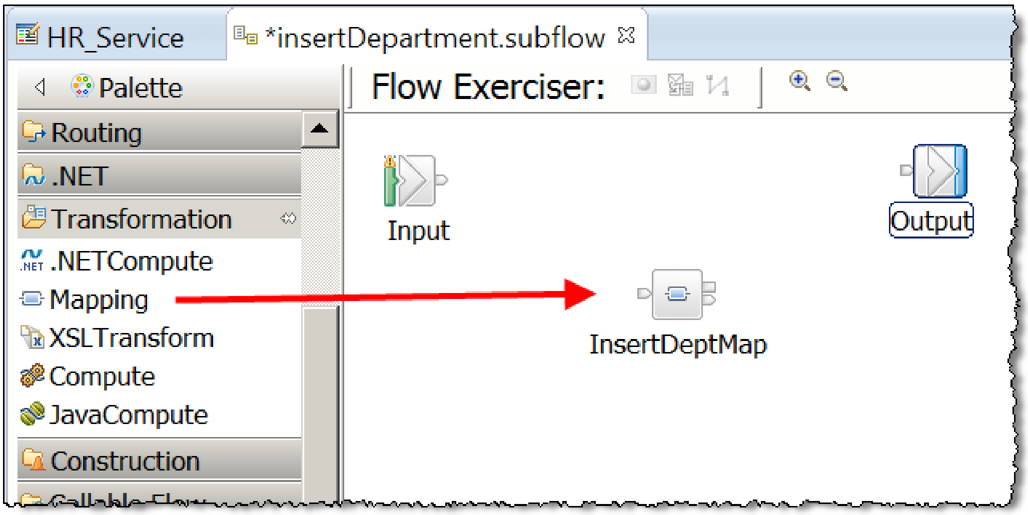
Insert a department

| Name | Parameter typ | Data type | Format | Required | Description |
|---|---|---|---|---|---|
| departmentKey | query | string | | ☑ | Department record key |

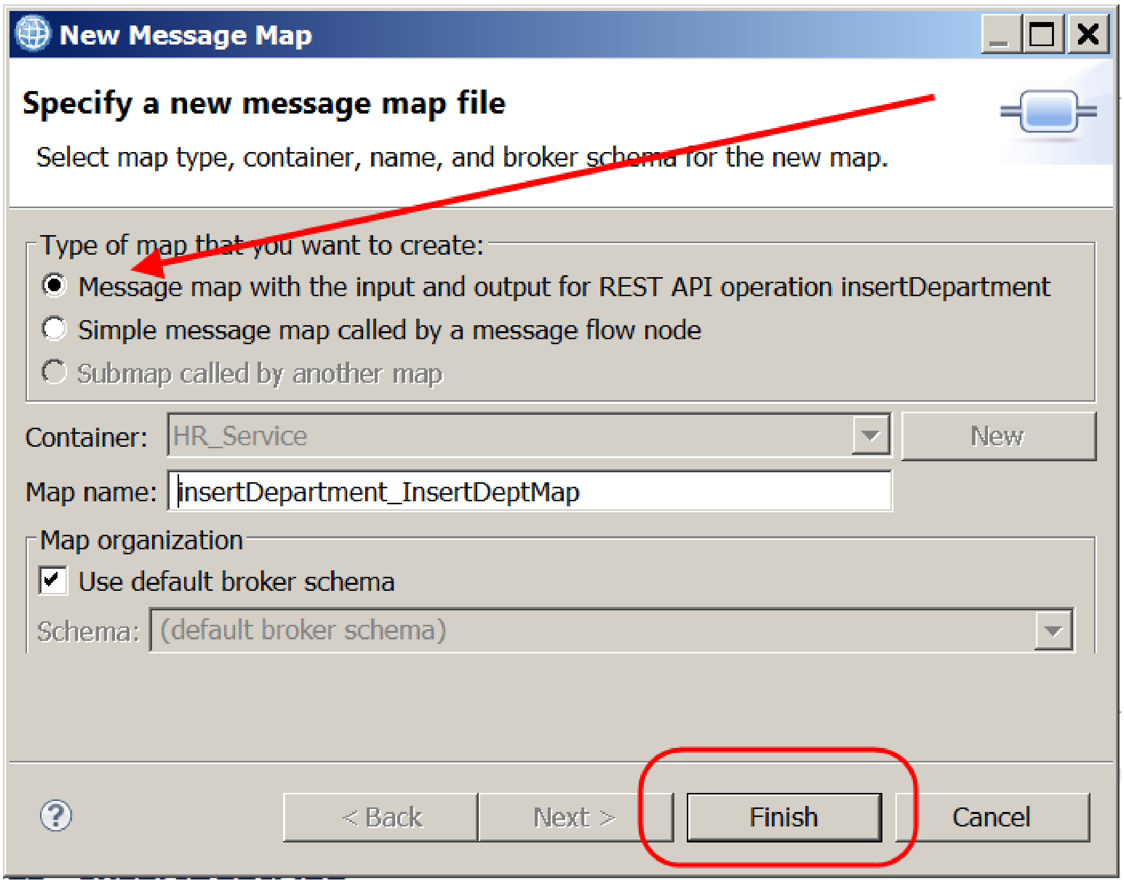
| Request body | Schema type | Allow null |
|---|---|---|

If you receive as message asking if its OK to save the REST API, click OK to dismiss the message.

_4. In the subflow editor, drop a Mapping node onto the canvas and call it “InsertDeptMap”:



_5. Double click on the mapping node and note that the map editor is aware that it has been called from within a REST API – the option “Message map with input and output for REST API operation inserMap” is selected by default. Click Finish:



**New Message Map**

**Specify a new message map file**

Select map type, container, name, and broker schema for the new map.

Type of map that you want to create:

● Message map with the input and output for REST API operation insertDepartment

○ Simple message map called by a message flow node

○ Submap called by another map

Container: HR_Service

Map name: insertDepartment_InsertDeptMap

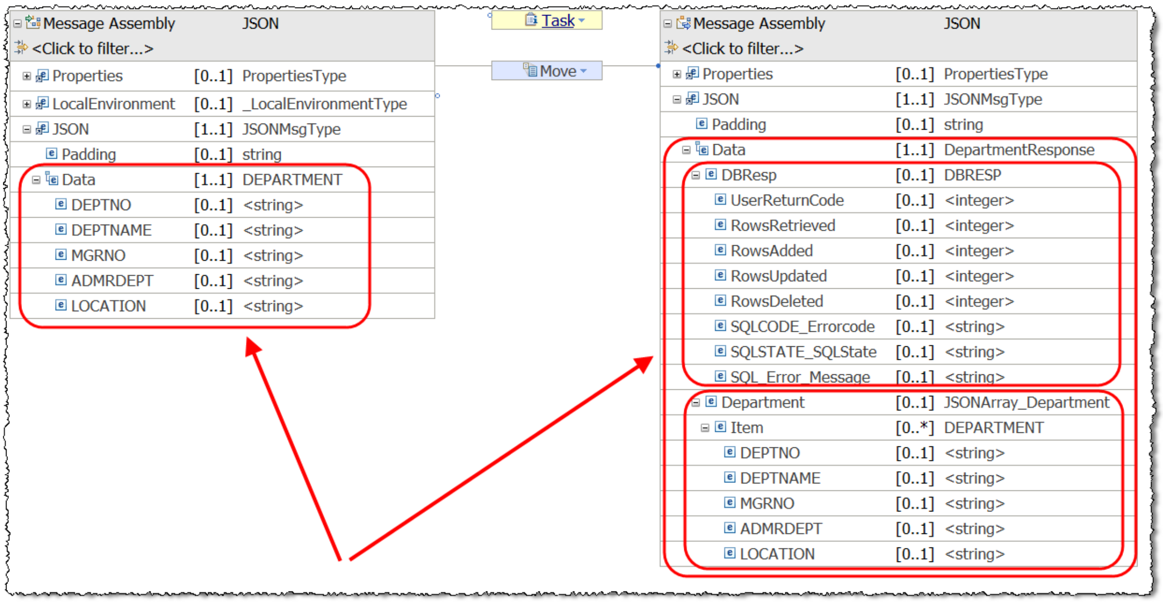
Map organization

☑ Use default broker schema

Schema: (default broker schema)

The mapping editor will open.

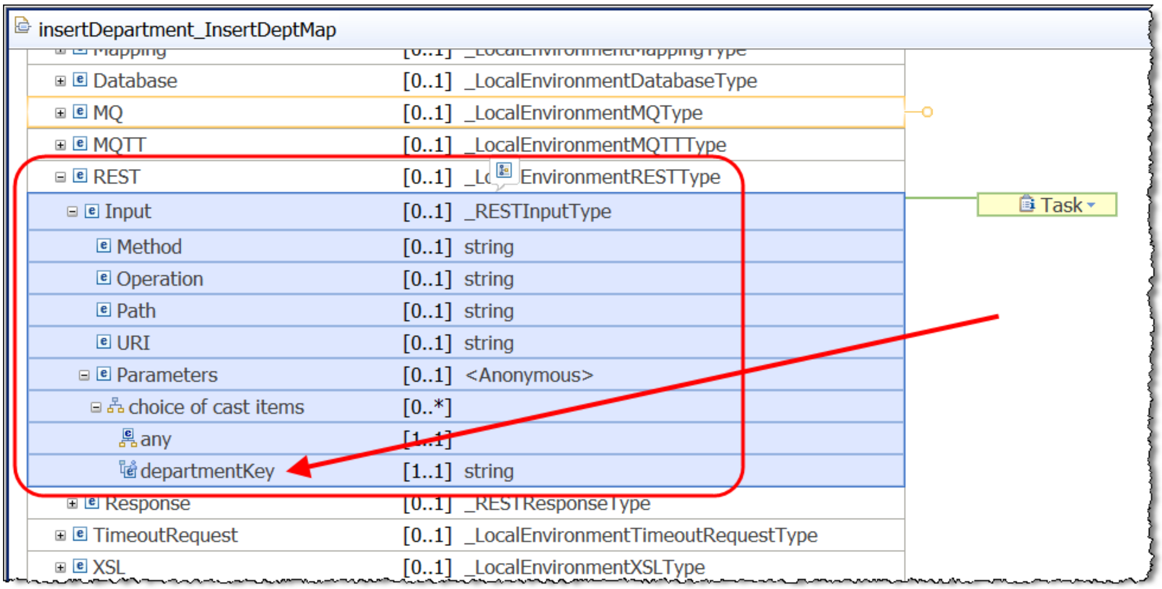
_6. In the mapping editor, on the input and output Message Assemblies, expand JSON > Data note the elements defined in DEPARTMENT (for the input) and DepartmentResponse have been automatically defined in the map:



| Message Assembly | | JSON |
|---|---|---|
| <Click to filter...> | | |
| Properties | [0..1] | PropertiesType |
| LocalEnvironment | [0..1] | _LocalEnvironmentType |
| JSON | [1..1] | JSONMsgType |
| Padding | [0..1] | string |
| Data | [1..1] | DEPARTMENT |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

| Message Assembly | | JSON |
|---|---|---|
| <Click to filter...> | | |
| Properties | [0..1] | PropertiesType |
| JSON | [1..1] | JSONMsgType |
| Padding | [0..1] | string |
| Data | [1..1] | DepartmentResponse |
| DBResp | [0..1] | DBRESP |
| UserReturnCode | [0..1] | <integer> |
| RowsRetrieved | [0..1] | <integer> |
| RowsAdded | [0..1] | <integer> |
| RowsUpdated | [0..1] | <integer> |
| RowsDeleted | [0..1] | <integer> |
| SQLCODE_Errorcode | [0..1] | <string> |
| SQLSTATE_SQLState | [0..1] | <string> |
| SQL_Error_Message | [0..1] | <string> |
| Department | [0..1] | JSONArray_Department |
| Item | [0..*] | DEPARTMENT |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

_7. Expand `LocalEnvironment> REST> Input> Parameters> Choice of cast items` and note the `departmentKey` parameter that you specified for the insertDepartment operation has been automatically defined in the map:



insertDepartment_InsertDeptMap

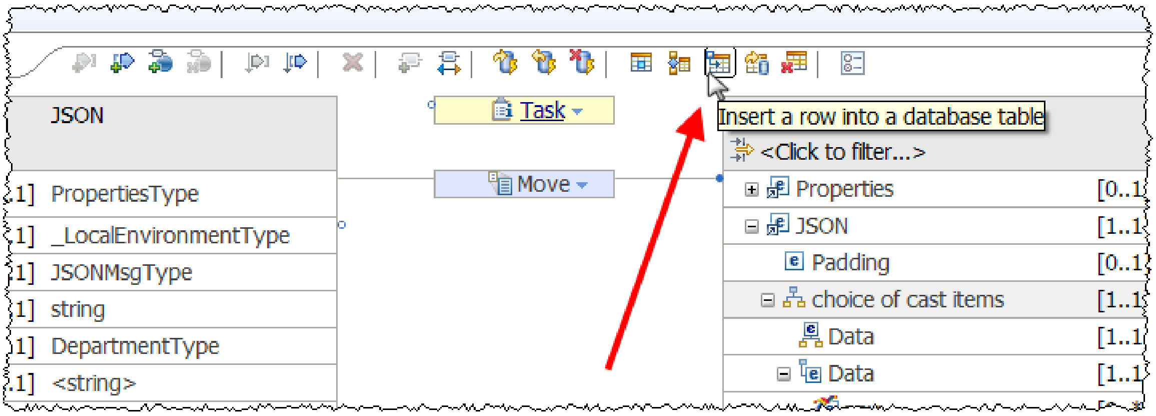
| | | |
|---|---|---|
| Mapping | [0..1] | _LocalEnvironmentMappingType |
| Database | [0..1] | _LocalEnvironmentDatabaseType |
| MQ | [0..1] | _LocalEnvironmentMQType |
| MQTT | [0..1] | _LocalEnvironmentMQTTType |
| REST | [0..1] | _LocalEnvironmentRESTType |
| Input | [0..1] | _RESTInputType |
| Method | [0..1] | string |
| Operation | [0..1] | string |
| Path | [0..1] | string |
| URI | [0..1] | string |
| Parameters | [0..1] | <Anonymous> |
| choice of cast items | [0..*] | |
| any | [1..1] | |
| departmentKey | [1..1] | string |
| Response | [0..1] | _RESTResponseType |
| TimeoutRequest | [0..1] | _LocalEnvironmentTimeoutRequestType |
| XSL | [0..1] | _LocalEnvironmentXSLType |

_8. Save the map (Ctrl S) to save your work so far – leave the mapping editor open.

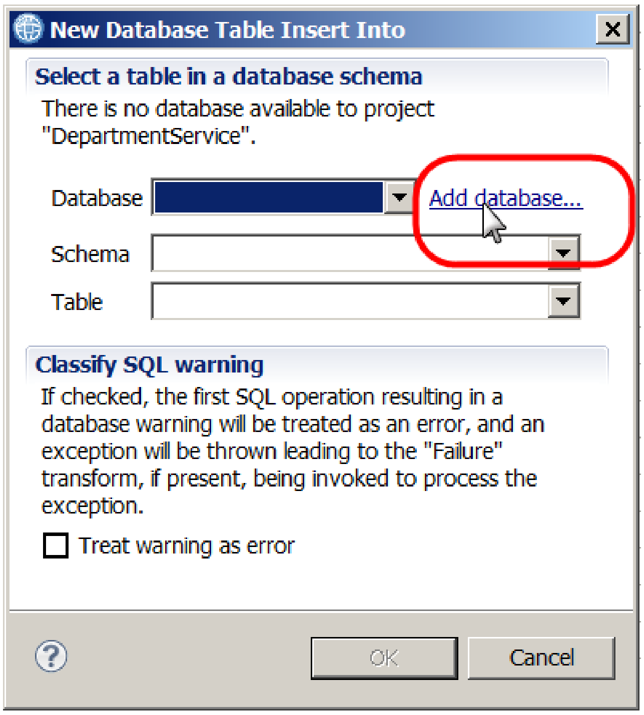
## 2.3.1 Add a row into the Database

In this next section you will add the mapping logic to add a record into the DEPARTMENT table in the HRDB database.

_1. Click the icon "Insert a row into a database table":



_2. In the "New Database Table Insert Into" window click Add Database:



New Database Table Insert Into

**Select a table in a database schema**

There is no database available to project "DepartmentService".

Database:

Add database...

Schema:

Table:

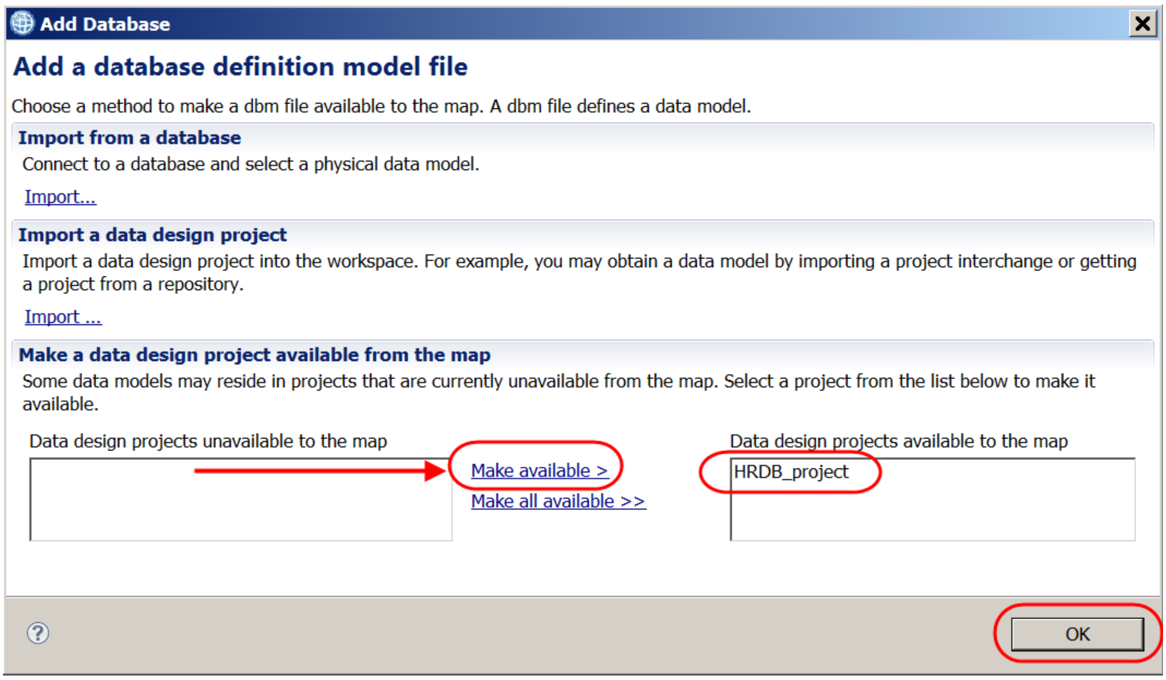
**Classify SQL warning**

If checked, the first SQL operation resulting in a database warning will be treated as an error, and an exception will be thrown leading to the "Failure" transform, if present, being invoked to process the exception.

□ Treat warning as error

**Note:** *The Database definition for HRDB is located in the HRDB Shared folder. If you add this folder as a Library Reference (right click on the HR_Service and choose Manage Library References, the HRDB database will automatically appear in the list of database schemas that you can choose above.*

_3. In the Add database window, select HRDB_project (*located in the HRDB Shared library*), then click “Make Available”. HRDB will then appear in the list of Data Design projects available to the map (screen capture shows the window after the HRDB_project has been made available). Press OK when finished.



Add Database

Add a database definition model file

Choose a method to make a dbm file available to the map. A dbm file defines a data model.

Import from a database

Connect to a database and select a physical data model.

Import...

Import a data design project

Import a data design project into the workspace. For example, you may obtain a data model by importing a project interchange or getting a project from a repository.

Import ...

Make a data design project available from the map

Some data models may reside in projects that are currently unavailable from the map. Select a project from the list below to make it available.

Data design projects unavailable to the map

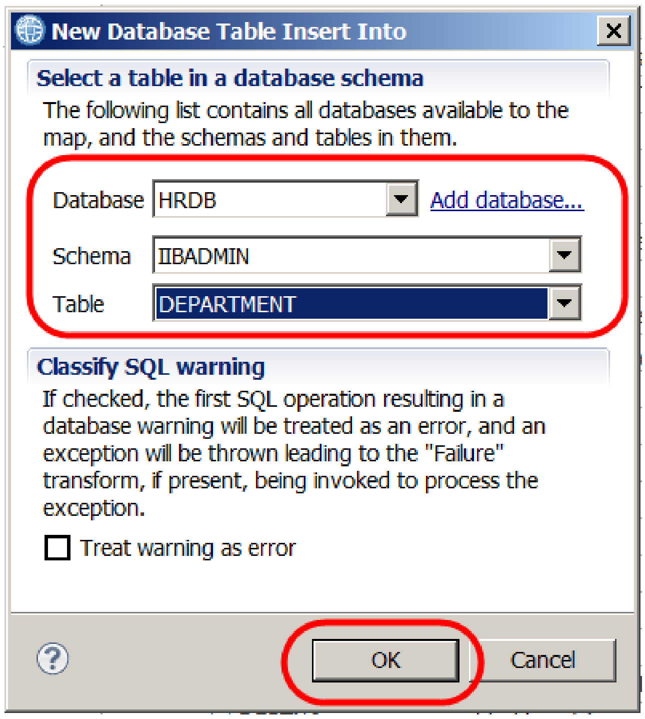
Make available >

Make all available >>

Data design projects available to the map

HRDB_project

_4. The New Database Table Insert Into window will now be updated to include the HRDB database. Select the DEPARTMENT table and click OK:



New Database Table Insert Into

Select a table in a database schema

The following list contains all databases available to the map, and the schemas and tables in them.

Database HRDB Add database...

Schema IIBADMIN

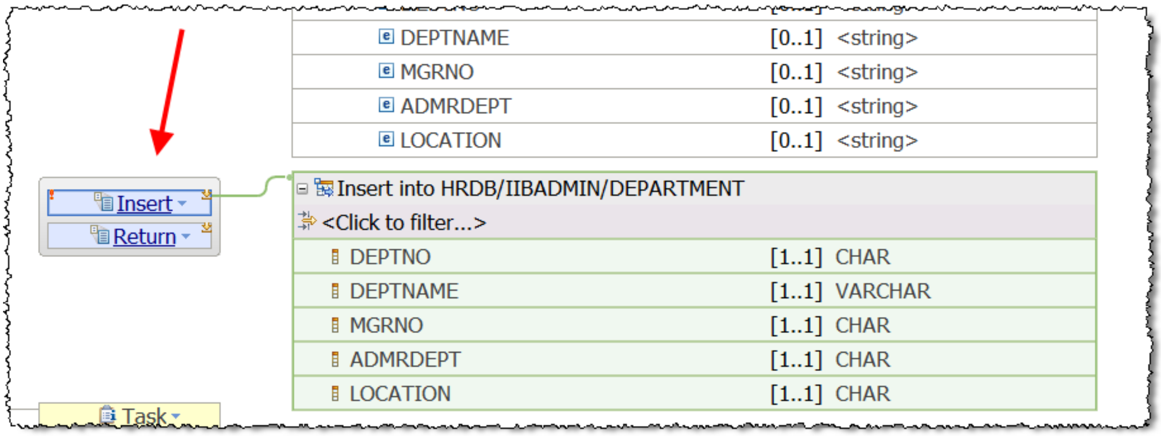
Table DEPARTMENT

Classify SQL warning

If checked, the first SQL operation resulting in a database warning will be treated as an error, and an exception will be thrown leading to the "Failure" transform, if present, being invoked to process the exception.

□ Treat warning as error

_5. The map will be updated with an Insert and Return transform:



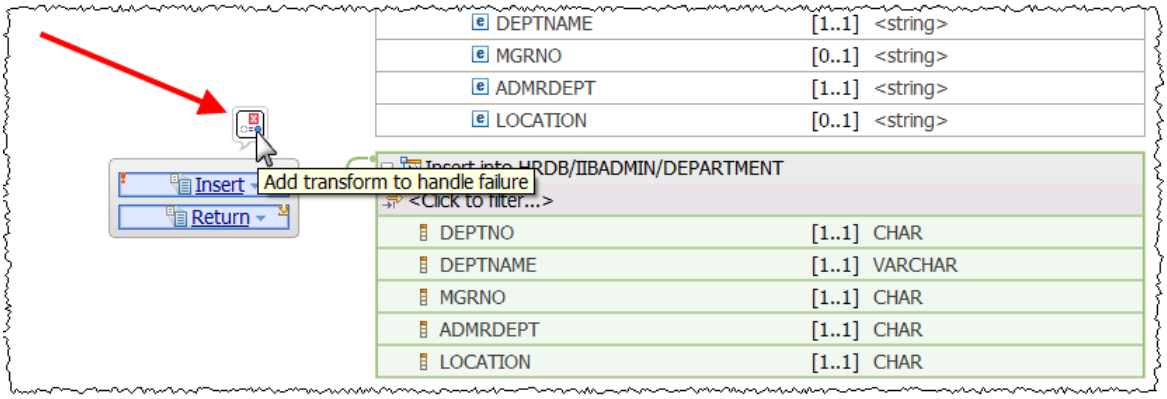
| | | |
|---|---|---|
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

Insert into HRDB/IIBADMIN/DEPARTMENT

<Click to filter...>

| | | |
|---|---|---|
| DEPTNO | [1..1] | CHAR |
| DEPTNAME | [1..1] | VARCHAR |
| MGRNO | [1..1] | CHAR |
| ADMRDEPT | [1..1] | CHAR |
| LOCATION | [1..1] | CHAR |

_6. Click the grey box around the Insert and Return transforms, a small icon will appear above the grey box. Click this icon to add a Failure transform to the box:



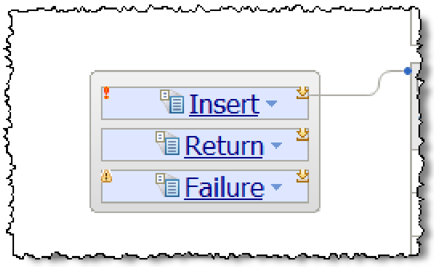
| | | |
|---|---|---|
| DEPTNAME | [1..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [1..1] | <string> |
| LOCATION | [0..1] | <string> |

Insert into HRDB/IIBADMIN/DEPARTMENT

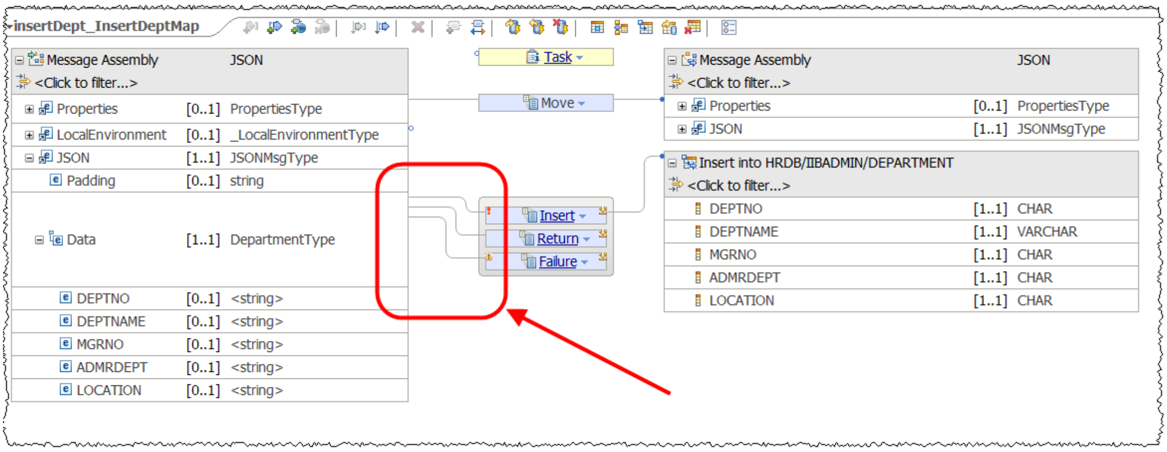
<Click to filter...>

| | | |
|---|---|---|
| DEPTNO | [1..1] | CHAR |
| DEPTNAME | [1..1] | VARCHAR |
| MGRNO | [1..1] | CHAR |
| ADMRDEPT | [1..1] | CHAR |
| LOCATION | [1..1] | CHAR |

There should now be three transforms in the grey box:



_7. Connect Data (of type DepartmentType) (defined in the Input Message Assembly) to each of the Insert, Return and Failure transforms (*minimise the JSON output structure so that you can see the insert box). To connect an element click on it and drag it to the transform you are going to connect it to*):



insertDept_InsertDeptMap

| Message Assembly | | JSON |
|---|---|---|
| <Click to filter...> | | |
| Properties | [0..1] | PropertiesType |
| LocalEnvironment | [0..1] | _LocalEnvironmentType |
| JSON | [1..1] | JSONMsgType |
| Padding | [0..1] | string |
| Data | [1..1] | DepartmentType |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

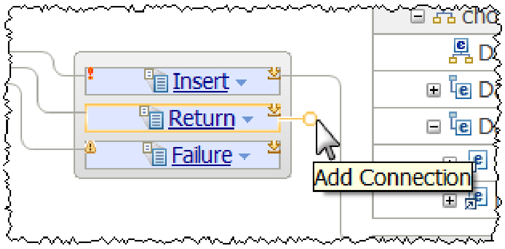
| Message Assembly | | JSON |
|---|---|---|
| <Click to filter...> | | |
| Properties | [0..1] | PropertiesType |
| JSON | [1..1] | JSONMsgType |

| Insert into HRDB/IIBADMIN/DEPARTMENT | | |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [1..1] | CHAR |
| DEPTNAME | [1..1] | VARCHAR |
| MGRNO | [1..1] | CHAR |
| ADMRDEPT | [1..1] | CHAR |
| LOCATION | [1..1] | CHAR |

_8. Expand the Output Message Assembly until you see the Data (of type `DepartmentResponse)` element.

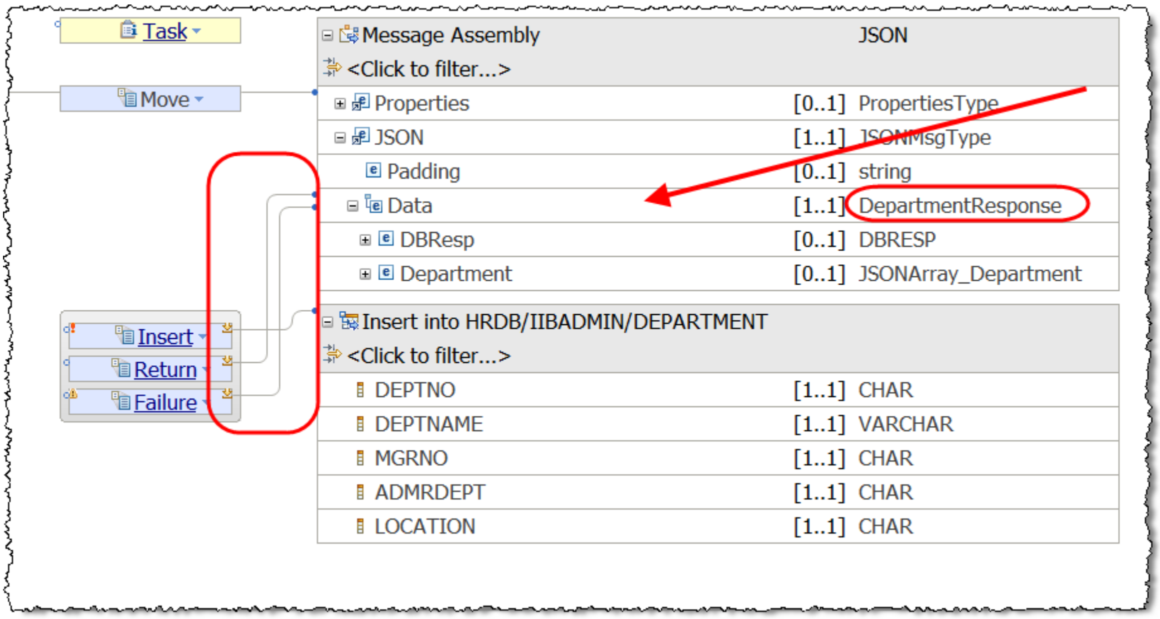
Connect the `Return` and `Failure` transforms to the Data element defined by `DepartmentResponse (type)` (in the Output Message Assembly).

(Note: to connect a transform to an element, hover over the transform until a yellow border



appears, then select the yellow circle marker and drag it to the destination elment).

The map will look like this when you have completed this step:



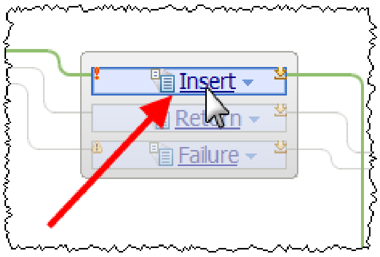
| Message Assembly | | JSON |
|---|---|---|
| <Click to filter...> | | |
| Properties | [0..1] | PropertiesType |
| JSON | [1..1] | JSONMsgType |
| Padding | [0..1] | string |
| Data | [1..1] | DepartmentResponse |
| DBResp | [0..1] | DBRESP |
| Department | [0..1] | JSONArray_Department |

| Insert into HRDB/IIBADMIN/DEPARTMENT | | |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [1..1] | CHAR |
| DEPTNAME | [1..1] | VARCHAR |
| MGRNO | [1..1] | CHAR |
| ADMRDEPT | [1..1] | CHAR |
| LOCATION | [1..1] | CHAR |

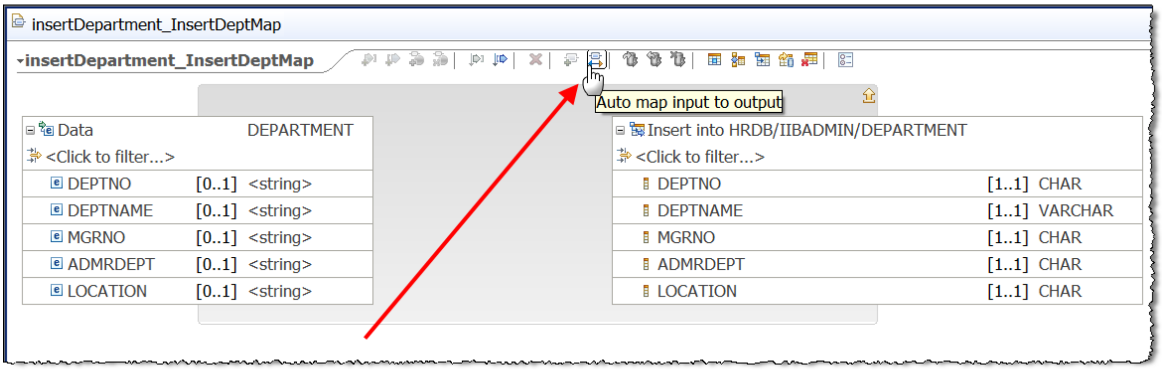
### 2.3.1.1Configure the Insert transform

In this next section you will navigate into the Insert transform and using a nested map, configure the input fields that will be used when the DEPARTMENT entry is added to the table.

_1. Click the word “Insert” in the Insert transform box:



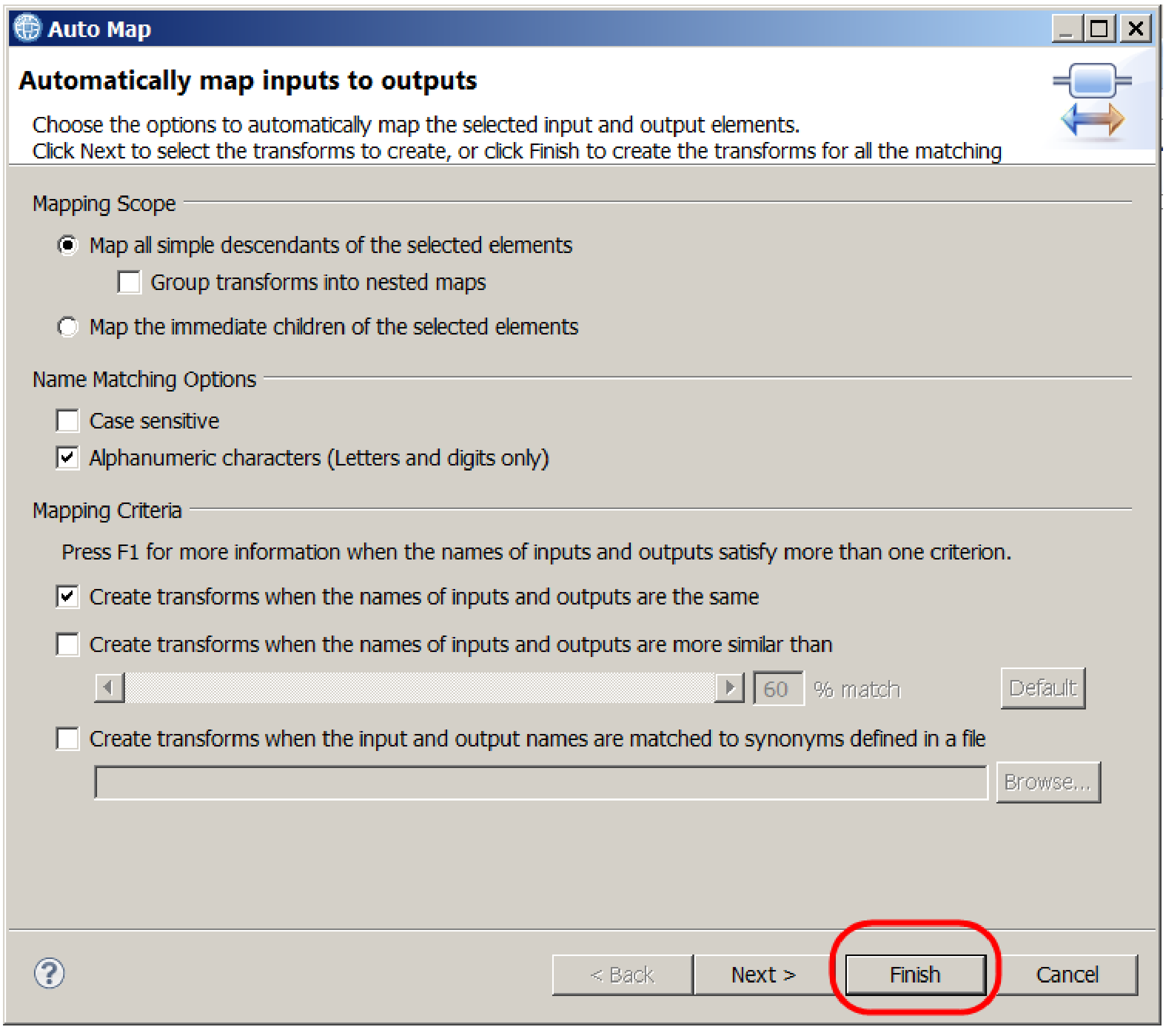
_2. The mapping editor will navigate you to the next lower level within the editor. The left of the map shows you the input data that is available on this level of the map, the right of the map shows you what DEPARTMENT table columns are available. The names of these are the same/similar, so we can use the “Auto map input to output” function. Click the Auto map icon:


insertDepartment_InsertDeptMap
insertDepartment_InsertDeptMap
Auto map input to output

| Data | | DEPARTMENT |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

| Insert into HRDB/IIBADMIN/DEPARTMENT | | |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [1..1] | CHAR |
| DEPTNAME | [1..1] | VARCHAR |
| MGRNO | [1..1] | CHAR |
| ADMRDEPT | [1..1] | CHAR |
| LOCATION | [1..1] | CHAR |

_3. Accept the defaults on the Auto Map window and click Finish:



**Auto Map**

**Automatically map inputs to outputs**

Choose the options to automatically map the selected input and output elements.
Click Next to select the transforms to create, or click Finish to create the transforms for all the matching

Mapping Scope

● Map all simple descendants of the selected elements

□ Group transforms into nested maps

○ Map the immediate children of the selected elements

Name Matching Options

□ Case sensitive

☑ Alphanumeric characters (Letters and digits only)

Mapping Criteria

Press F1 for more information when the names of inputs and outputs satisfy more than one criterion.

☑ Create transforms when the names of inputs and outputs are the same

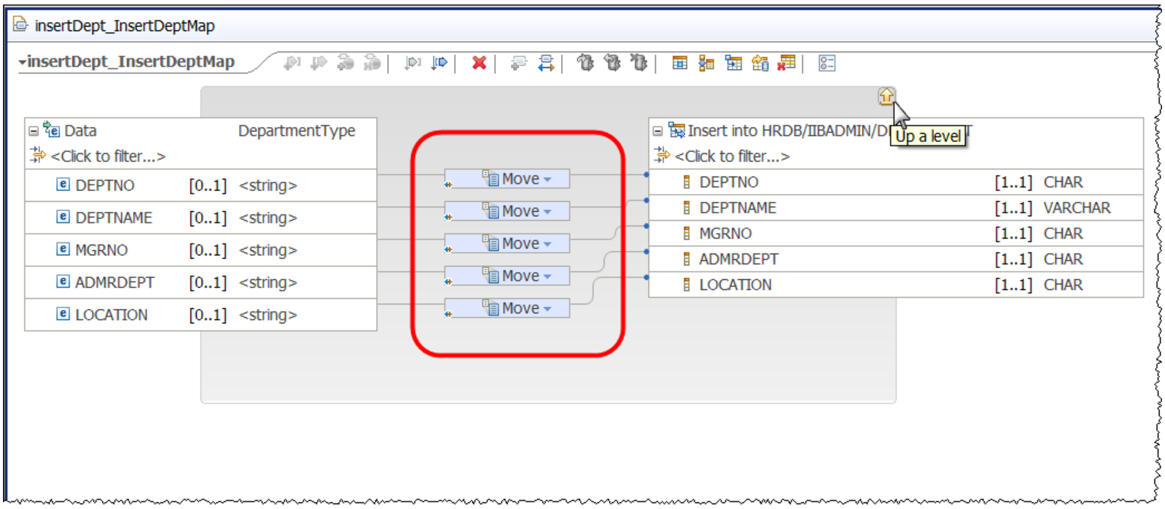
□ Create transforms when the names of inputs and outputs are more similar than

60 % match Default

□ Create transforms when the input and output names are matched to synonyms defined in a file

Browse...

_4. The map will now have a Move transform for every like-named element. When you have reviewed the transforms click the yellow up-arrow to navigate back to the previous map level:



insertDept_InsertDeptMap

insertDept_InsertDeptMap

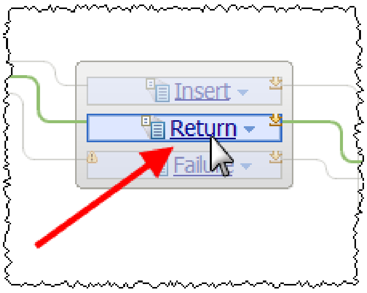
| Data | | DepartmentType |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

| Insert into HRDB/IIBADMIN/D | | |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [1..1] | CHAR |
| DEPTNAME | [1..1] | VARCHAR |
| MGRNO | [1..1] | CHAR |
| ADMRDEPT | [1..1] | CHAR |
| LOCATION | [1..1] | CHAR |

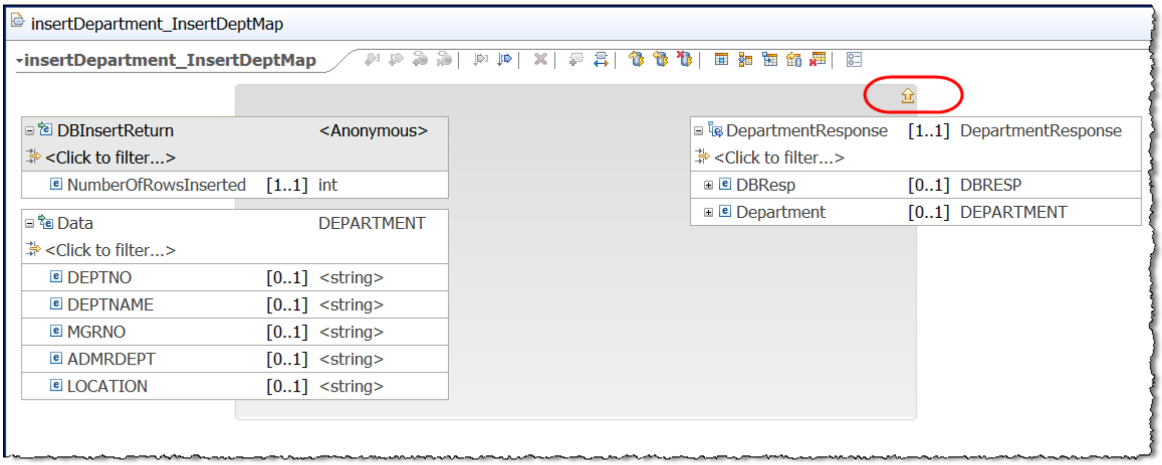
### 2.3.1.2 Configure the Return transform

In this section you will navigate into the `Return` transform where using a nested map, you will configure what data will be returned to the calling REST API when the request is successful.

_1. Click the word "Return" in the Return transform box:



This will navigate the map editor into the nested map for the Return transform (the yellow up arrow shows you that you are in a nested map) :



insertDepartment_InsertDeptMap

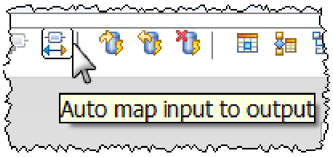
insertDepartment_InsertDeptMap

| DBInsertReturn | | <Anonymous> |
|---|---|---|
| <Click to filter...> | | |
| NumberOfRowsInserted | [1..1] | int |

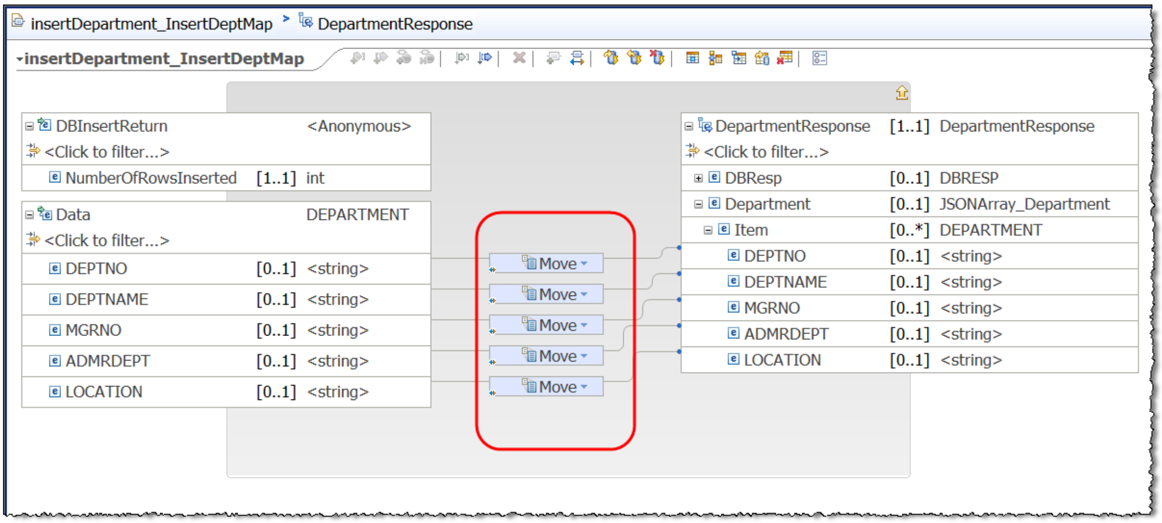
| Data | | DEPARTMENT |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

| DepartmentResponse | [1..1] | DepartmentResponse |
|---|---|---|
| <Click to filter...> | | |
| DBResp | [0..1] | DBRESP |
| Department | [0..1] | DEPARTMENT |

_2. Use the Auto map feature to map like-named elements (refer to previous section for detail):



When complete the nested map will look like this:



insertDepartment_InsertDeptMap > DepartmentResponse

insertDepartment_InsertDeptMap

| DBInsertReturn | | <Anonymous> |
|---|---|---|
| <Click to filter...> | | |
| NumberOfRowsInserted | [1..1] | int |

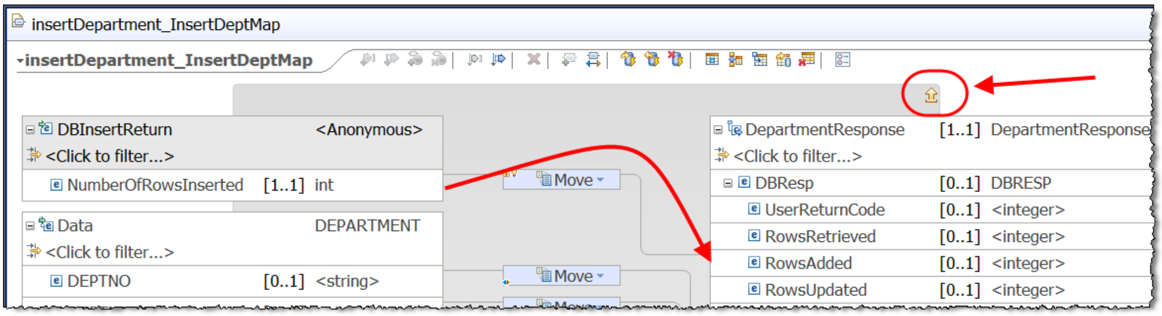
| Data | | DEPARTMENT |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

Move<br>Move<br>Move<br>Move<br>Move

| DepartmentResponse | [1..1] | DepartmentResponse |
|---|---|---|
| <Click to filter...> | | |
| DBResp | [0..1] | DBRESP |
| Department | [0..1] | JSONArray_Department |
| Item | [0..*] | DEPARTMENT |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

_3. In the output message Assemply, expand DBRESP and connect "`NumberOfRowsInserted`" (in the input message assembly) to `RowsAdded` in `DBRESP`.

When Complete use the yellow up arrow to navigate back to the higher level in the map:



insertDepartment_InsertDeptMap

insertDepartment_InsertDeptMap

| DBInsertReturn | | <Anonymous> |
|---|---|---|
| <Click to filter...> | | |
| NumberOfRowsInserted | [1..1] | int |

| Data | | DEPARTMENT |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [0..1] | <string> |

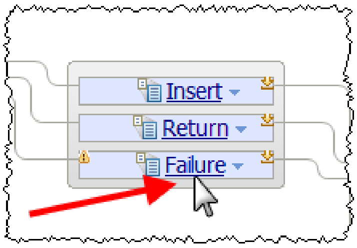
Move<br>Move

| DepartmentResponse | [1..1] | DepartmentResponse |
|---|---|---|
| <Click to filter...> | | |
| DBResp | [0..1] | DBRESP |
| UserReturnCode | [0..1] | <integer> |
| RowsRetrieved | [0..1] | <integer> |
| RowsAdded | [0..1] | <integer> |
| RowsUpdated | [0..1] | <integer> |

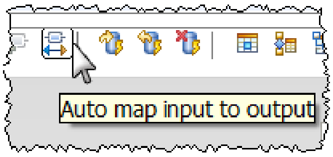
## 2.3.1.3 Configure the Failure transform

In this section you will navigate into the Failure transform where you will configure (in a nested map) what data will be returned to the subflow when the request fails.

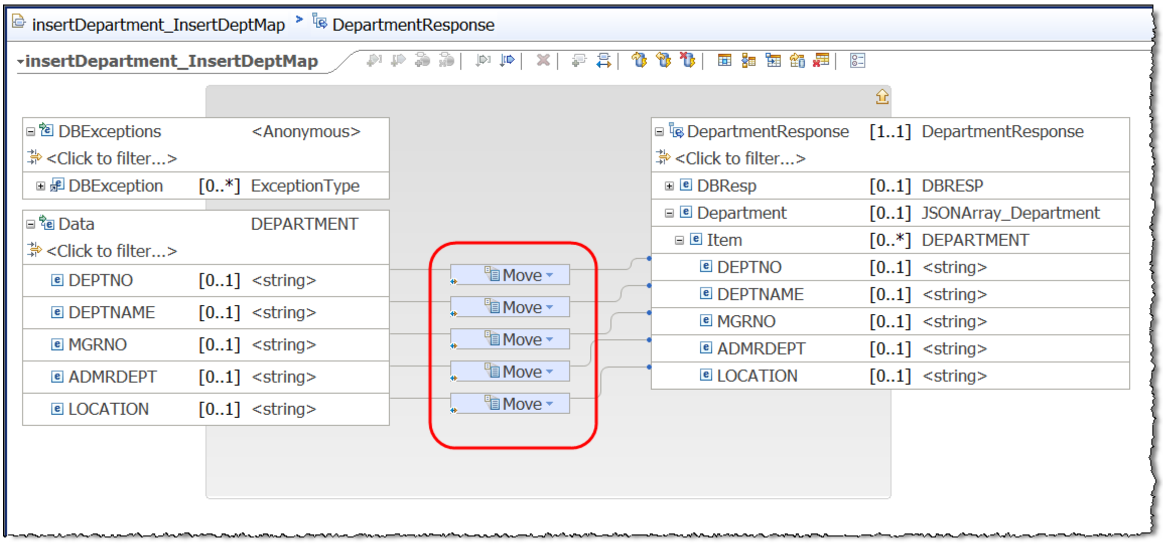
_1. Click the word "Failure" in the Failure transform box:



_2. Use the Auto map feature to map like-named elements (refer to previous section for detail):



Your nested map will look like this:

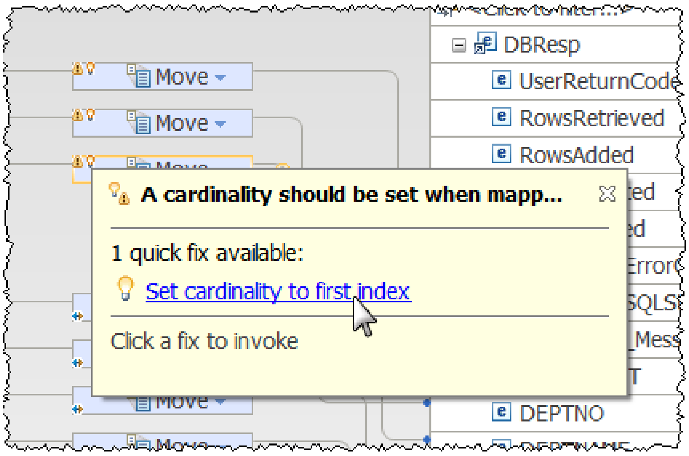


_3. Expand (the Input) DBException element and (output) DBResp elements.

Map

1) `DBException.Message` -> `DBResp.SQL_Error_Message`
2) `DBException.SQLState` -> `DBResp.SQLState_SQLState`
3) `DBException.ErrorCode` -> `DBResp.SQLCODE_Errorcode`

In all three cases set the cardinality to the first index (hover over the Yellow light bulb and select “Set cardinality to first Index”:

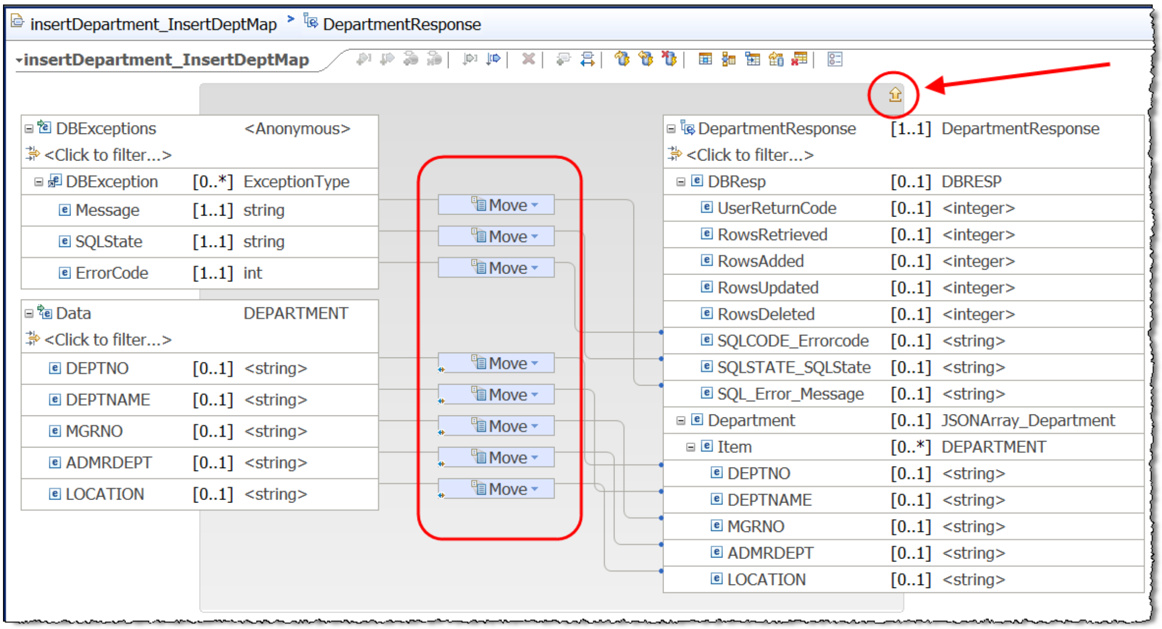

A cardinality should be set when mapp...

1 quick fix available:

Set cardinality to first index

Click a fix to invoke

_4. When complete the nested map will look like this. Use the yellow up-arrow to return to the higher level within the  map:



insertDepartment_InsertDeptMap > DepartmentResponse

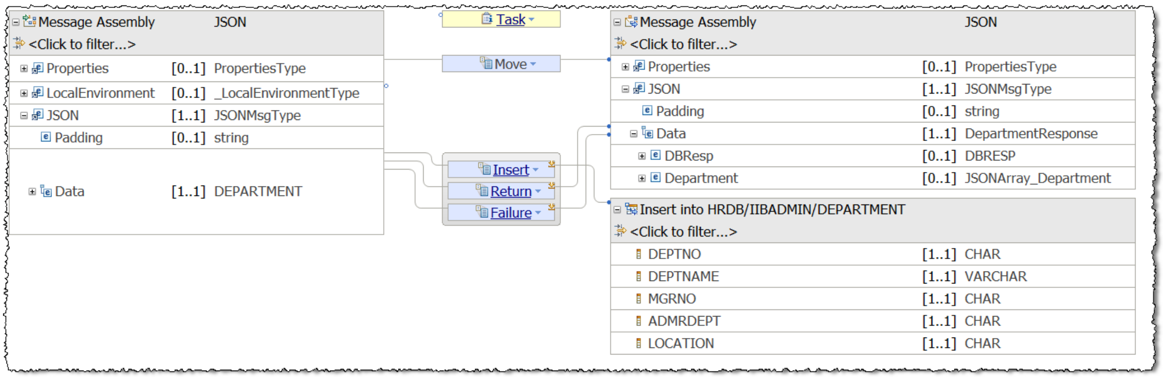
insertDepartment_InsertDeptMap

| | | |
|---|---|---|
| DBExceptions | | <Anonymous> |
| <Click to filter...> | | |
| DBException | [0..*] | ExceptionType |
| Message | [1..1] | string |
| SQLState | [1..1] | string |
| ErrorCode | [1..1] | int |
| Data | | DEPARTMENT |
| <Click to filter...> | | |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

| | | |
|---|---|---|
| DepartmentResponse | [1..1] | DepartmentResponse |
| <Click to filter...> | | |
| DBResp | [0..1] | DBRESP |
| UserReturnCode | [0..1] | <integer> |
| RowsRetrieved | [0..1] | <integer> |
| RowsAdded | [0..1] | <integer> |
| RowsUpdated | [0..1] | <integer> |
| RowsDeleted | [0..1] | <integer> |
| SQLCODE_Errorcode | [0..1] | <string> |
| SQLSTATE_SQLState | [0..1] | <string> |
| SQL_Error_Message | [0..1] | <string> |
| Department | [0..1] | JSONArray_Department |
| Item | [0..*] | DEPARTMENT |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

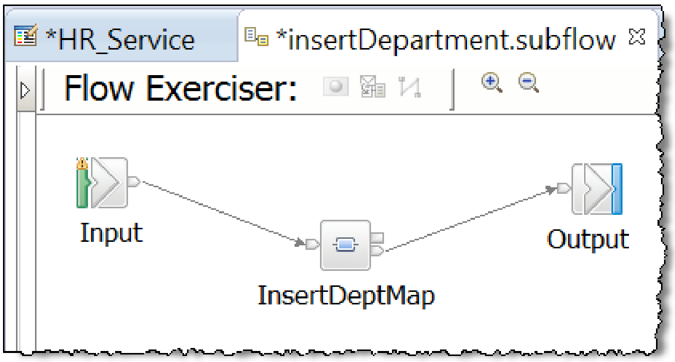
## 2.3.2 Review and Save the InsertDept mapping node

The mapping node now has sufficient configuration to be able to insert a row into the DEPARTMENT table based on data passed to the REST API.

| | |
|---|---|
| _1. | The main insertDept map will look like this after the configuration of the Insert Return and Failure transforms:<br><br>Message Assembly JSON<br><Click to filter...><br>Properties [0..1] PropertiesType<br>LocalEnvironment [0..1] _LocalEnvironmentType<br>JSON [1..1] JSONMsgType<br>Padding [0..1] string<br>Data [1..1] DEPARTMENT<br>Task<br>Move<br>Insert<br>Return<br>Failure<br>Message Assembly JSON<br><Click to filter...><br>Properties [0..1] PropertiesType<br>JSON [1..1] JSONMsgType<br>Padding [0..1] string<br>Data [1..1] DepartmentResponse<br>DBResp [0..1] DBRESP<br>Department [0..1] JSONArray_Department<br>Insert into HRDB/IIBADMIN/DEPARTMENT<br><Click to filter...><br>DEPTNO [1..1] CHAR<br>DEPTNAME [1..1] VARCHAR<br>MGRNO [1..1] CHAR<br>ADMRDEPT [1..1] CHAR<br>LOCATION [1..1] CHAR |
| _2. | Save (ctrl s) and close the insertDept mapping editor. |

## 2.3.3 Complete the insertDepartment subflow

With the mapping node complete, complete the insertDept subflow

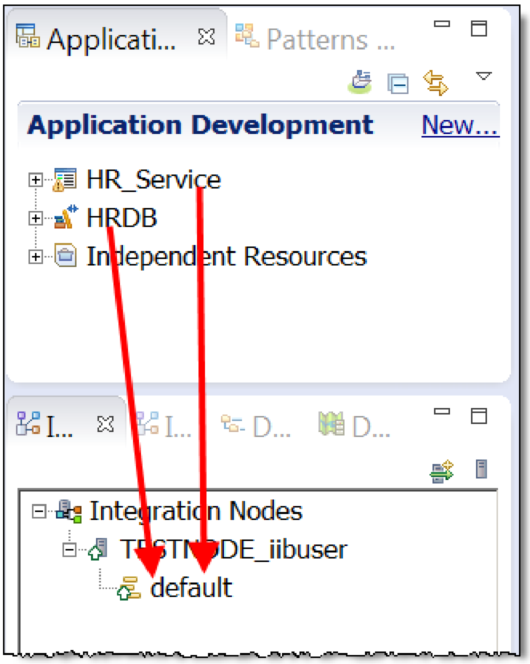
| | |
|---|---|
| _1. | Connect the Input and Output nodes to the InsertDept map as follows:<br><br> *HR_Service *insertDepartment.subflow Flow Exerciser: Input InsertDeptMap Output |
| _2. | Save (ctrl S) and exit the subflow editor. |
| _3. | Save and exit the HR_Service REST API Editor. |

# 3. Test the REST API

You will now test the REST API with the insertDepartment operation implemented. Note: the implementation of the retrieveDepartment operation is documented in the appendix of this lab guide.

## 3.1 Deploy the resources

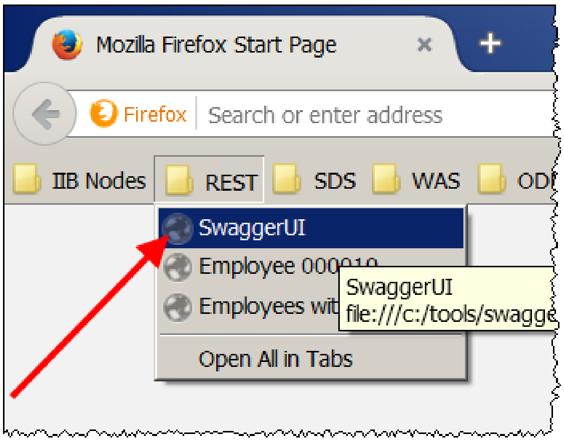
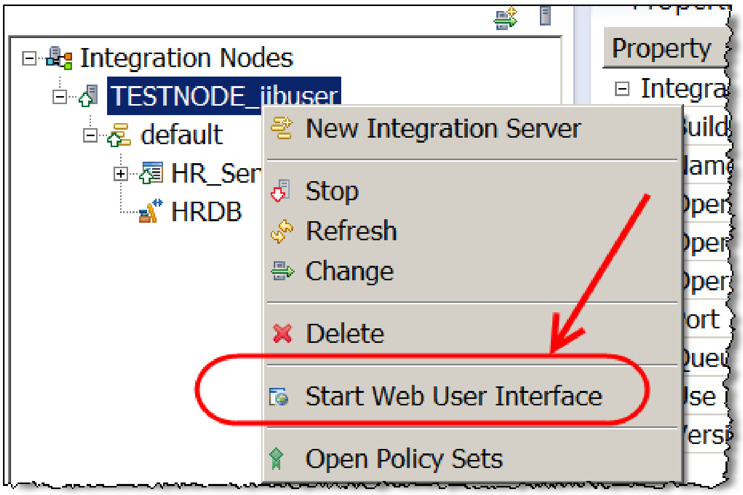
<table>
<tr><td>_1.</td><td>In the Integration Nodes view, right click on the default Integration server for TESTNODE_iibuser and select <code>Delete &gt; All flows and resources</code> to clear out any existing applications or REST APIs.</td></tr>
<tr><td>_2.</td><td>Deploy (drag and drop) the <code>HRDB</code> shared library and <code>HR_Service</code> to the default integration server (<em>in that order – if HRDB is not already deployed on the Integration Server when HR_Service is deployed, the deployment of HR_Service will fail as HR_Service has dependencies on resources in HRDB</em>)



</td></tr>
</table>

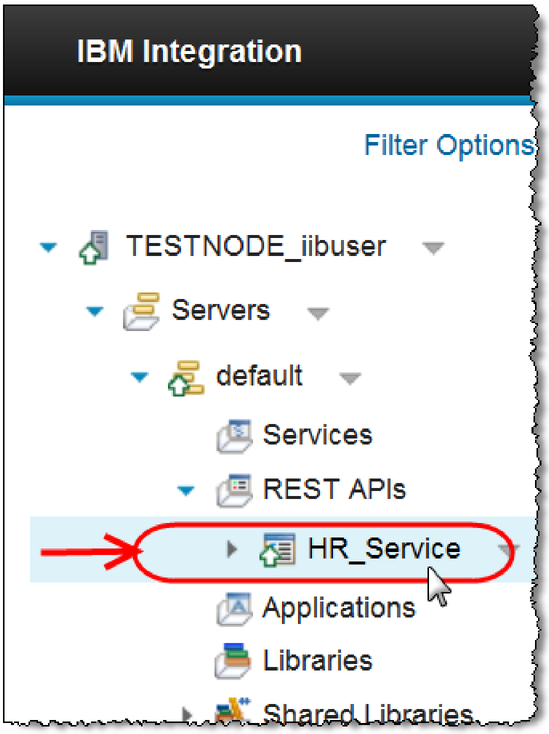
## 3.2 Test using SwaggerUI

Note: If you are using your own installation for this test, you will need to provide your own instance of SwaggerUI.

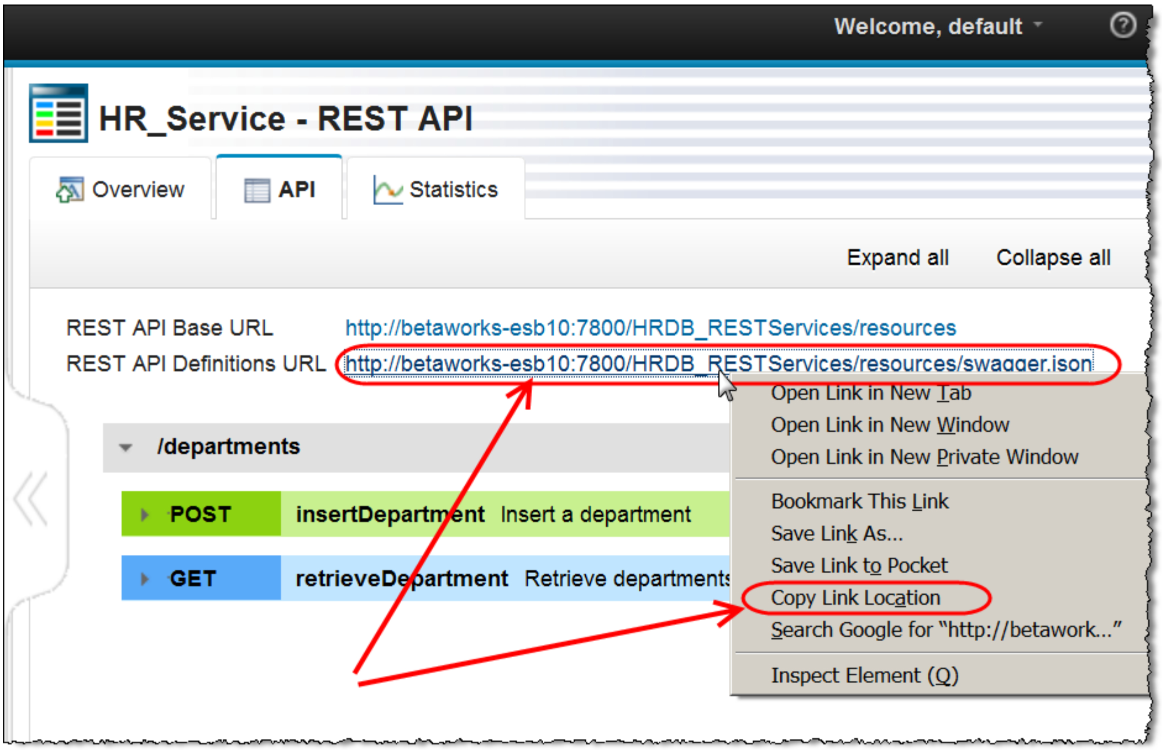
| | |
|---|---|
| __1. | Open a Firefox browser and select SwaggerUI from the REST folder.<br><br> Mozilla Firefox Start Page<br>Firefox Search or enter address<br>IIB Nodes<br>REST<br>SDS<br>WAS<br>ODI<br>SwaggerUI<br>Employee 000010<br>Employees wit<br>SwaggerUI<br>file:///c:/tools/swagge<br>Open All in Tabs<br>The default "Pet Store" application will open in SwaggerUI. |
| __2. | Switch to the Integration Toolkit.<br>Right click on TESTNODE_iibuser and click "Start Web User interface":<br><br> Integration Nodes<br>TESTNODE_iibuser<br>default<br>HR_Ser<br>HRDB<br>New Integration Server<br>Stop<br>Refresh<br>Change<br>Delete<br>Start Web User Interface<br>Open Policy Sets<br>Property<br>Integra |

_3. After a few seconds the Integration Web UI for TESTNODE_iibuser will open in a browser window.

In the Web UI navigate to the HR_Service (TESTNODE_iibuser > Servers > default > REST APIs). Click the HR_Service name.



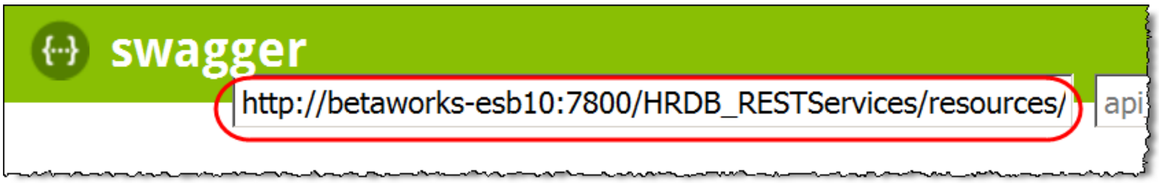
_4. In the API tab right click on the REST API Definitions URL and select “Copy Link location”



REST API Base URL http://betaworks-esb10:7800/HRDB_RESTServices/resources

REST API Definitions URL http://betaworks-esb10:7800/HRDB_RESTServices/resources/swagger.json

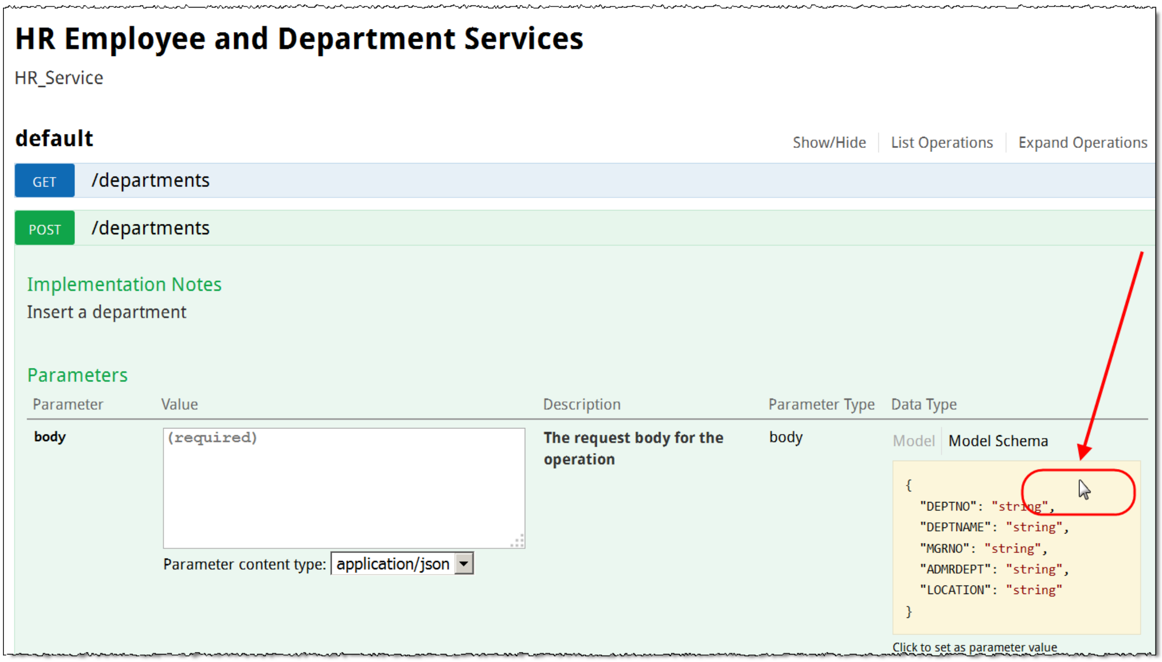
_5. Switch back in the SwaggerUI tool, paste the value into the URL field and press Enter:



_6. Click default where the HR_Service is running (note: this is a collective name for the resources in the swagger.json file – not the name of the Intregration Server). This will show the two operations GET and POST.

_7. Click POST to expose the implementation you added earlier.

Note the Body of the operation does not have pre-filled values.
Click the yellow background describing the model schema to copy the required fields into the body parameter:



## HR Employee and Department Services

HR_Service

**default**

GET /departments

POST /departments

Implementation Notes

Insert a department

Parameters

| Parameter | Value | Description | Parameter Type | Data Type |
|---|---|---|---|---|
| **body** | (required)<br>Parameter content type: application/json | **The request body for the operation** | body | Model Model Schema |

```
{
  "DEPTNO": "string",
  "DEPTNAME": "string",
  "MGRNO": "string",
  "ADMRDEPT": "string",
  "LOCATION": "string"
}
```

Click to set as parameter value

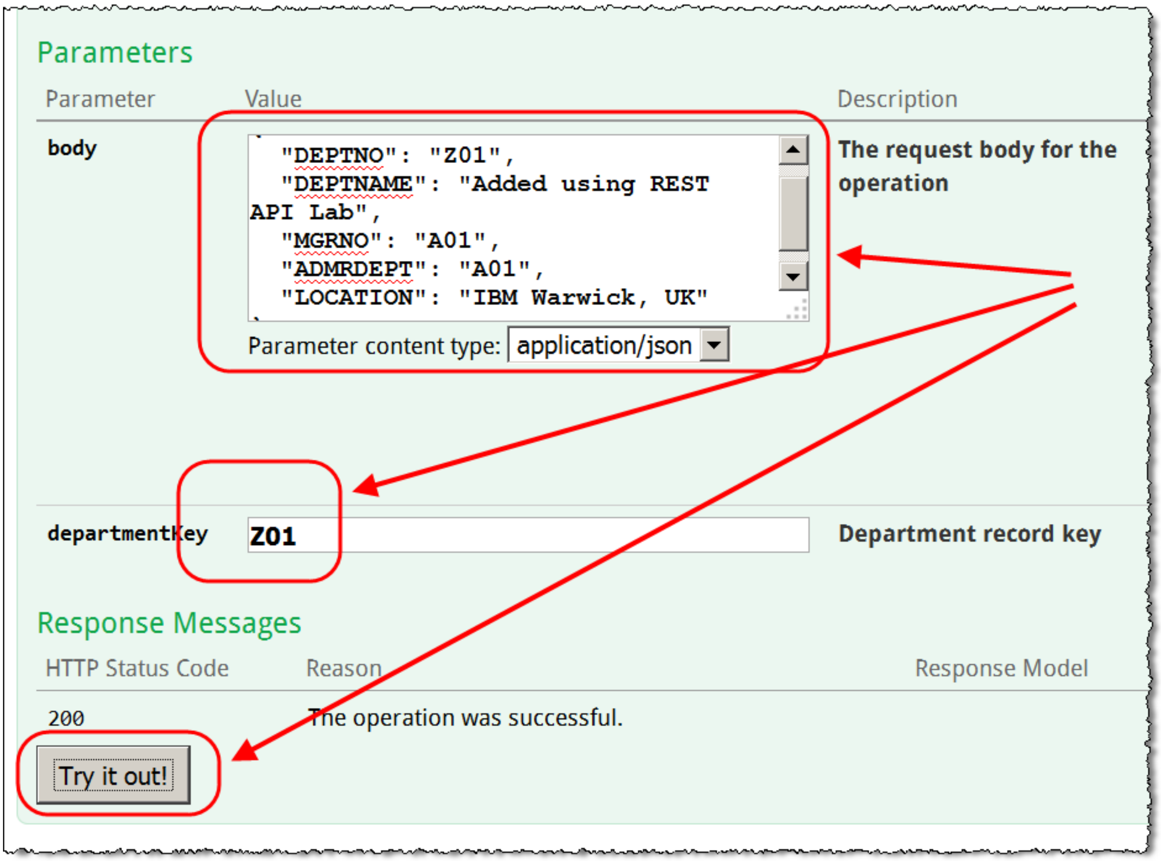
_8. In the **body** parameter, overtype the word “string” with:

DEPTNO: `Z01`
DEPTNAME: `Added using Rest API Lab`
MGRNO: `A01`
ADMRDEPT: `A01`
LOCATION: `IBM Warwick`

Note that double-clicking each element name automatically highlights it, in common with most ediors, making it easy to change the value.

Set **departmentKey** parameter to `Z01`

Press Try it Out when complete:



Parameters

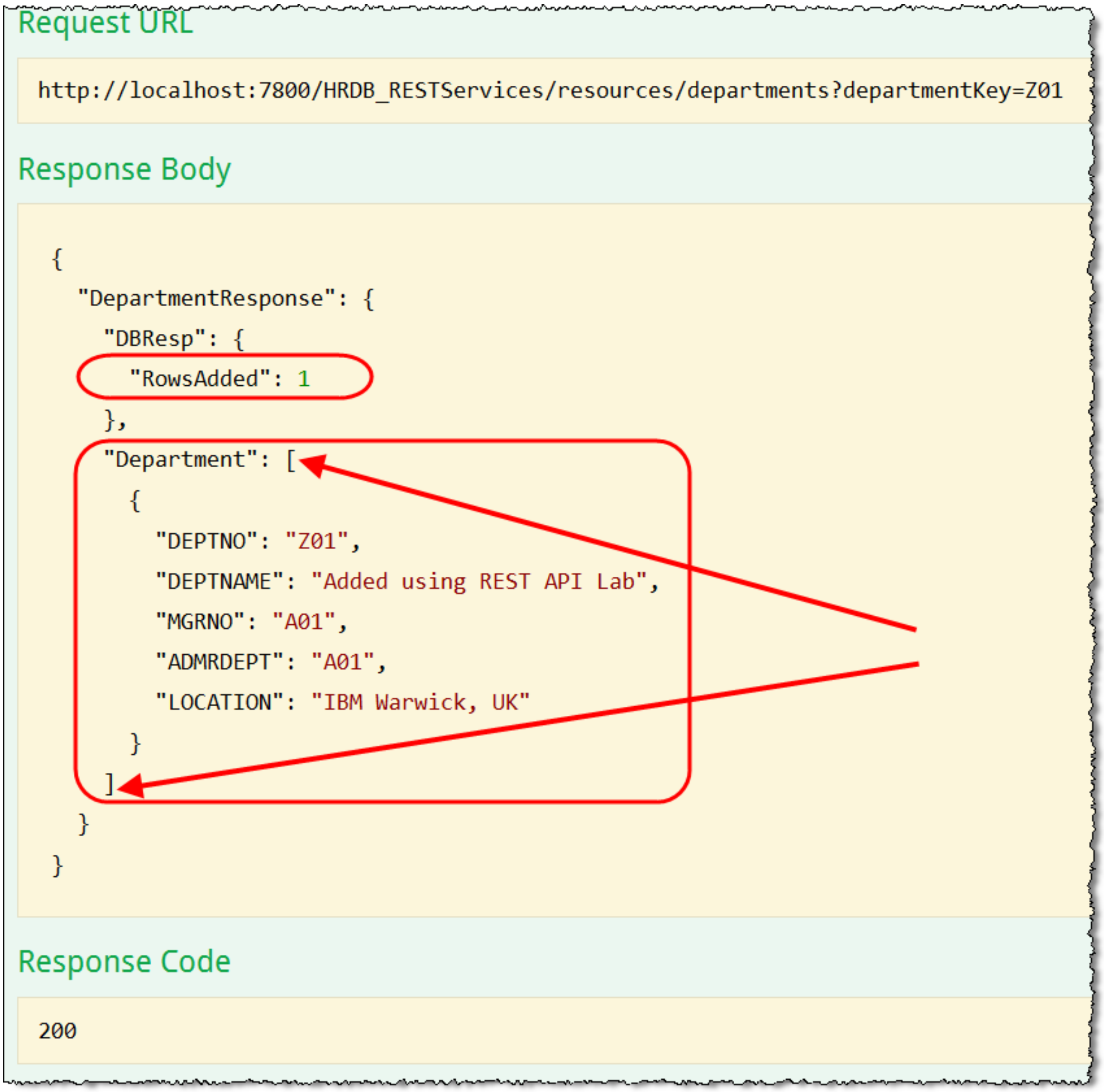
| Parameter | Value | Description |
|---|---|---|
| body | "DEPTNO": "Z01",<br>"DEPTNAME": "Added using REST API Lab",<br>"MGRNO": "A01",<br>"ADMRDEPT": "A01",<br>"LOCATION": "IBM Warwick, UK"<br>Parameter content type: application/json | The request body for the operation |
| departmentKey | Z01 | Department record key |

Response Messages

| HTTP Status Code | Reason | Response Model |
|---|---|---|
| 200 | The operation was successful. | |

_9. After a few seconds the response (response code 200) will appear under the Try it out! Button:

Notes:
(1) RowsAdded will be set to 1 and the inserted entry will be part of the JSON response:
(2) DepartmentResponse contains the DBResp and Department elements, the Department element response is a json array the data for the array is encased in square brackets [ ]



Request URL

```
http://localhost:7800/HRDB_RESTServices/resources/departments?departmentKey=Z01
```

Response Body

```
{
  "DepartmentResponse": {
    "DBResp": {
      "RowsAdded": 1
    },
    "Department": [
      {
        "DEPTNO": "Z01",
        "DEPTNAME": "Added using REST API Lab",
        "MGRNO": "A01",
        "ADMRDEPT": "A01",
        "LOCATION": "IBM Warwick, UK"
      }
    ]
  }
}
```

Response Code

```
200
```

_10 Optionally open a DB2 window to check the DB2 table has the inserted entry.

_11 Leave the SwaggerUI page open you will be using it to test out the optional part of this lab.

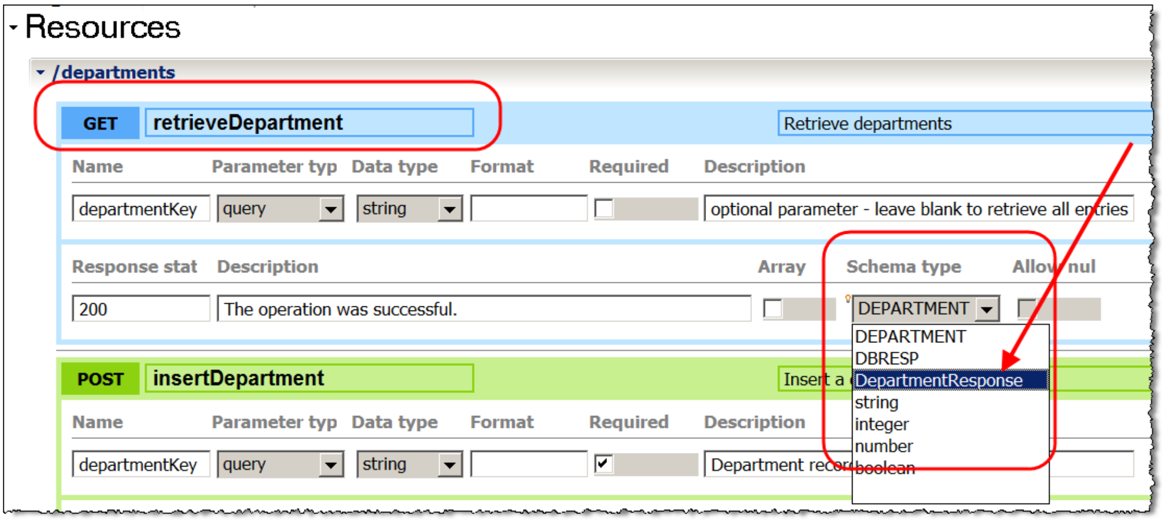
# Appendix (Optional)

## 4. Implement the retrieveDepartment (GET) operation

This optional part of this lab guide will guide you through implementing the retrieveDepartment (GET operation). It will demonstrate how an optional query parameter in a REST API can be used to return different values depending on the value passed in the input parameter. For example, when the `departmentKey` query input parameter is omitted, all entries from the DEPARTMENT table will be returned in the response. If a value is specified in the input, the response will return entries from the department table with a key “like” the value specified in the input.

__1. Double click on the **REST API Description** in the REST API for HR_Service to open the REST API editor.

__2. In the blue retrieveDepartment GET operation, change the value of Schema type for the response to **DepartmentResponse** (this will ensure that the map you will create in the next few sections has the correct DepartmentResponse output schema):



Resources

/departments

GET retrieveDepartment — Retrieve departments

| Name | Parameter typ | Data type | Format | Required | Description |
|---|---|---|---|---|---|
| departmentKey | query | string | | □ | optional parameter - leave blank to retrieve all entries |

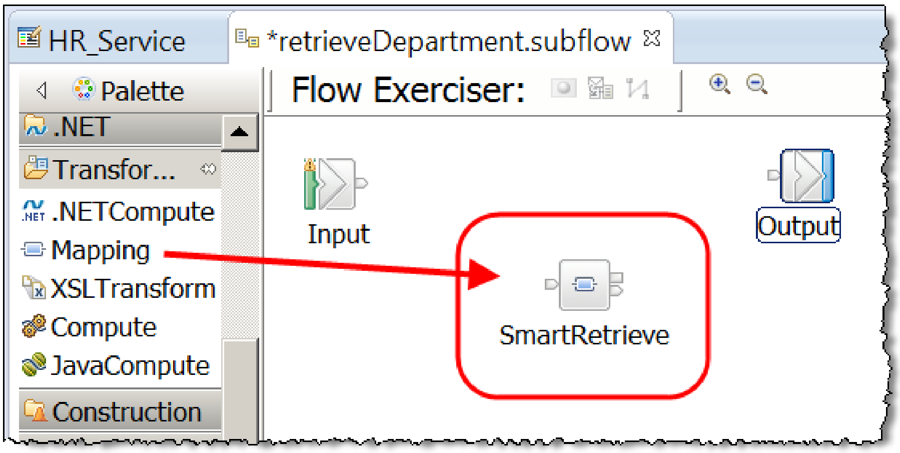
| Response stat | Description | Array | Schema type | Allow nul |
|---|---|---|---|---|
| 200 | The operation was successful. | □ | DEPARTMENT | □ |

POST insertDepartment — Insert a

| Name | Parameter typ | Data type | Format | Required | Description |
|---|---|---|---|---|---|
| departmentKey | query | string | | ☑ | Department reco |

__3. Click “**Create a subflow for the operation**” in the (blue) retrieveDepartment Operation.

__4. Drop a mapping node on to the subflow, call it **SmartRetrieve**.



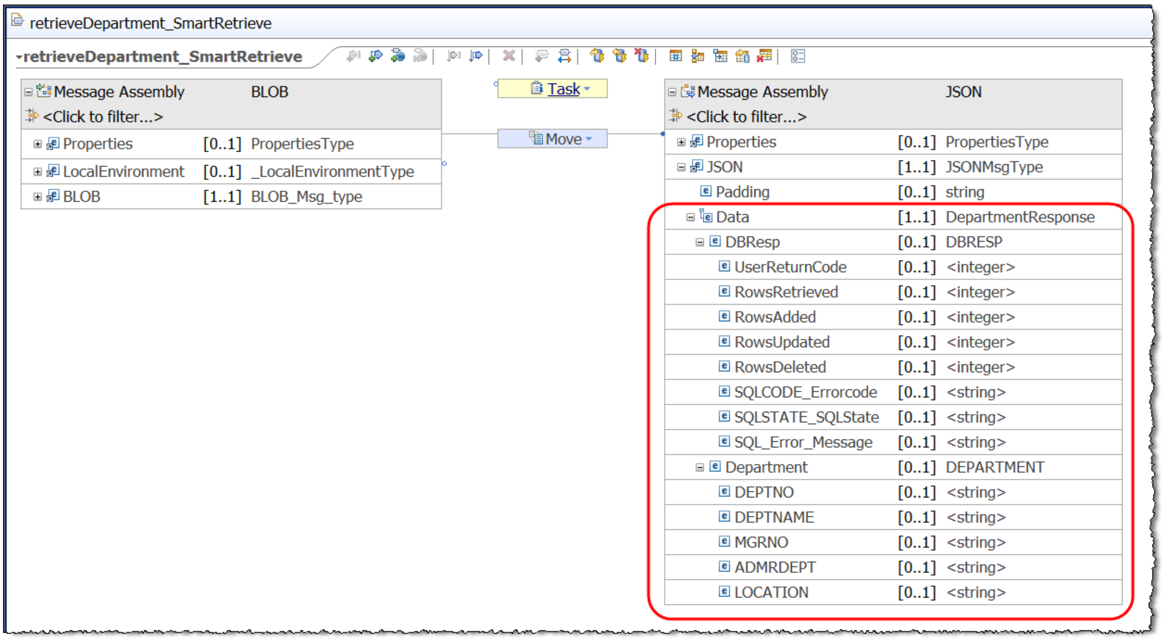
## 4.1 Configure the SmartRetrieve Map

This map will be used to retrieve data from the DEPARTMENT table. What is retrieved will be dependent on the input (optional) parameter **departmentKey**. If departmentKey contains data, it will be used as a key retrieve a specific record from the table. If departmentKey does not contain data, all records from the table will be retrieved and passed back in the REST response.

To facilitate this logic you will add and “`if`” and “`else`” transform to the map and configure each accordingly.

_1. Double click the SmartRetrieve mapping node, accept the defaults (ie. the mapping node will be aware of (and tied to) the REST API) and click Finish.

_2. Note the mapping node has three main input elements: Properties; LocalEnvironment; BLOB and a JSON Output Message Assembly with an element called Data, which will have been defined as having a type of “DepartmentResponse” (which also contains the DBResp and Department elements):



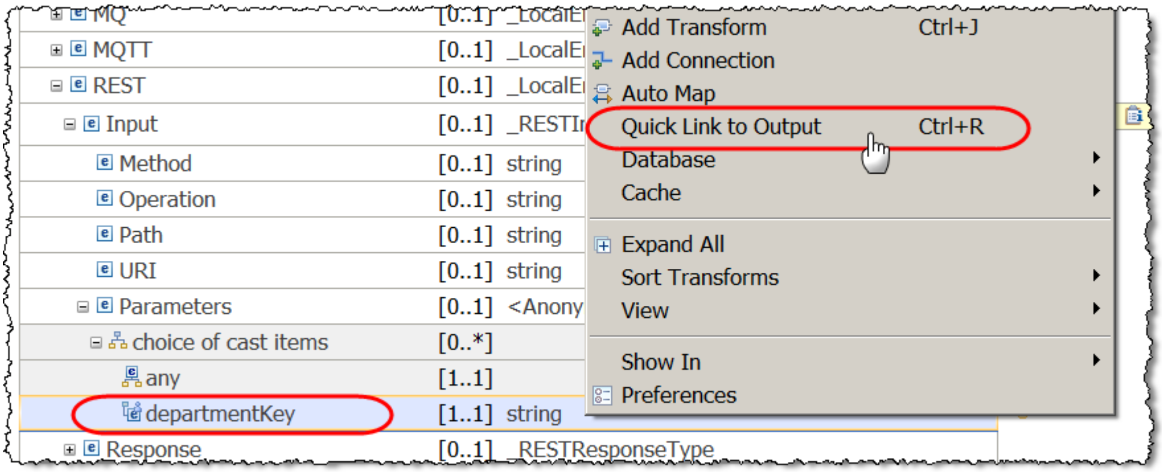
retrieveDepartment_SmartRetrieve

retrieveDepartment_SmartRetrieve

| Message Assembly | | BLOB |
|---|---|---|
| <Click to filter...> | | |
| Properties | [0..1] | PropertiesType |
| LocalEnvironment | [0..1] | _LocalEnvironmentType |
| BLOB | [1..1] | BLOB_Msg_type |

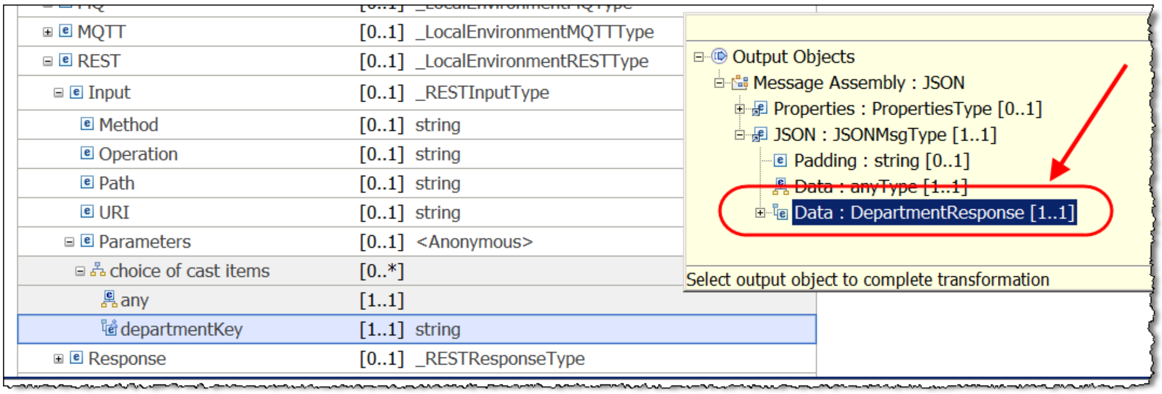
| Message Assembly | | JSON |
|---|---|---|
| <Click to filter...> | | |
| Properties | [0..1] | PropertiesType |
| JSON | [1..1] | JSONMsgType |
| Padding | [0..1] | string |
| Data | [1..1] | DepartmentResponse |
| DBResp | [0..1] | DBRESP |
| UserReturnCode | [0..1] | <integer> |
| RowsRetrieved | [0..1] | <integer> |
| RowsAdded | [0..1] | <integer> |
| RowsUpdated | [0..1] | <integer> |
| RowsDeleted | [0..1] | <integer> |
| SQLCODE_Errorcode | [0..1] | <string> |
| SQLSTATE_SQLState | [0..1] | <string> |
| SQL_Error_Message | [0..1] | <string> |
| Department | [0..1] | DEPARTMENT |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

_3. On the Input Message Assembly, expand `LocalEnvironment> REST> Input> Parameters> choice of cast items.` Right click on `departmentKey` and choose “Quick Link to Output”:



```
⊞ MQ                              [0..1] _LocalE
⊞ MQTT                            [0..1] _LocalE
⊟ REST                            [0..1] _LocalE
  ⊟ Input                         [0..1] _RESTIr
      Method                      [0..1] string
      Operation                   [0..1] string
      Path                        [0..1] string
      URI                         [0..1] string
    ⊟ Parameters                  [0..1] <Anony
      ⊟ choice of cast items      [0..*]
          any                     [1..1]
          departmentKey           [1..1] string
  ⊞ Response                      [0..1] _RESTResponseType
```

Choose Data : DepartmentResponse from the list of options:

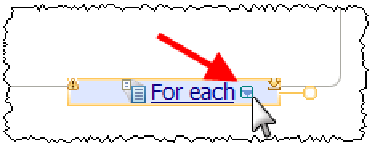


```
⊞ MQTT                            [0..1] _LocalEnvironmentMQTTType
⊟ REST                            [0..1] _LocalEnvironmentRESTType
  ⊟ Input                         [0..1] _RESTInputType
      Method                      [0..1] string
      Operation                   [0..1] string
      Path                        [0..1] string
      URI                         [0..1] string
    ⊟ Parameters                  [0..1] <Anonymous>
      ⊟ choice of cast items      [0..*]
          any                     [1..1]
          departmentKey           [1..1] string
  ⊞ Response                      [0..1] _RESTResponseType
```

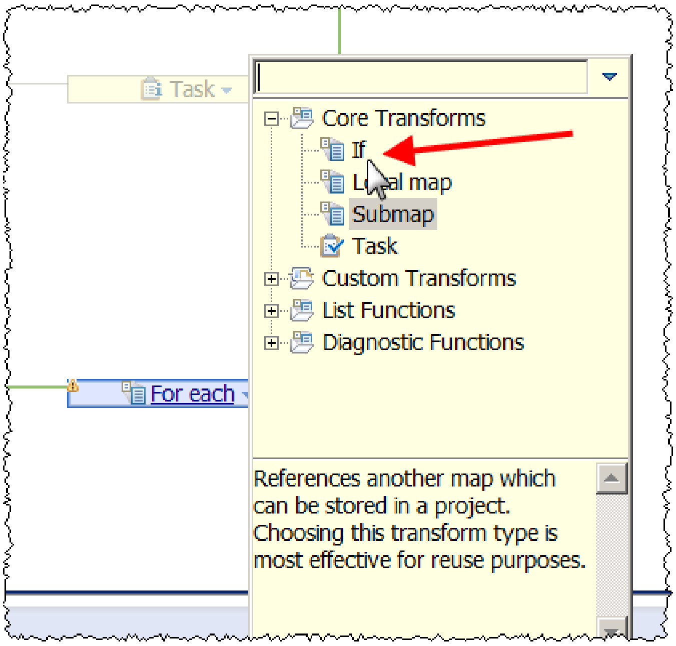
Output Objects
Message Assembly : JSON
Properties : PropertiesType [0..1]
JSON : JSONMsgType [1..1]
Padding : string [0..1]
Data : anyType [1..1]
Data : DepartmentResponse [1..1]
Select output object to complete transformation

A “For each” transform will be created and departmentKey (in the input message assembly) will be connected to DepartmentResponse (in the output message assembly).

_4. Click the small triangle to the right of the word “For each”

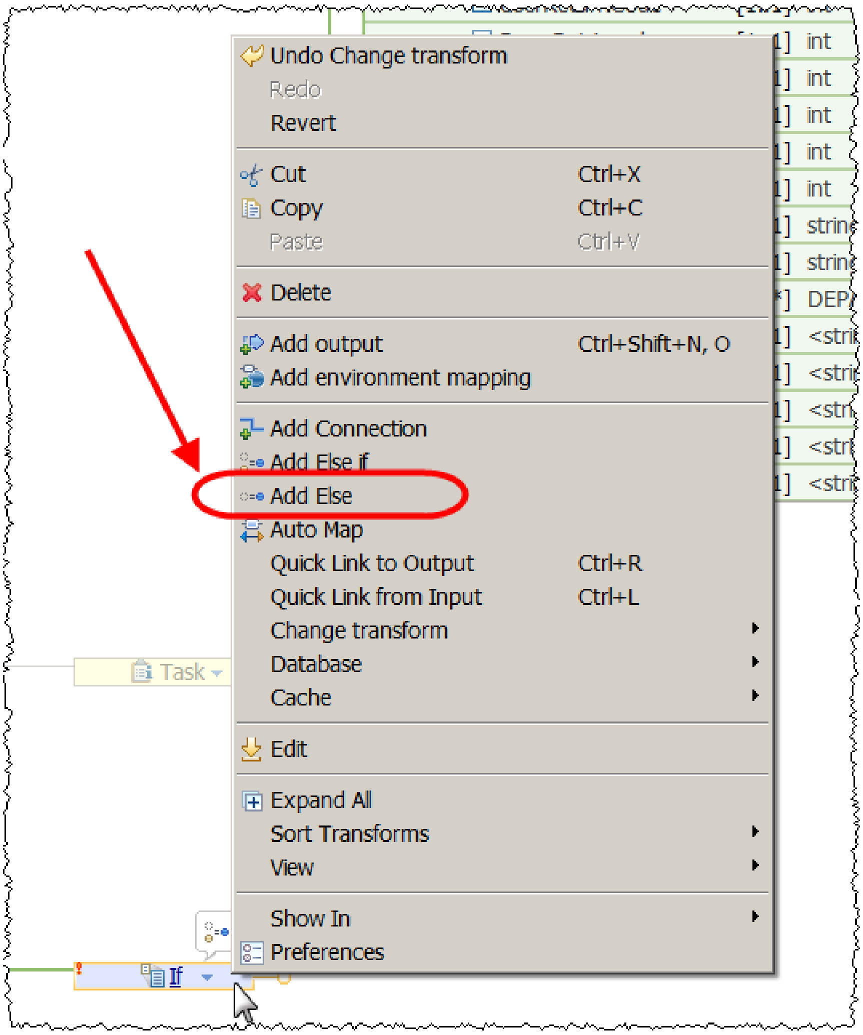


_5. Change the `For each` to `If` from the list of Core Transforms:

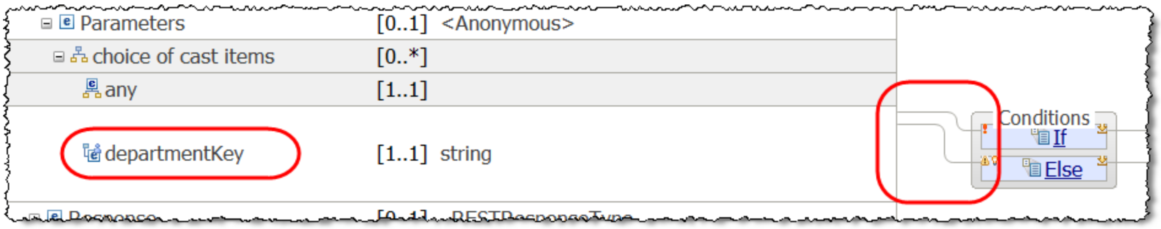


References another map which can be stored in a project. Choosing this transform type is most effective for reuse purposes.

_6. Right click on the background of the If transform and select “Add Else”:



_7. Connect `departmentKey` in the Input Message Assembly to the `Else` tranform, *this makes the input parameter available to the logic in the next level of the map behind the two transforms (you will configure later)*:

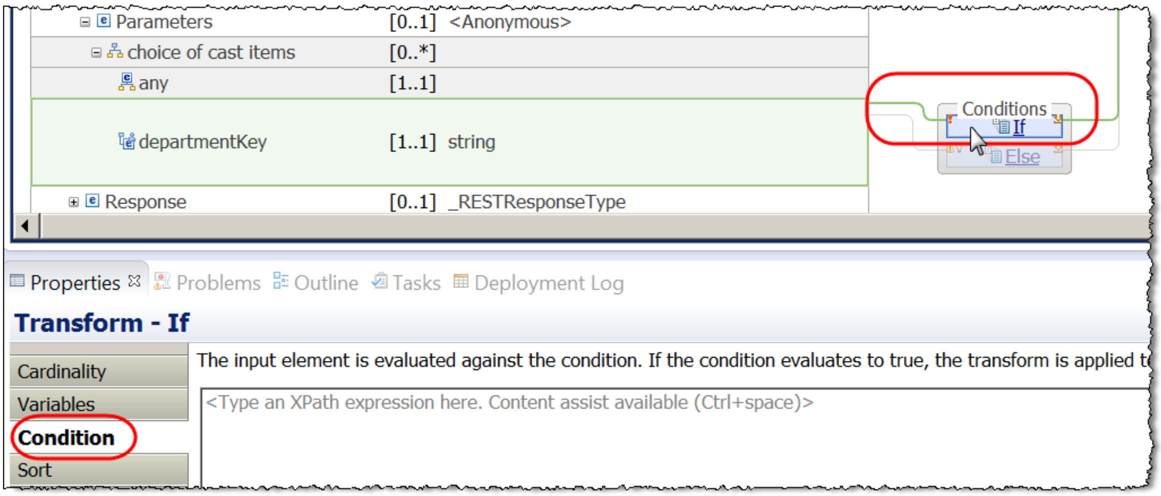
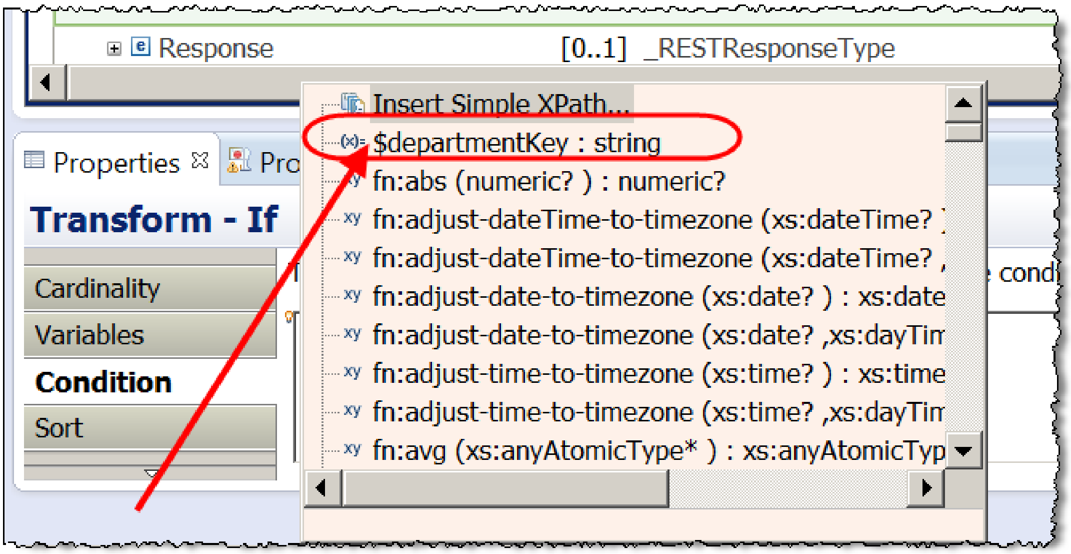


| | | |
|---|---|---|
| Parameters | [0..1] | <Anonymous> |
| choice of cast items | [0..*] | |
| any | [1..1] | |
| departmentKey | [1..1] | string |
| Response | [0..1] | RESTResponseType |

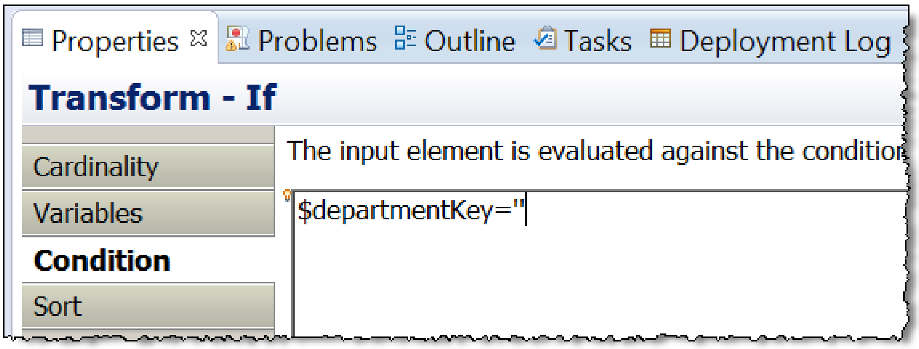
_8. Right click on the background of the **Else** transform and choose "**Quick Link to Output**", from the list of elements choose Data : DepartmentResponse. The Else Transform will then be connected to Data element (of type DepartmentResponse) in the Output Message Assembly.

### 4.1.1 Configure the If transform condition

In this section you will configure the condition for the nested transform (associated with the If transform) to be executed. The nested transform will be executed if the (input Message Assembly) parameter `departmentKey` does not contain data.

| | |
|---|---|
| __1. | Click on the blue background of the `If` transform and select `Condition` in the Properties tab:<br><br> Parameters [0..1] <Anonymous><br>choice of cast items [0..*]<br>any [1..1]<br>departmentKey [1..1] string<br>Response [0..1] _RESTResponseType<br>Conditions<br>If<br>Else<br>Properties<br>Problems<br>Outline<br>Tasks<br>Deployment Log<br>Transform - If<br>Cardinality<br>Variables<br>Condition<br>Sort<br>The input element is evaluated against the condition. If the condition evaluates to true, the transform is applied t<br><Type an XPath expression here. Content assist available (Ctrl+space)> |
| __2. | Select "Type an XPath Expression here" to remove the text.<br><br>Press CTRL and the Space bar and select (*double click*) "$departmentKey" from the list provided by "content assist":<br><br> Response [0..1] _RESTResponseType<br>Properties<br>Pro<br>Transform - If<br>Cardinality<br>Variables<br>Condition<br>Sort<br>cond<br>Insert Simple XPath...<br>$departmentKey : string<br>fn:abs (numeric? ) : numeric?<br>fn:adjust-dateTime-to-timezone (xs:dateTime? )<br>fn:adjust-dateTime-to-timezone (xs:dateTime? ,<br>fn:adjust-date-to-timezone (xs:date? ) : xs:date<br>fn:adjust-date-to-timezone (xs:date? ,xs:dayTim<br>fn:adjust-time-to-timezone (xs:time? ) : xs:time<br>fn:adjust-time-to-timezone (xs:time? ,xs:dayTim<br>fn:avg (xs:anyAtomicType* ) : xs:anyAtomicTyp |

_3. After the $departmentKey type: `=''` (*an equal sign and two single quotes*):



The input element is evaluated against the conditio

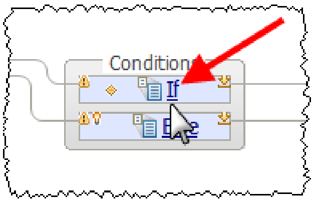
$departmentKey=''

_4. This means that if the (input Message Assembly) parameter `departmentKey` does not contain data, the nested map defined in the `If` transform will be executed.

## 4.1.2 Configure the transform associated with the If Condition

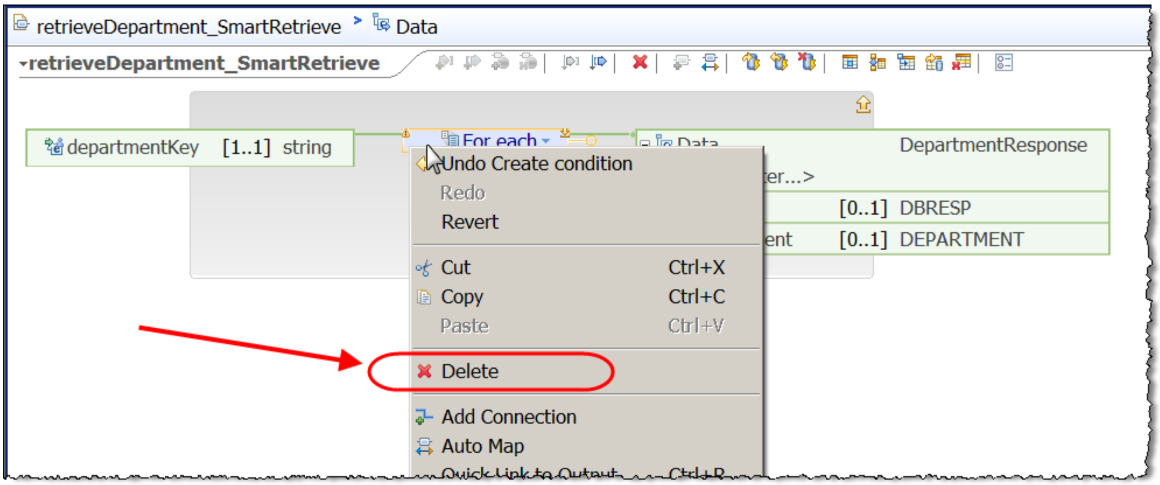
In this section you will configure (in a nested transform) the logic for when the input parameter `departmentKey` does not contain data.

_1. Click the word `If` in the `If` Condition box :

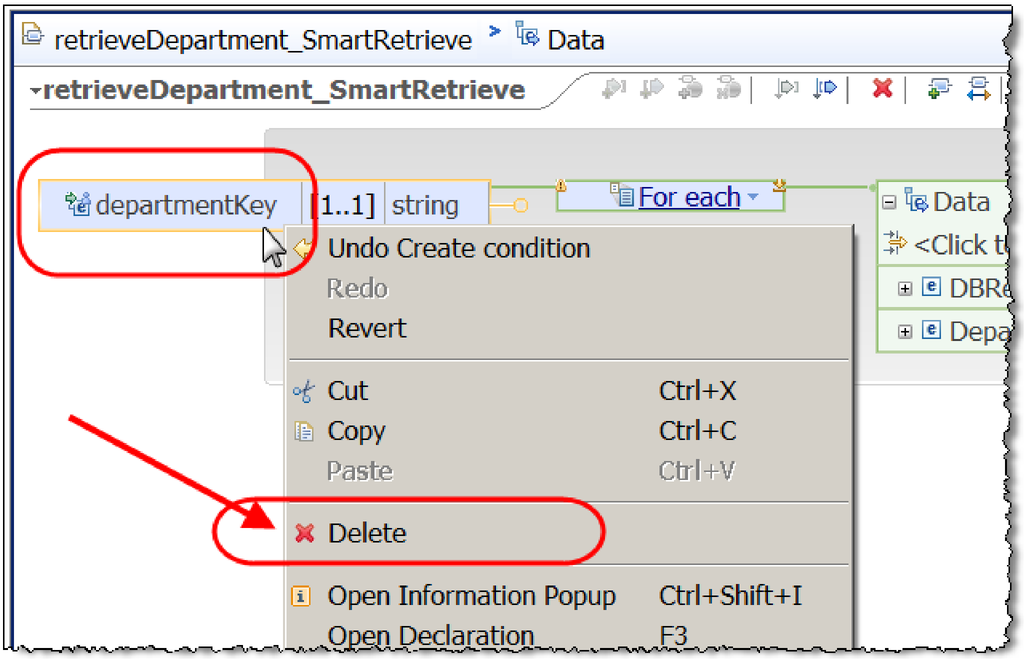


This will navigate the map editor to a nested transform.

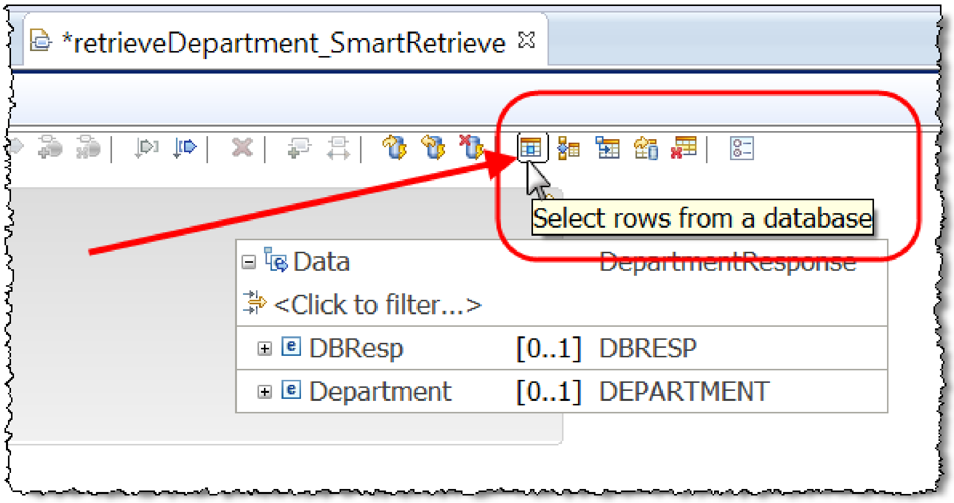
_2. In the nested transform, right click on blue background of the `For each` transform and delete it:



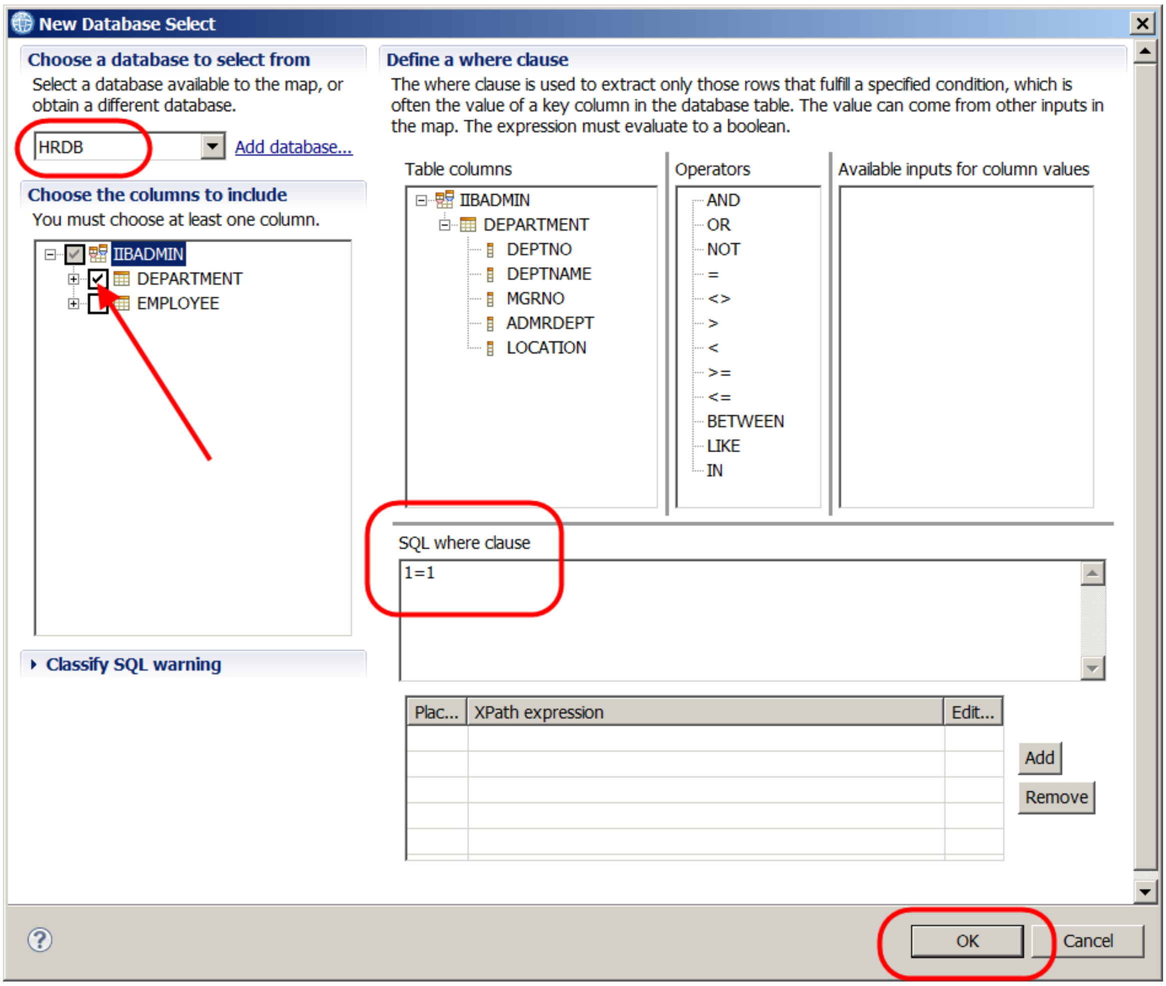
_3. The input Message Assembly parameter `departmentKey` is null in this nested map so right click on the `departmentKey` input parameter and delete this parameter.



_4. You will now add the database call to select all rows from the DEPARTMENT database, click "Select rows from a database":


*retrieveDepartment_SmartRetrieve
Select rows from a database
Data
DepartmentResponse
<Click to filter...>
DBResp [0..1] DBRESP
Department [0..1] DEPARTMENT

__5. In the “New Database Select” window, select `DEPARTMENT` table from the `HRDB` database. Leave the SQL where clause to the default `1=1`:



New Database Select

Choose a database to select from

Select a database available to the map, or obtain a different database.

HRDB ▾ Add database...

Choose the columns to include

You must choose at least one column.

☑ IIBADMIN
☑ DEPARTMENT
□ EMPLOYEE

Classify SQL warning

Define a where clause

The where clause is used to extract only those rows that fulfill a specified condition, which is often the value of a key column in the database table. The value can come from other inputs in the map. The expression must evaluate to a boolean.

| Table columns | Operators | Available inputs for column values |
|---|---|---|
| IIBADMIN<br>DEPARTMENT<br>DEPTNO<br>DEPTNAME<br>MGRNO<br>ADMRDEPT<br>LOCATION | AND<br>OR<br>NOT<br>=<br><><br>><br><<br>>=<br><=<br>BETWEEN<br>LIKE<br>IN | |

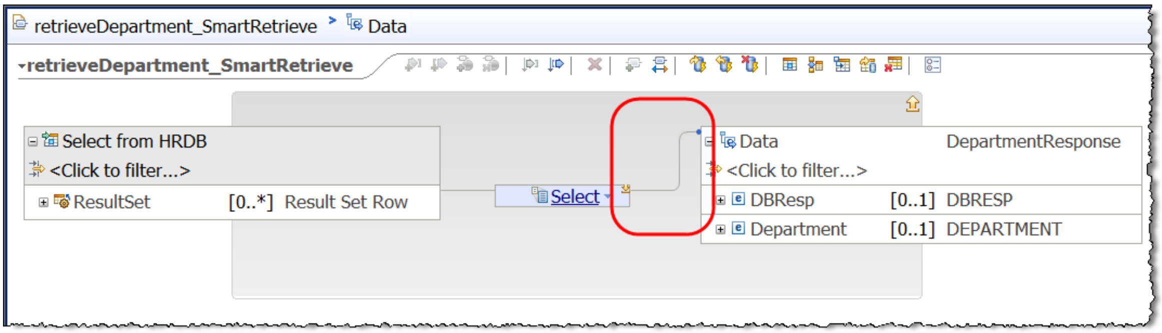
SQL where clause

```
1=1
```

| Plac... | XPath expression | Edit... |
|---|---|---|

This will select all records from the department table.

__6. Connect the Select transform to the Data element (DepartmentResponse) on the Output Message Assembly:

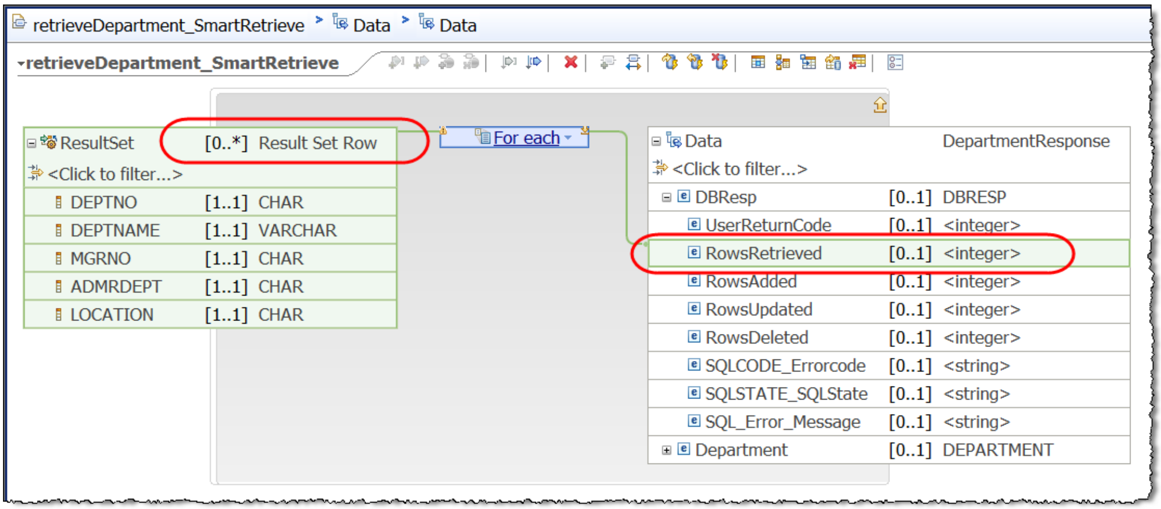


retrieveDepartment_SmartRetrieve > Data
retrieveDepartment_SmartRetrieve
Select from HRDB
<Click to filter...>
ResultSet [0..*] Result Set Row
Select
Data DepartmentResponse
<Click to filter...>
DBResp [0..1] DBRESP
Department [0..1] DEPARTMENT

__7. Click the word Select in the Select Transform. This will allow you to create the individual elements mappings for the Select transform.

_8. Expand `DBResp` on the Output Message Assembly.

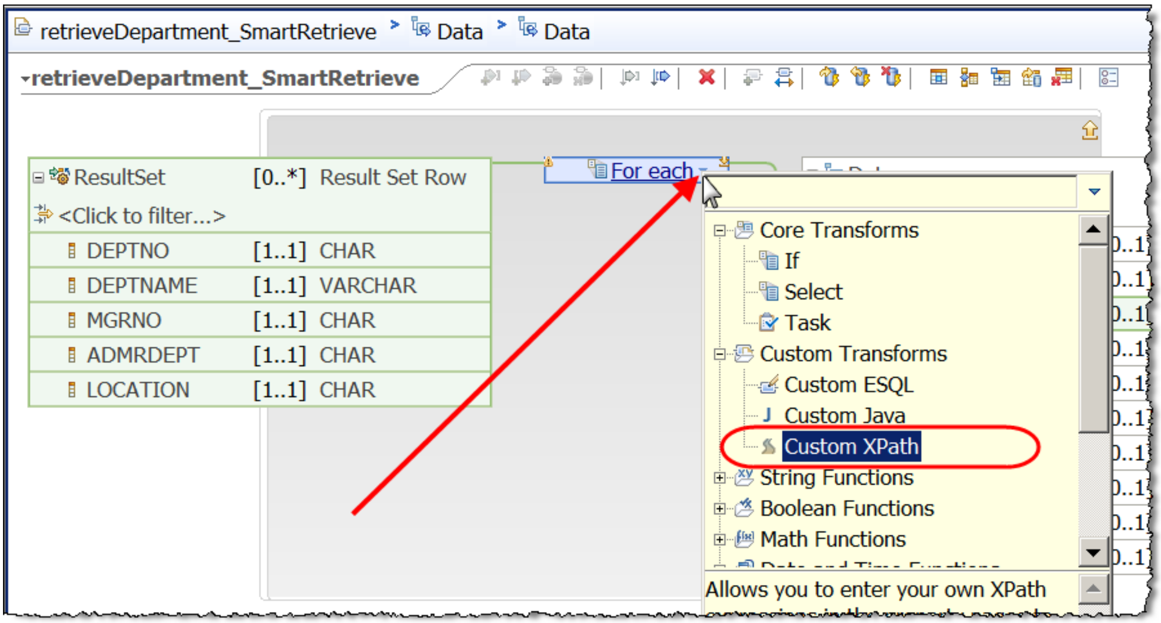
Connect `Result Set Row` on the input Message Assembly to the `RowsRetrieved` element in `DBResp`. This will create a `For each` transform.


retrieveDepartment_SmartRetrieve > Data > Data
retrieveDepartment_SmartRetrieve
For each

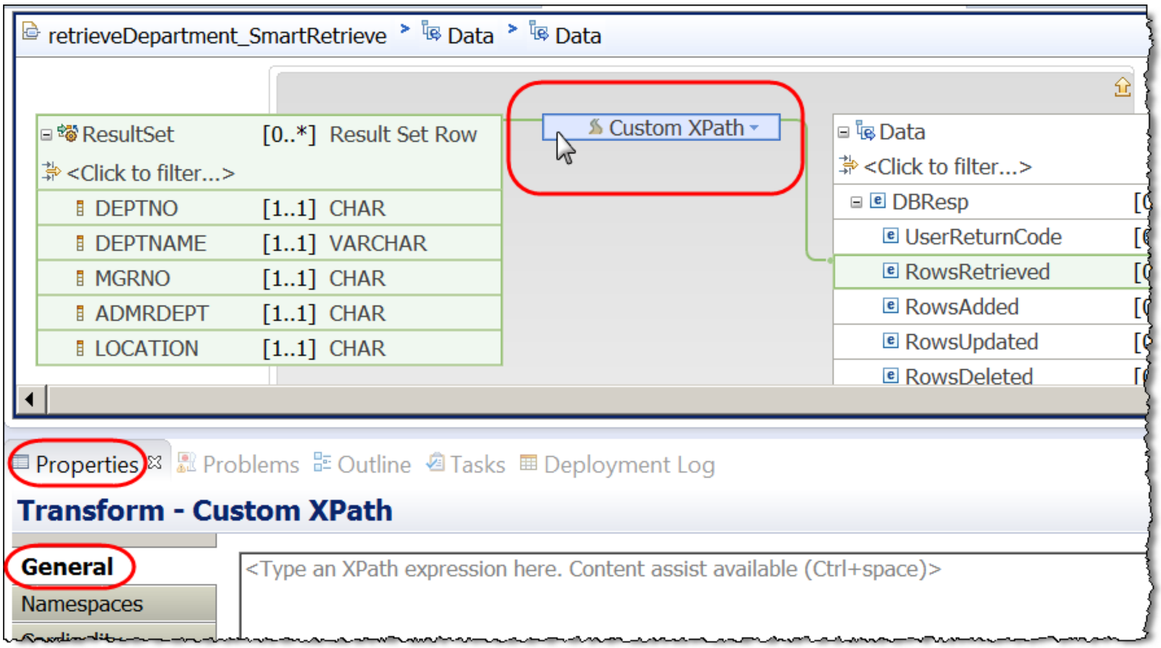
| ResultSet | [0..*] | Result Set Row |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [1..1] | CHAR |
| DEPTNAME | [1..1] | VARCHAR |
| MGRNO | [1..1] | CHAR |
| ADMRDEPT | [1..1] | CHAR |
| LOCATION | [1..1] | CHAR |

| Data | | DepartmentResponse |
|---|---|---|
| <Click to filter...> | | |
| DBResp | [0..1] | DBRESP |
| UserReturnCode | [0..1] | <integer> |
| RowsRetrieved | [0..1] | <integer> |
| RowsAdded | [0..1] | <integer> |
| RowsUpdated | [0..1] | <integer> |
| RowsDeleted | [0..1] | <integer> |
| SQLCODE_Errorcode | [0..1] | <string> |
| SQLSTATE_SQLState | [0..1] | <string> |
| SQL_Error_Message | [0..1] | <string> |
| Department | [0..1] | DEPARTMENT |

_9. Click the blue triangle in this `For each` transform and change it to a `Custom XPath` transform:



| ResultSet | [0..*] | Result Set Row |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [1..1] | CHAR |
| DEPTNAME | [1..1] | VARCHAR |
| MGRNO | [1..1] | CHAR |
| ADMRDEPT | [1..1] | CHAR |
| LOCATION | [1..1] | CHAR |

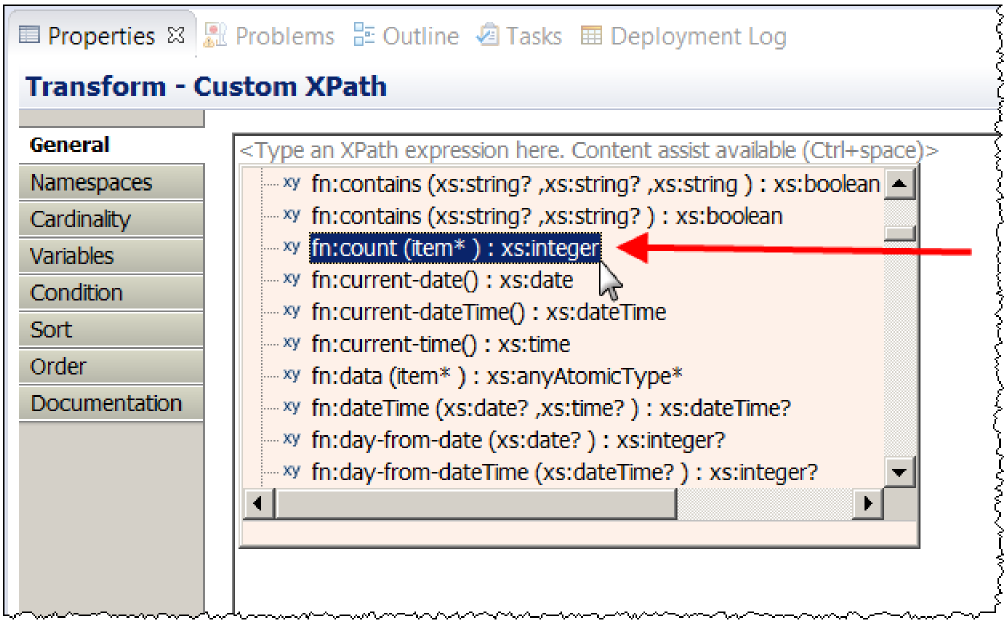
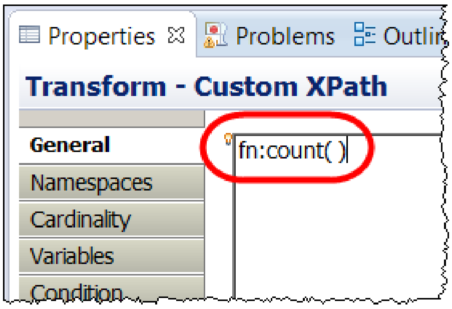
| | |
|---|---|
| _10 | Click the background of the `Custom XPath` box and select `General` in the Properties tab:<br> |

retrieveDepartment_SmartRetrieve > Data > Data

| ResultSet | [0..*] | Result Set Row |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [1..1] | CHAR |
| DEPTNAME | [1..1] | VARCHAR |
| MGRNO | [1..1] | CHAR |
| ADMRDEPT | [1..1] | CHAR |
| LOCATION | [1..1] | CHAR |

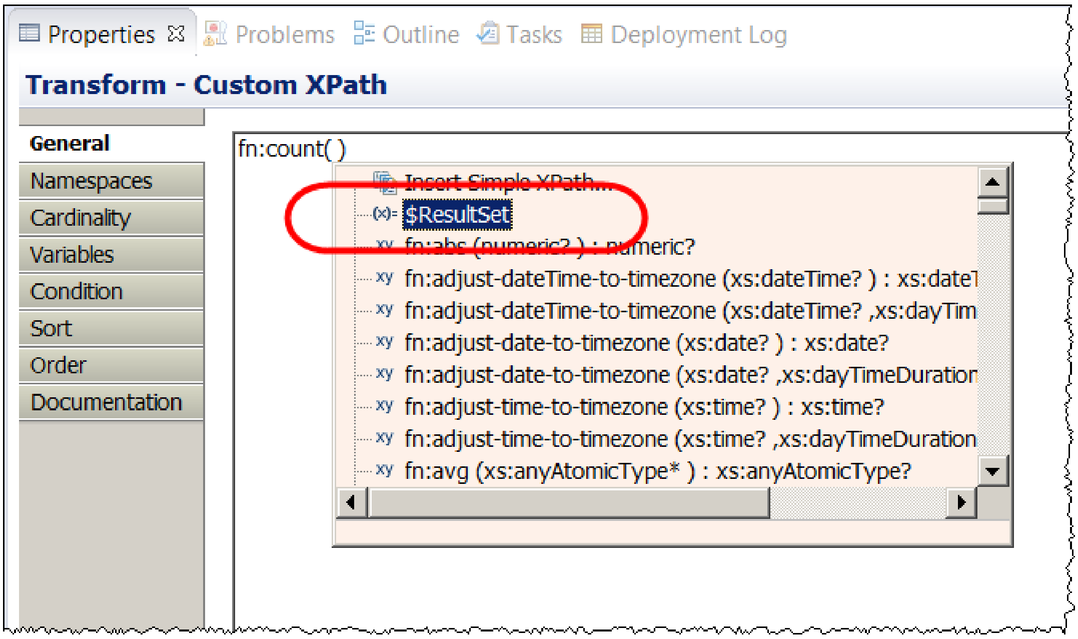
Transform - Custom XPath

<Type an XPath expression here. Content assist available (Ctrl+space)>

| | |
|---|---|
| __11 | Click the box that begins with the expression “`Type an XPath expression here.`”<br>Press the <Ctrl> key and the <Space bar> keys simultaneously to initate the Content Assist feature.<br>Add `fn:count` to the XPath Transform (Double click on `fn:count(item*)` from the list to add it to the box:<br><br>Properties Problems Outline Tasks Deployment Log<br>Transform - Custom XPath<br>General<br>Namespaces<br>Cardinality<br>Variables<br>Condition<br>Sort<br>Order<br>Documentation<br><Type an XPath expression here. Content assist available (Ctrl+space)><br>fn:contains (xs:string? ,xs:string? ,xs:string ) : xs:boolean<br>fn:contains (xs:string? ,xs:string? ) : xs:boolean<br>fn:count (item* ) : xs:integer<br>fn:current-date() : xs:date<br>fn:current-dateTime() : xs:dateTime<br>fn:current-time() : xs:time<br>fn:data (item* ) : xs:anyAtomicType*<br>fn:dateTime (xs:date? ,xs:time? ) : xs:dateTime?<br>fn:day-from-date (xs:date? ) : xs:integer?<br>fn:day-from-dateTime (xs:dateTime? ) : xs:integer?<br>`fn:count()` will appear in the box:<br><br>Properties Problems Outlin<br>Transform - Custom XPath<br>General<br>Namespaces<br>Cardinality<br>Variables<br>Condition<br>fn:count( ) |

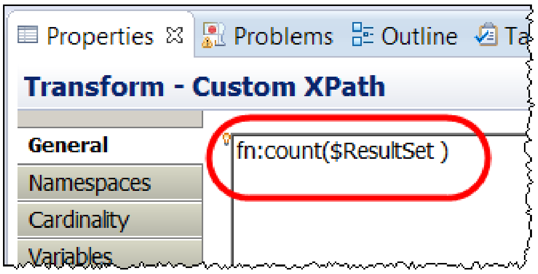
_12 Place the cursor inbetween the braces and press the Ctrl and space bar simulaneously again.

Select (double click) `$ResultSet` from the list of options:


Properties
Problems
Outline
Tasks
Deployment Log
Transform - Custom XPath
General
Namespaces
Cardinality
Variables
Condition
Sort
Order
Documentation
Insert Simple XPath...
$ResultSet
fn:abs (numeric? ) : numeric?
fn:adjust-dateTime-to-timezone (xs:dateTime? ) : xs:dateT
fn:adjust-dateTime-to-timezone (xs:dateTime? ,xs:dayTim
fn:adjust-date-to-timezone (xs:date? ) : xs:date?
fn:adjust-date-to-timezone (xs:date? ,xs:dayTimeDuration
fn:adjust-time-to-timezone (xs:time? ) : xs:time?
fn:adjust-time-to-timezone (xs:time? ,xs:dayTimeDuration
fn:avg (xs:anyAtomicType* ) : xs:anyAtomicType?

```
fn:count( )
```

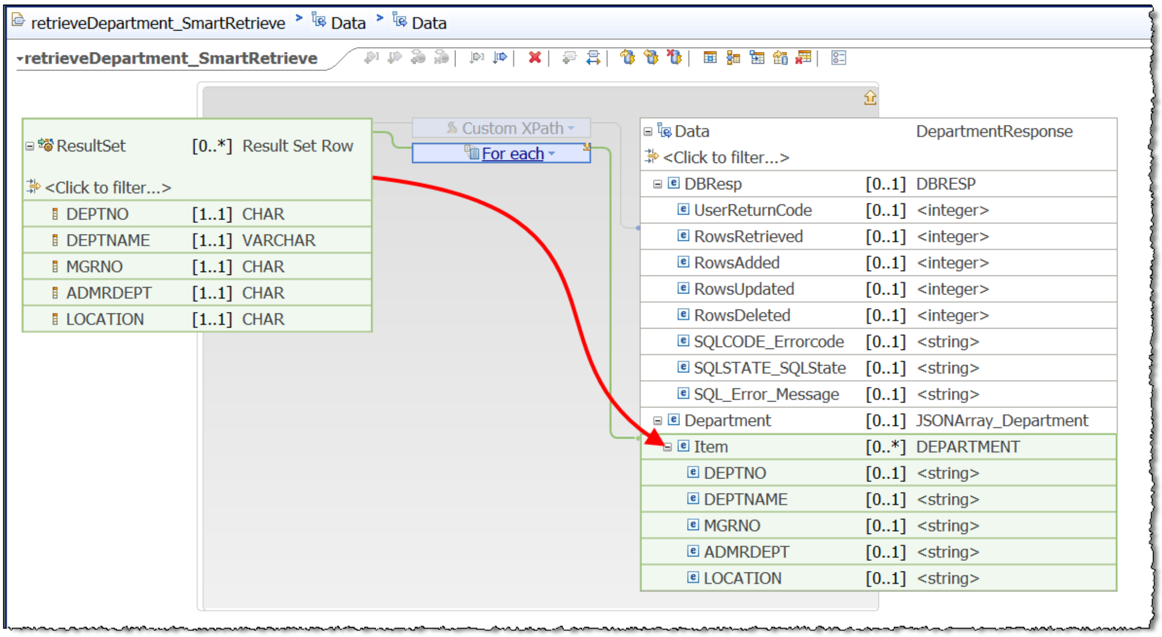
_13 The General property for the Custom XPath transform will look like this when complete:



```
fn:count($ResultSet )
```

This will count the results returned from the select statement and use the number to set the value for RowsRetrieved in DBResp.

_14 Drag the ResultSet Row to the Item element in Department (array) in the Output Message Assembly. This will create a `For each` transform (since there will be more than one entry returned from the database select statement you added earlier):

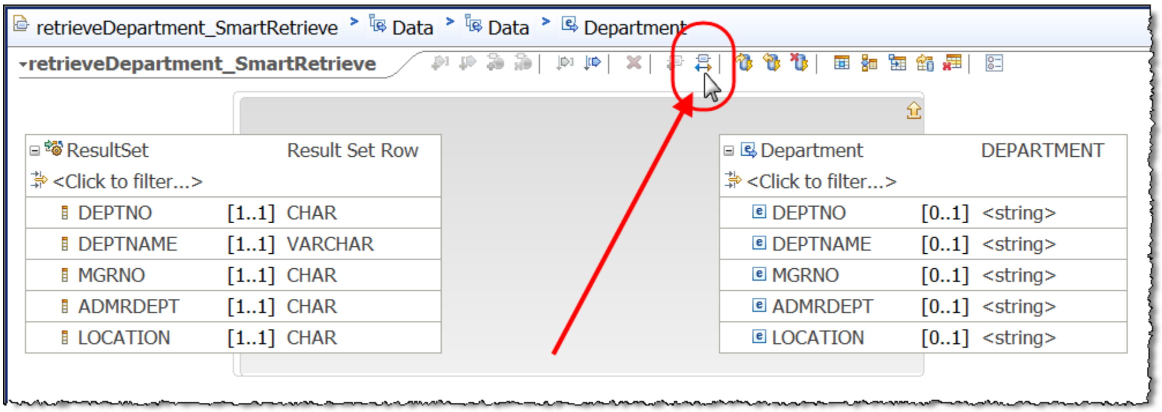

retrieveDepartment_SmartRetrieve > Data > Data
retrieveDepartment_SmartRetrieve
Custom XPath
For each

| ResultSet | [0..*] | Result Set Row |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [1..1] | CHAR |
| DEPTNAME | [1..1] | VARCHAR |
| MGRNO | [1..1] | CHAR |
| ADMRDEPT | [1..1] | CHAR |
| LOCATION | [1..1] | CHAR |

| Data | | DepartmentResponse |
|---|---|---|
| <Click to filter...> | | |
| DBResp | [0..1] | DBRESP |
| UserReturnCode | [0..1] | <integer> |
| RowsRetrieved | [0..1] | <integer> |
| RowsAdded | [0..1] | <integer> |
| RowsUpdated | [0..1] | <integer> |
| RowsDeleted | [0..1] | <integer> |
| SQLCODE_Errorcode | [0..1] | <string> |
| SQLSTATE_SQLState | [0..1] | <string> |
| SQL_Error_Message | [0..1] | <string> |
| Department | [0..1] | JSONArray_Department |
| Item | [0..*] | DEPARTMENT |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

_15 Click the word `For each` in the For each transform box, this will take you to a nested transform that will be executed for every row returned from the select statement.

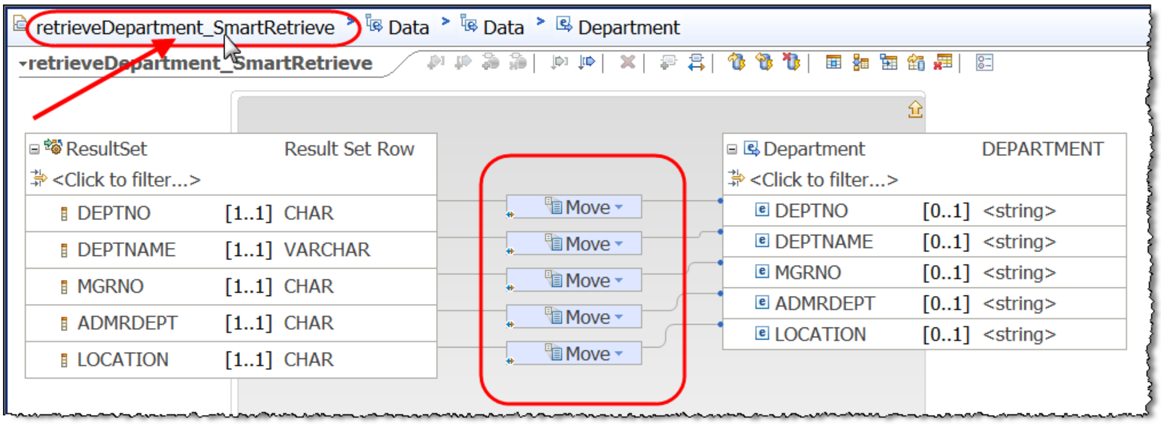
_16 In this nested transform select the Auto map Input to Output icon. Accept the defaults and press Finish:


retrieveDepartment_SmartRetrieve > Data > Data > Department
retrieveDepartment_SmartRetrieve

| ResultSet | | Result Set Row |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [1..1] | CHAR |
| DEPTNAME | [1..1] | VARCHAR |
| MGRNO | [1..1] | CHAR |
| ADMRDEPT | [1..1] | CHAR |
| LOCATION | [1..1] | CHAR |

| Department | | DEPARTMENT |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

_17 The resulting nested transform will look this when complete (every row returned by the select statement will be copied to the Department element):


retrieveDepartment_SmartRetrieve > Data > Data > Department
retrieveDepartment_SmartRetrieve
Move
Move
Move
Move
Move

| ResultSet | | Result Set Row |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [1..1] | CHAR |
| DEPTNAME | [1..1] | VARCHAR |
| MGRNO | [1..1] | CHAR |
| ADMRDEPT | [1..1] | CHAR |
| LOCATION | [1..1] | CHAR |

| Department | | DEPARTMENT |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

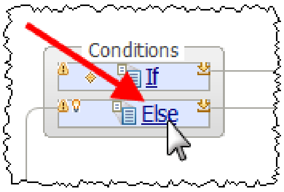
Click `retrieveDepartment_SmartRetrieve` to navigate back to the first level in the map.

_18 Save the map changes (Ctrl s), stay in the editor.

## 4.1.3 Configure the nested transform associated with the `Else` condition

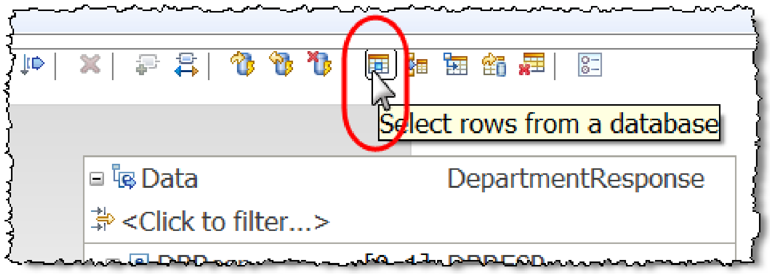
In this section you will configure (in a nested map) the logic for when the input parameter `departmentKey` does contain data.

_1. Click the word `Else` in the `Else` Condition box :



This will navigate the map editor to a nested transform.

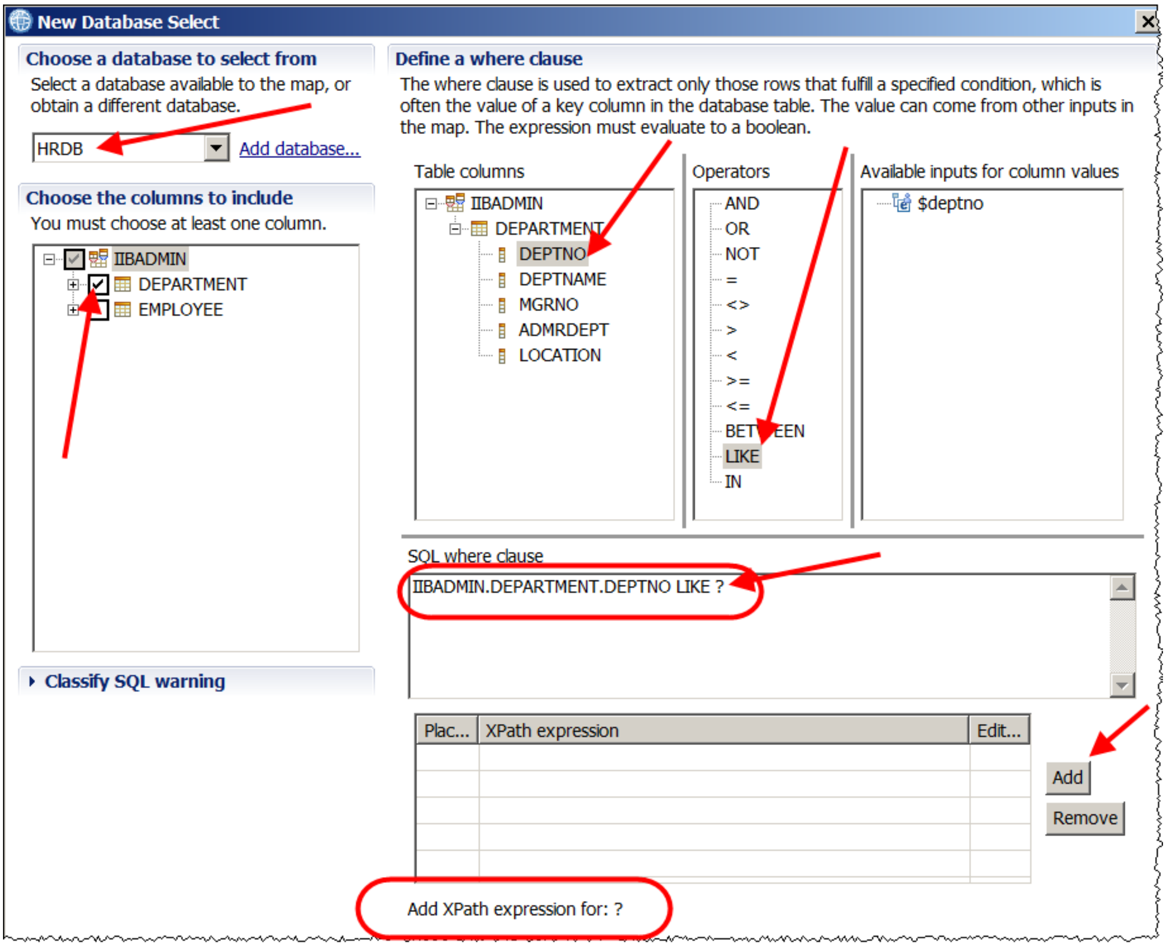
_2. As with the `If` transform, Click the “Select rows from a database” icon:



| Data | DepartmentResponse |
|---|---|
| <Click to filter...> | |

_3. In the **New Database Select** window

a) Select HRDB and all fields from the DEPARTMENT table.
b) Delete 1=1 from the SQL where clause
c) Double click on DEPTNO the table column
d) Double click LIKE in the Operators column
e) Add a space followed by a question mark (?)



New Database Select

Choose a database to select from

Select a database available to the map, or obtain a different database.

HRDB

Choose the columns to include

You must choose at least one column.

Define a where clause

The where clause is used to extract only those rows that fulfill a specified condition, which is often the value of a key column in the database table. The value can come from other inputs in the map. The expression must evaluate to a boolean.

SQL where clause: IIBADMIN.DEPARTMENT.DEPTNO LIKE ?

Add XPath expression for: ?

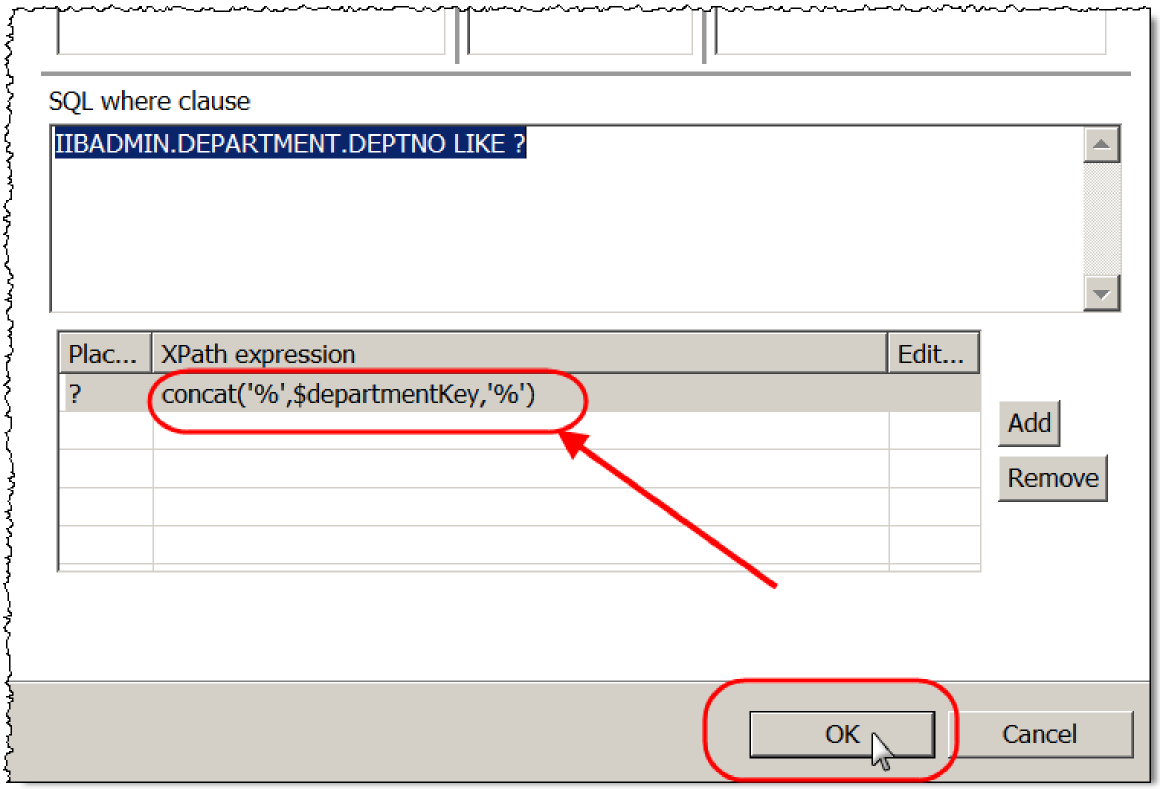
Note the Instruction to “Add XPath expression for : ?.”

_4. Click the Add button to the right of the XPath Expression table.

In the XPath expression column for ? type:

```
concat('%',$departmentKey,'%')
```

Click OK when complete.



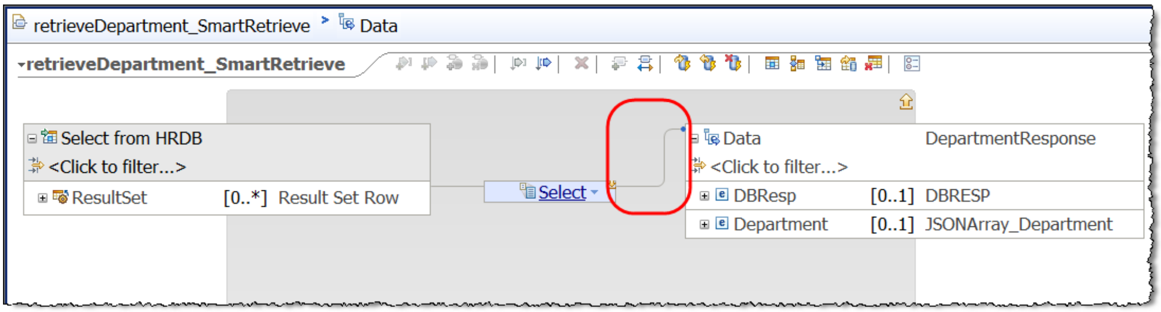
SQL where clause

IIBADMIN.DEPARTMENT.DEPTNO LIKE ?

| Plac... | XPath expression | Edit... |
|---|---|---|
| ? | concat('%',$departmentKey,'%') | |

This will create a Select transform in the nested map.

_5. Connect the Select transform to the Data element with DepartmentResponse in the Output Message Assembly:



retrieveDepartment_SmartRetrieve > Data

retrieveDepartment_SmartRetrieve

| | | |
|---|---|---|
| Select from HRDB | | |
| <Click to filter...> | | |
| ResultSet | [0..*] | Result Set Row |

| | | |
|---|---|---|
| Data | | DepartmentResponse |
| <Click to filter...> | | |
| DBResp | [0..1] | DBRESP |
| Department | [0..1] | JSONArray_Department |

***The following instructions are very similar to those outlined for the If transform in the main lab guide above (please refer to the If transform for detailed screen captures).***

_6. Click the word `Select` in the Select transform.

_7. Expand `DBResp` and connect `Result Set Row` to `RowsRetrieved` in `DBResp`.

_8. Change the `For each` transform into a `Custom XPath` transform.

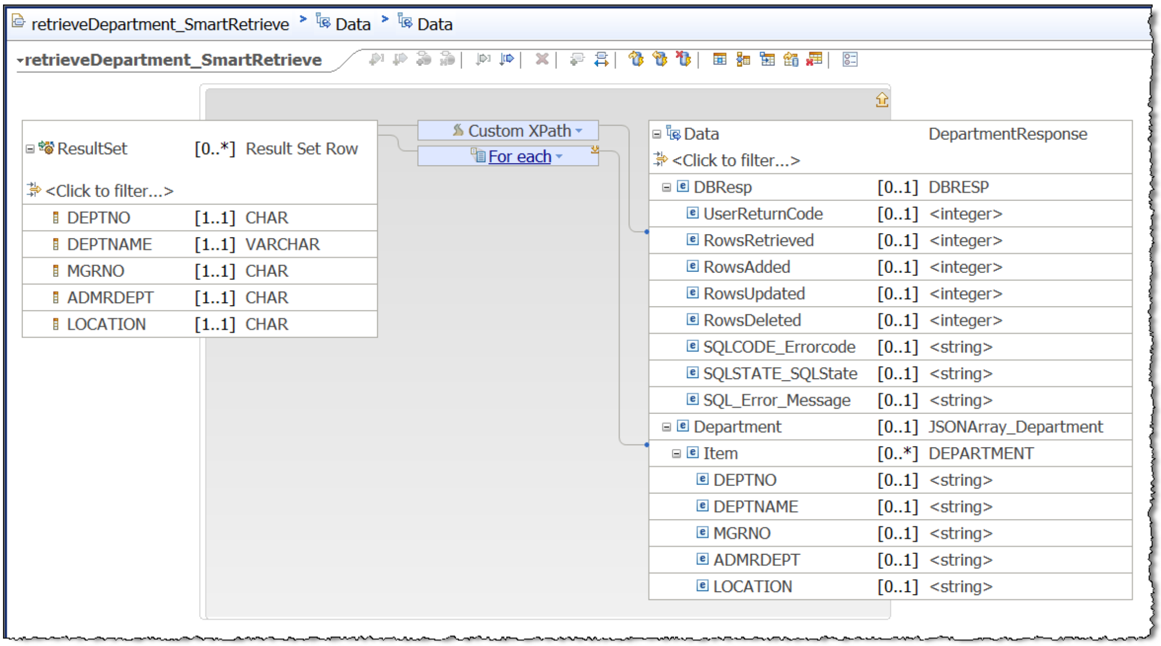
_9. Use Content Assist to set the Custom XPath **General tab** to “`fn:count($ResultSet1)`“

**Note:** using Content assist will give you the correct ResultSet suffix for your particular environment.

_10 Connect `ResultSet` to the element called `Item` in the JSONArray called `Department`(this will create a `For each` transform)

_11 Select the `For each` transform and use Auto map to map the Input and Output elements.

_12 The nested transform will look like this when complete:



retrieveDepartment_SmartRetrieve > Data > Data

retrieveDepartment_SmartRetrieve

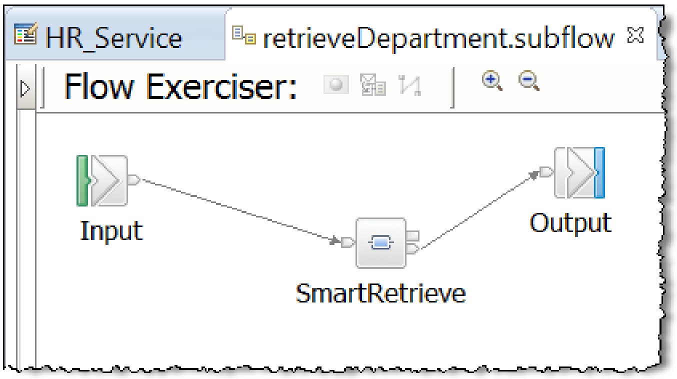
| ResultSet | [0..*] | Result Set Row |
|---|---|---|
| <Click to filter...> | | |
| DEPTNO | [1..1] | CHAR |
| DEPTNAME | [1..1] | VARCHAR |
| MGRNO | [1..1] | CHAR |
| ADMRDEPT | [1..1] | CHAR |
| LOCATION | [1..1] | CHAR |

| Data | | DepartmentResponse |
|---|---|---|
| <Click to filter...> | | |
| DBResp | [0..1] | DBRESP |
| UserReturnCode | [0..1] | <integer> |
| RowsRetrieved | [0..1] | <integer> |
| RowsAdded | [0..1] | <integer> |
| RowsUpdated | [0..1] | <integer> |
| RowsDeleted | [0..1] | <integer> |
| SQLCODE_Errorcode | [0..1] | <string> |
| SQLSTATE_SQLState | [0..1] | <string> |
| SQL_Error_Message | [0..1] | <string> |
| Department | [0..1] | JSONArray_Department |
| Item | [0..*] | DEPARTMENT |
| DEPTNO | [0..1] | <string> |
| DEPTNAME | [0..1] | <string> |
| MGRNO | [0..1] | <string> |
| ADMRDEPT | [0..1] | <string> |
| LOCATION | [0..1] | <string> |

_13 Click `retrieveDepartment_SmartRetrieve` to navigate back to the highest level of the map when complete.

_14 Save the map and close the mapping editor.

_15 In the retrieveDepartment subflow, connect the mapping node to the Input and Output nodes and save the subflow:



_16 Close the subflow editor and the REST API Editor.

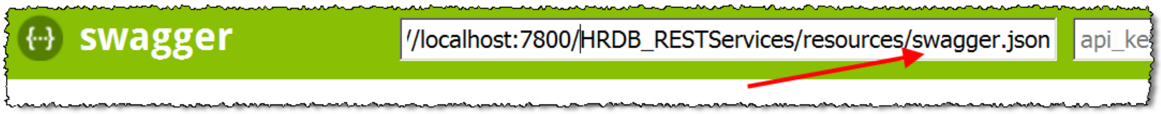
# 5. Test the retrieveDepartment operation

_1. Deploy the HR_Service REST API to the default server in TESTNODE_iibuser

## 5.1 Test using SwaggerUI

In this section you will use SwaggerUI to test your REST API.

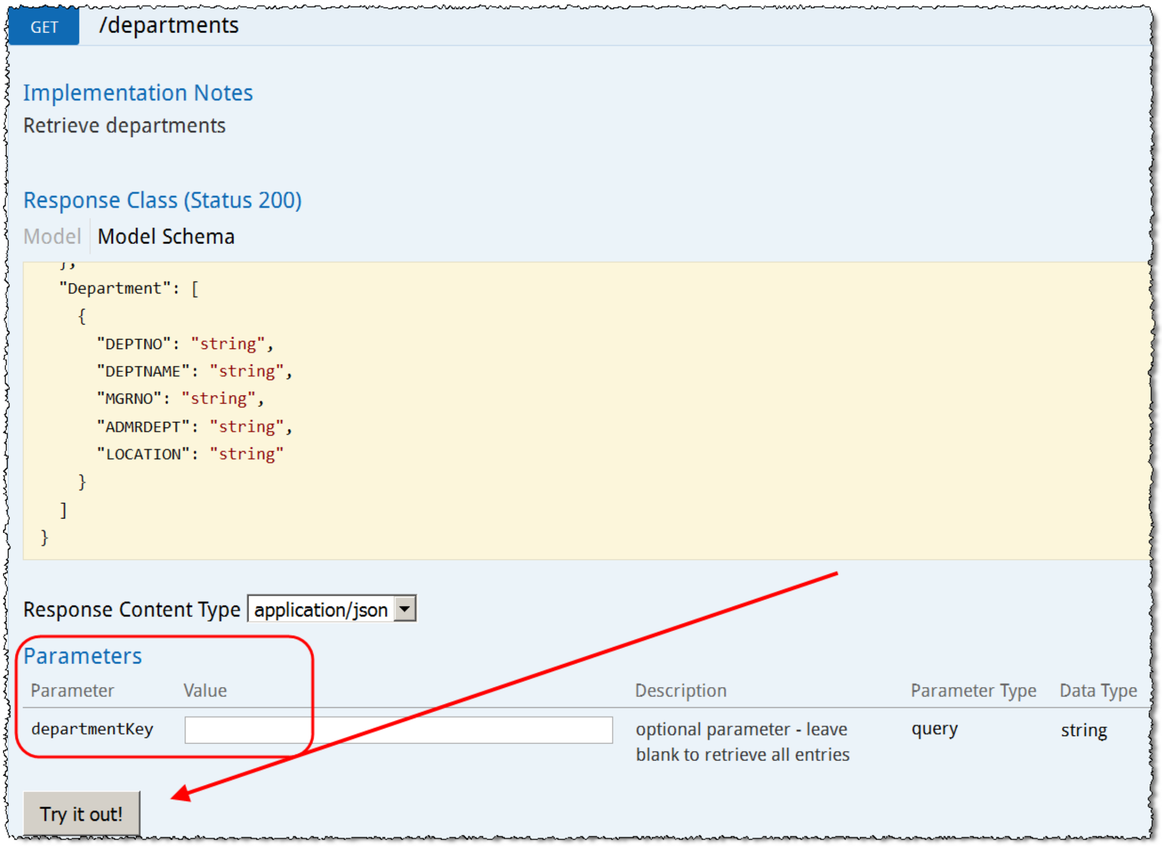
_1. In your SwaggerUI web page, press enter on the (green) URL section at the top of the SwaggerUI testing page:



'/localhost:7800/HRDB_RESTServices/resources/swagger.json

(If you previously closed your swaggerUI refer to the testing section for the insertDepartment operation above on how to open swaggerUI with the correct URL.

_2. Expand the blue GET operation for /departments, note the departmentKey you defined in the REST API and click the “Try it out!” button:



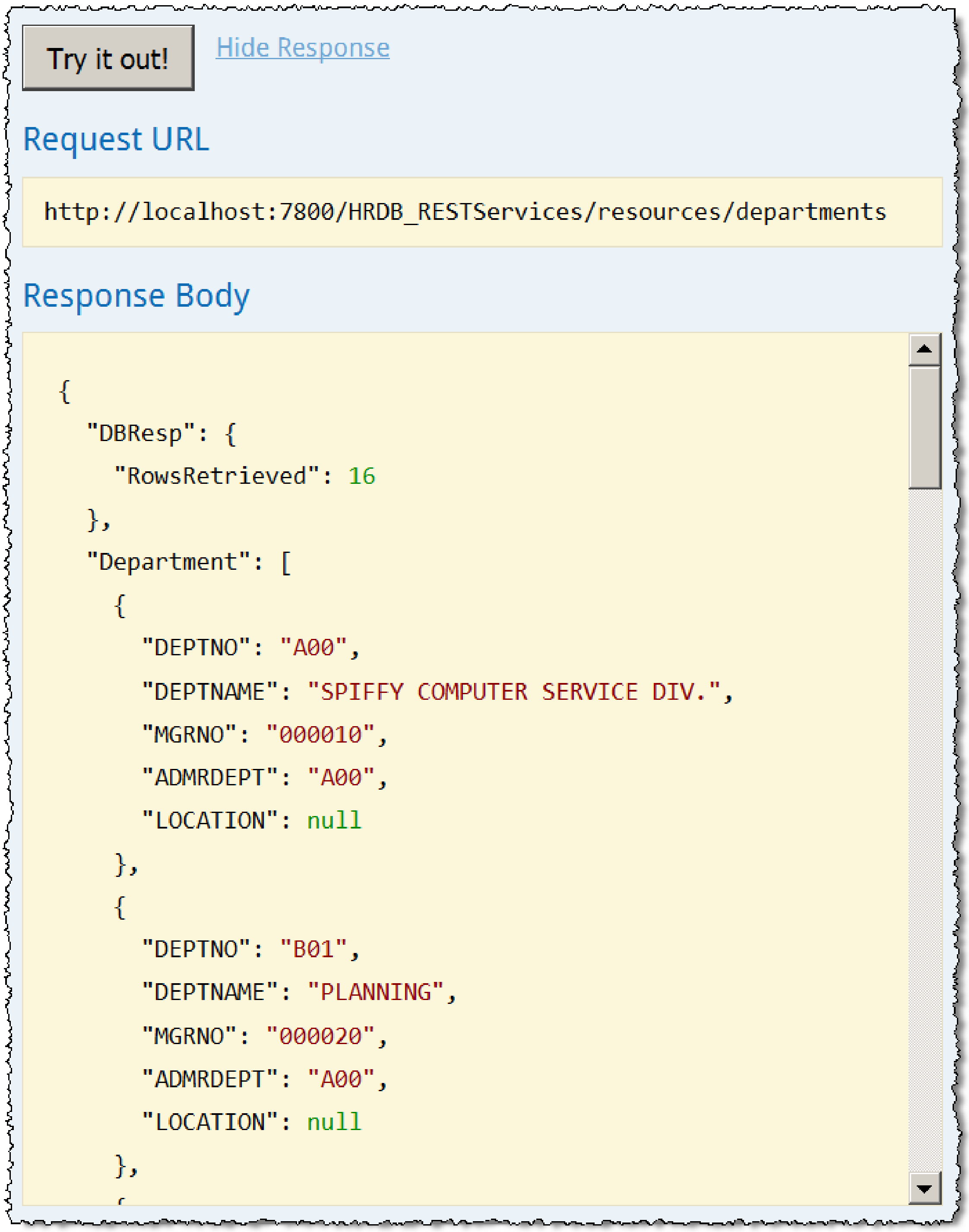
Retrieve departments

```
  }],
  "Department": [
    {
      "DEPTNO": "string",
      "DEPTNAME": "string",
      "MGRNO": "string",
      "ADMRDEPT": "string",
      "LOCATION": "string"
    }
  ]
}
```

Response Content Type application/json

| Parameter | Value | Description | Parameter Type | Data Type |
|---|---|---|---|---|
| departmentKey | | optional parameter - leave blank to retrieve all entries | query | string |

_3. After a few seconds, the Response body (below the Try it out! Button) will be updated with all the entries from the DEPARTMENT table. RowsRetrieved value will show the total number of entries returned. The Department json array will be delimited using square brackets [ ] :



Request URL

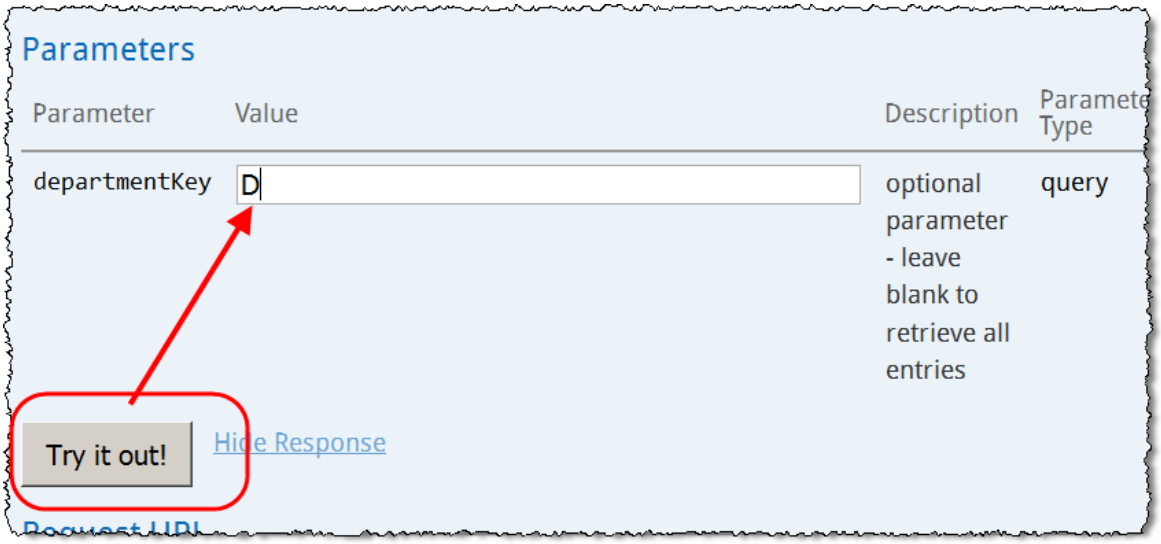
```
http://localhost:7800/HRDB_RESTServices/resources/departments
```

Response Body

```
{
  "DBResp": {
    "RowsRetrieved": 16
  },
  "Department": [
    {
      "DEPTNO": "A00",
      "DEPTNAME": "SPIFFY COMPUTER SERVICE DIV.",
      "MGRNO": "000010",
      "ADMRDEPT": "A00",
      "LOCATION": null
    },
    {
      "DEPTNO": "B01",
      "DEPTNAME": "PLANNING",
      "MGRNO": "000020",
      "ADMRDEPT": "A00",
      "LOCATION": null
    },
    {
```

_4. Note the number of entries returned which begin with a DEPTNO of “D”

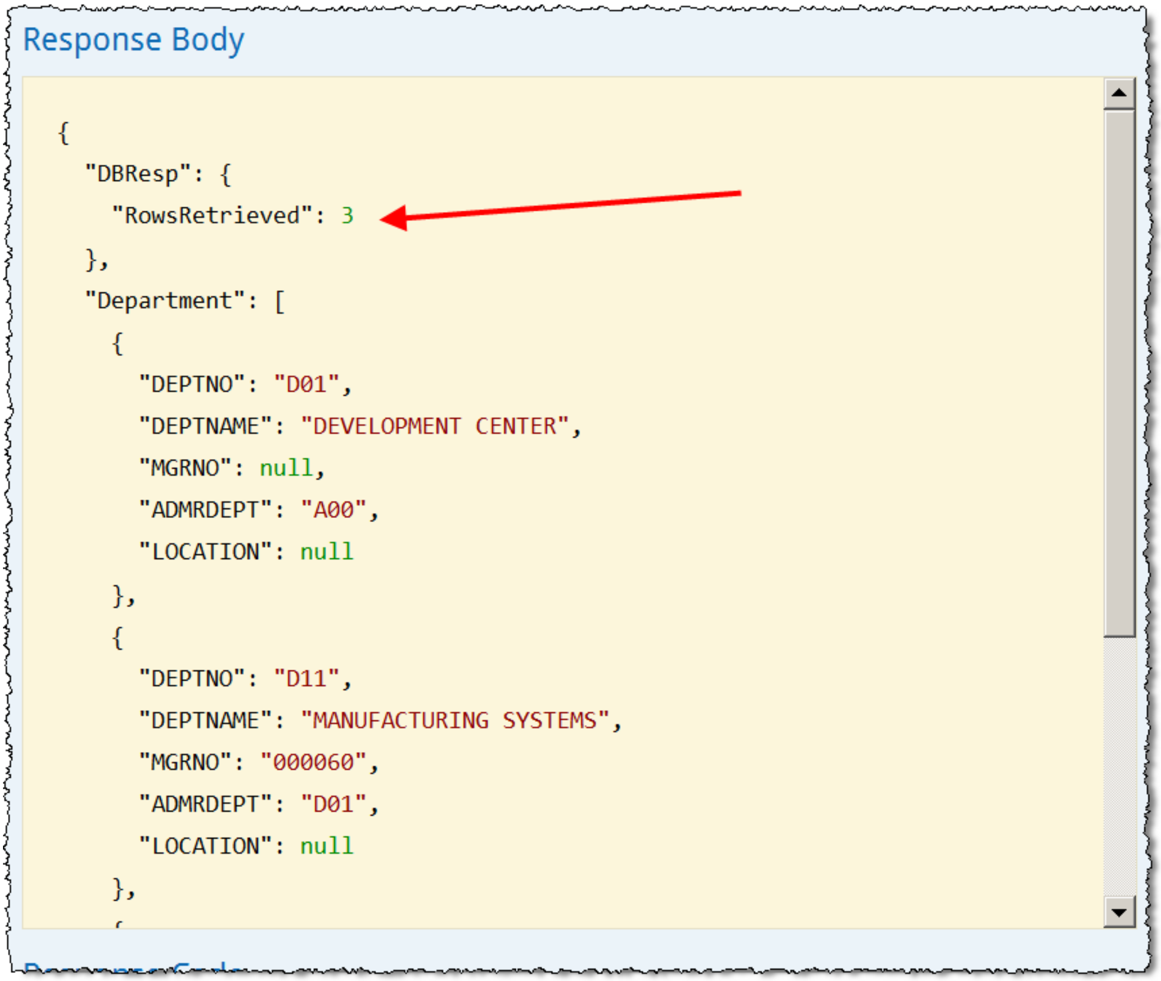
_5. Next you will test the “like” function that you configured in the map. In the departmentKey, enter “D” and click the Try it out! Button:



Parameters

| Parameter | Value | Description | Paramete Type |
|---|---|---|---|
| departmentKey | D | optional parameter - leave blank to retrieve all entries | query |

_6. After a few seconds the service will respond with those entries which have D in the key (in this example there are 3 entries):



Response Body

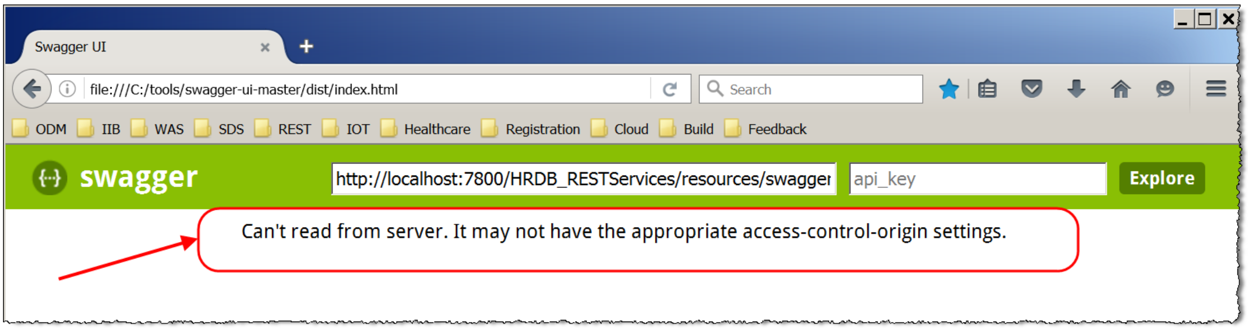
```
{
  "DBResp": {
    "RowsRetrieved": 3
  },
  "Department": [
    {
      "DEPTNO": "D01",
      "DEPTNAME": "DEVELOPMENT CENTER",
      "MGRNO": null,
      "ADMRDEPT": "A00",
      "LOCATION": null
    },
    {
      "DEPTNO": "D11",
      "DEPTNAME": "MANUFACTURING SYSTEMS",
      "MGRNO": "000060",
      "ADMRDEPT": "D01",
      "LOCATION": null
    },
```

_7. Try entering a full key too, you should receive a response containing an exact match on the key you entered.

# 6. Troubleshooting

## 6.1 CORS Testing Issue

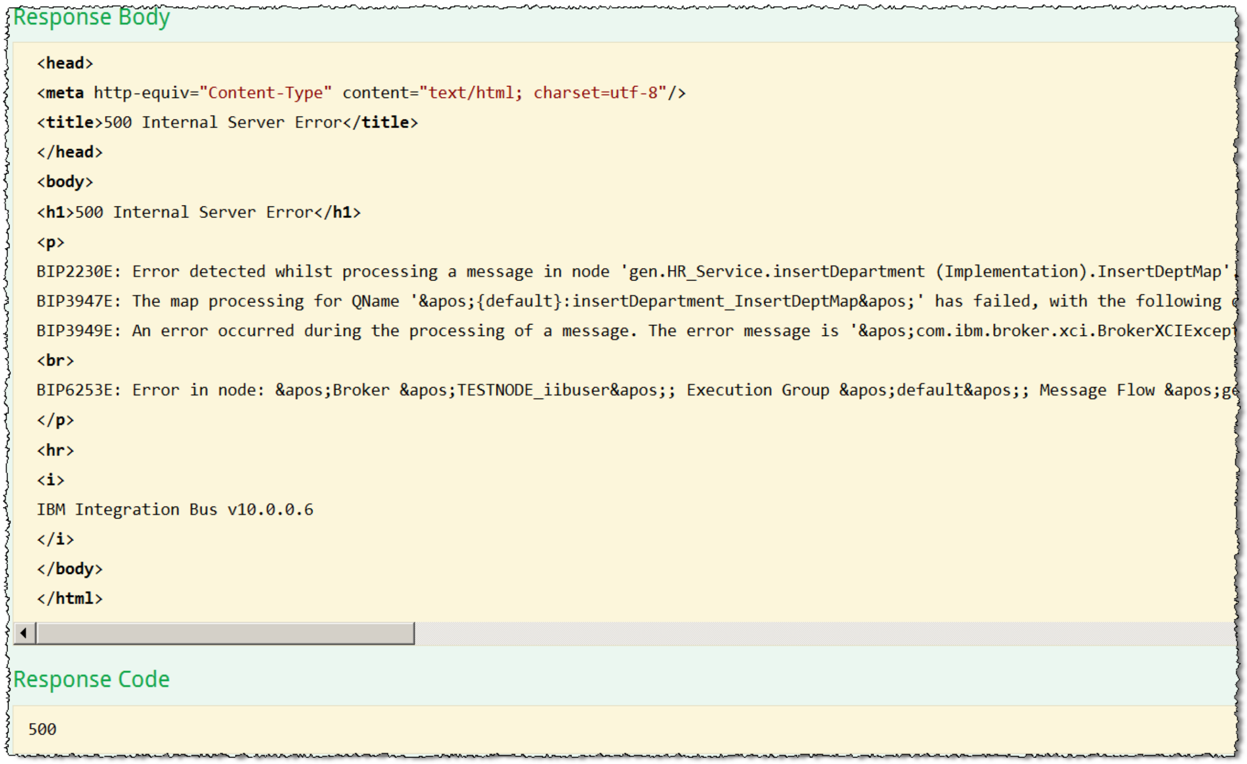
If you experience the following error message when attempting to test your REST API. Check your configuration of your IIB node. Cross Origin Resource Sharing needs to be configured correctly. Details of how to do this are available at the beginning of the lab guide.


Swagger UI
file:///C:/tools/swagger-ui-master/dist/index.html
Search
ODM
IIB
WAS
SDS
REST
IOT
Healthcare
Registration
Cloud
Build
Feedback
swagger
http://localhost:7800/HRDB_RESTServices/resources/swagger
api_key
Explore

Can't read from server. It may not have the appropriate access-control-origin settings.

## 6.2 JDBC Testing Issue

If you see the following error message or similar (the entry in the attached error message contains the text “**Could not locate JDBC Provider entry**”), check your configuration of the JDBC entry for the HRDB database. Details of how to configure the JDBC entry for HRDB are included at the beginning of this lab guide.



Response Body

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>500 Internal Server Error</title>
</head>
<body>
<h1>500 Internal Server Error</h1>
<p>
BIP2230E: Error detected whilst processing a message in node 'gen.HR_Service.insertDepartment (Implementation).InsertDeptMap'
BIP3947E: The map processing for QName '&apos;{default}:insertDepartment_InsertDeptMap&apos;' has failed, with the following
BIP3949E: An error occurred during the processing of a message. The error message is '&apos;com.ibm.broker.xci.BrokerXCIExcep
<br>
BIP6253E: Error in node: &apos;Broker &apos;TESTNODE_iibuser&apos;; Execution Group &apos;default&apos;; Message Flow &apos;g
</p>
<hr>
<i>
IBM Integration Bus v10.0.0.6
</i>
</body>
</html>
```

Response Code

```
500
```

## END OF LAB GUIDE