16L04

IBM Integration Bus

The RESTRequest Node

Featuring:

The REST Request node
Using a Mapping node to consolidate responses

September 2016
Hands-on lab built at product
Version 10.0.0.6

1. Introduction and Preparation

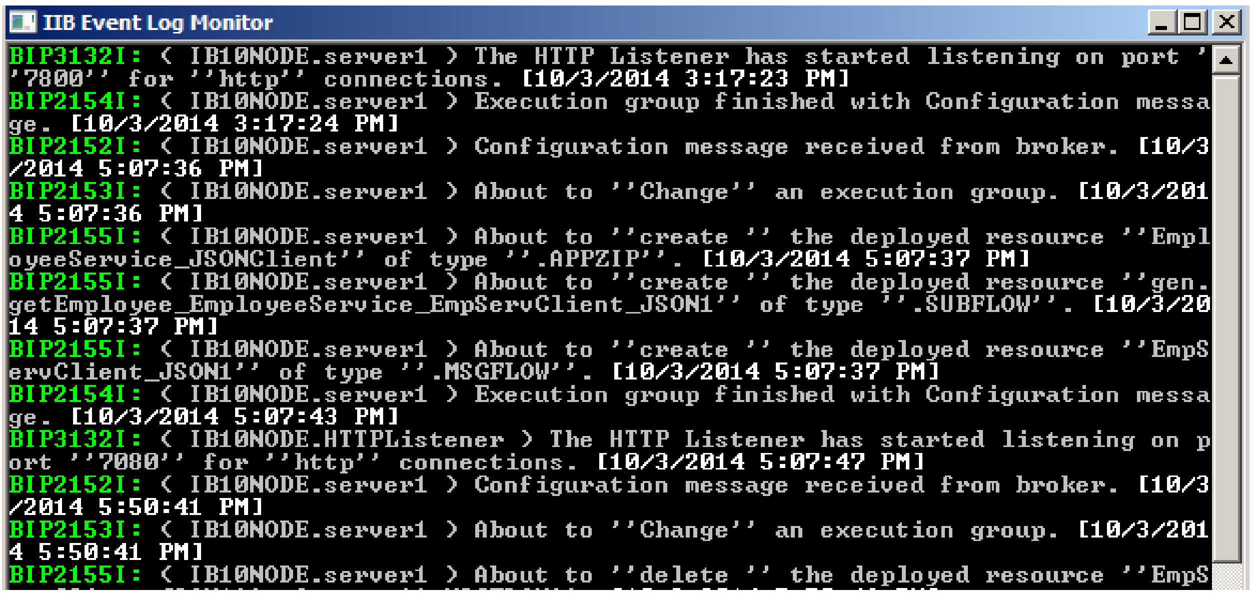
1.1 Introduction

In this lab you will use the RESTRequest node. This node is introduced in IIB v10.0.0.6.

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 Scenario

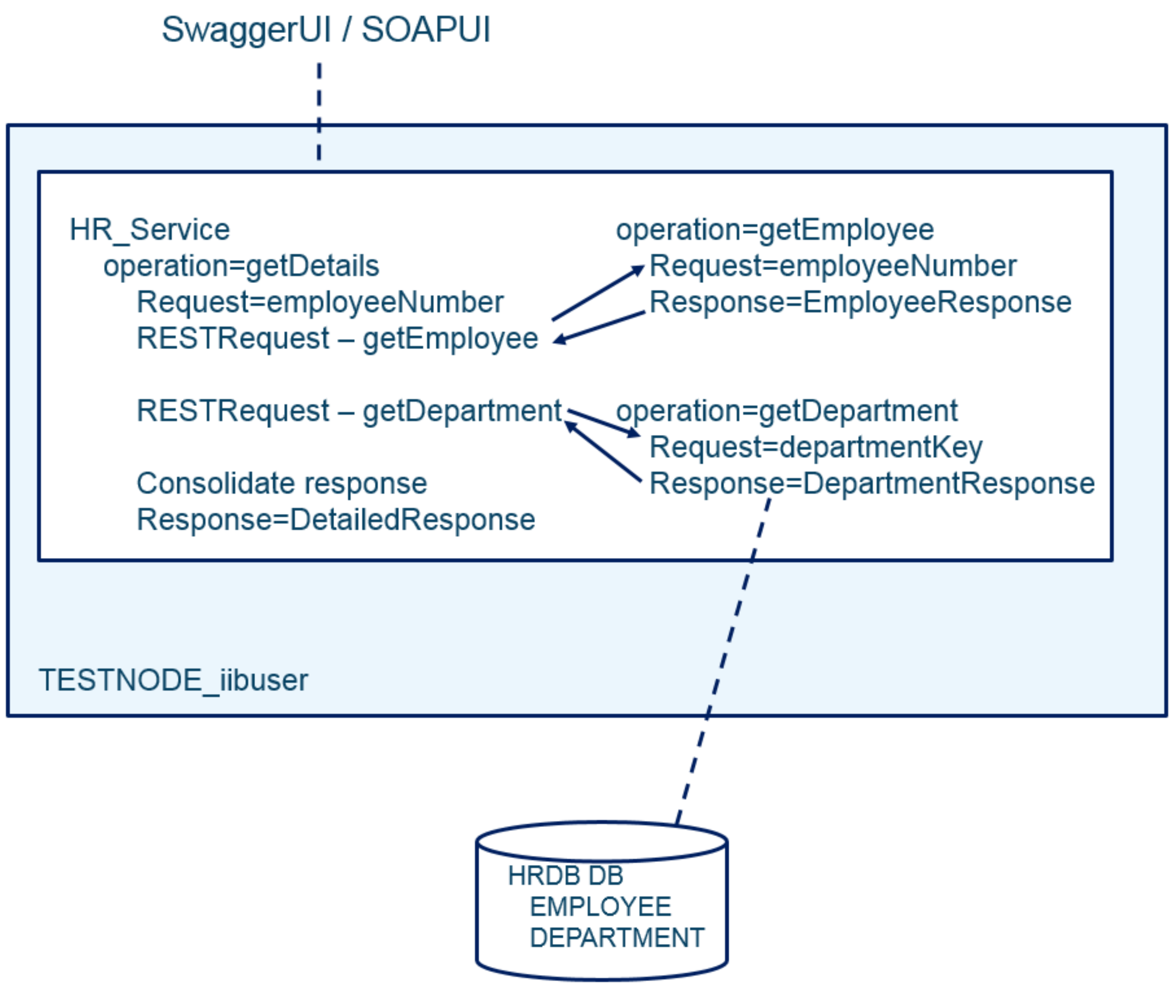
This lab will use the solution version of labs provided earlier in this series:

The provided solution version of the HR_Service REST API has implemented two operations, getEmployee and getDepartment. These retrieve data from the EMPLOYEE and DEPARTMENT tables of the HRDB database. In the getEmployee operation, one of the columns retrieved from the EMPLOYEE table is the WORKDEPT element; this is used as the parameter to retrieve the corresponding department details for the employee from the DEPARTMENT table.

In this lab, you will define and implement a third operation, getDetails. This operation will invoke the getEmployee and getDepartment operations, using the RESTRequest node. These nodes will be configured to return data which is placed into the Environment tree. Finally, a new Mapping node will consolidate the Employee and Department data (from the Environment tree), creating a new response message, DetailedResponse, before sending back to the originating client.

Important note
This lab scenario uses a REST Request node to invoke operations that are defined within the same REST API. Normally, local function calls like this would be performed with a more direct invocation (for example using the associated Mapping node directly, or an IIB subflow, or an IIB Callable Flow).

The REST Request node has been used in this lab to provide a simple illustration of the associated development and deployment tools, and to show how it can easily be extended to invoke REST services residing outside the IIB environment.



1.4 Message Models

The following message models are used by the HR_Service REST API.

- DBRESP – contains database response information
- EMPLOYEE – contains all returned columns from EMPLOYEE table
- DEPARTMENT - contains all returned columns from DEPARTMENT table
- EmployeeResponse
  - DBResp (type = DBRESP)
  - Employee (Array, type = EMPLOYEE)
- DepartmentResponse
  - DBResp (type = DBRESP)
  - Department (Array, type = DEPARTMENT)
- DetailedResponse
  - DBResp_employee (type = DBRESP)
  - Employee (type = EMPLOYEE, single object, not array)
  - DBResp_department (type = DBRESP)
  - Department (type = DEPARTMENT, single object, not array)

1.5 Configure TESTNODE_iibuser for REST APIs

The instructions in this lab guide are based on a Windows implementation, with a user named "iibuser".
The Windows VMWare image on which this lab is based is not available outside IBM, so you will need to provide your own software product installations where necessary.

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. Cross-Origin Resource Scripting (CORS) must be enabled for the IIB node to execute REST applications. This is also 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.

1. Ensure that TESTNODE_iibuser is started.

2. Check that CORS has been enabled on the IIB node by running the following command in an Integration Console:

```
mqsireportproperties TESTNODE_iibuser
      -e default
      -o HTTPConnector
      -r
```

3. If CORS is enabled, you will see the following lines (amongst others):

```
corsEnabled='true'
 corsAllowOrigins='*'
 corsAllowCredentials='false'
 corsExposeHeaders='Content-Type'
 corsMaxAge='-1'
 corsAllowMethods='GET,HEAD,POST,PUT,PATCH,DELETE,OPTIONS'
 corsAllowHeaders='Accept,Accept-Language,Content-Language,Content-
   Type'
```

4. If CORS has not been enabled, run the following commands:

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

mqsistop TESTNODE_iibuser

mqsistart TESTNODE_iibuser
```

1.6 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 Integration 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.

2. Extend the HR_Service REST API

In this section you will extend the provided REST API, HR_Service. The provided version of HR_Service has already implemented the getEmployee and getDepartment operations. (Review Labs 01 (department) and 02 (employee) for details of how to build these operations).

1. If you already have a workspace open, click File, Switch Workspace. Give the new workspace the name

   `c:\users\iibuser\IBM\IIB 10\workspace_RESTRequest`

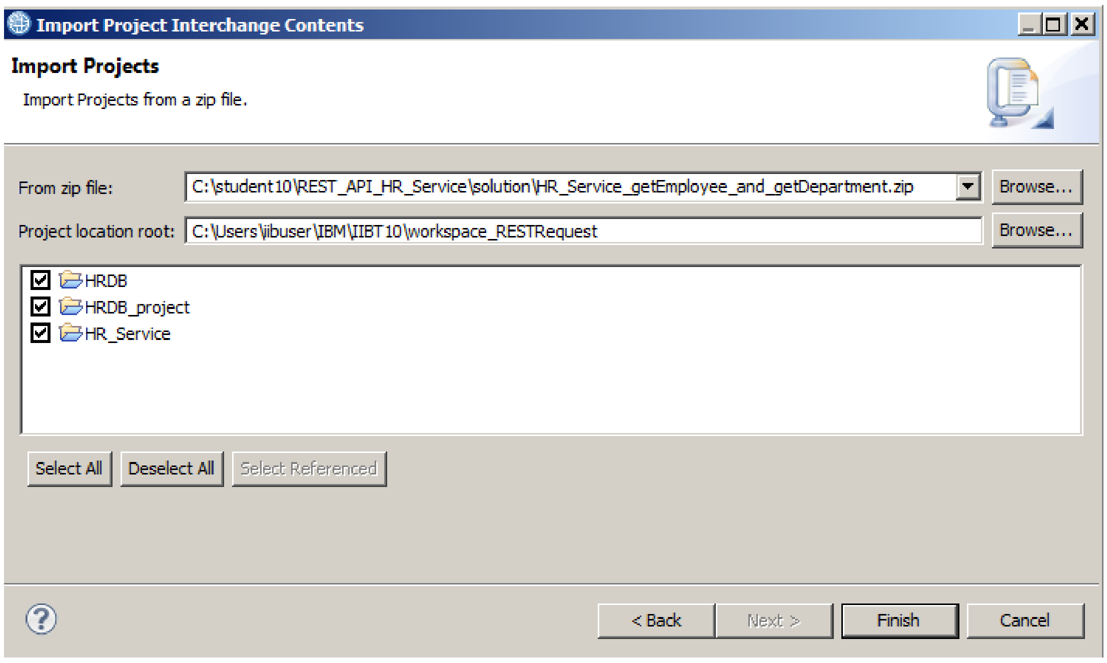
2. In the new workspace, import the Project Interchange file:

   `C:\student10\REST_API_HR_Service\solution\HR_Service_getEmployee_and_getDepartment.10.0.0.6.zip`

   Select all three projects from this PI file, and click Finish.

   HR_Service uses the EMPLOYEE and DEPARTMENT tables from the HRDB database. This requires the HRDB Database Definition project, which represents the tables schemas. This is used by the Mapping nodes that access these tables. The HRDB Shared Library and HRDB_project items contain the database definitions for the DB2 database HRDB.

   These items will not be developed in this lab - see the lab "Creating an Integration Service" for details of how to do this.

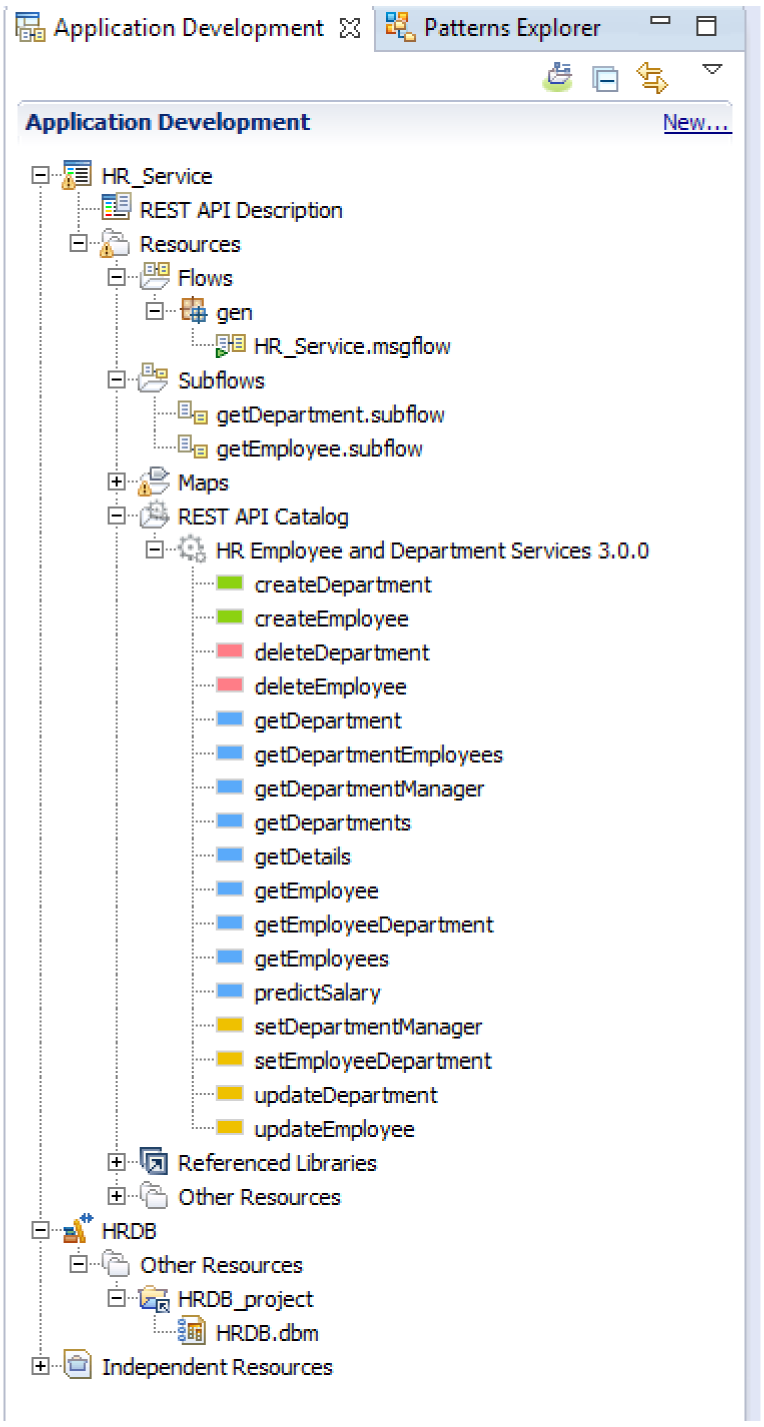

Import Project Interchange Contents
Import Projects
Import Projects from a zip file.
From zip file: C:\student10\REST_API_HR_Service\solution\HR_Service_getEmployee_and_getDepartment.zip
Project location root: C:\Users\iibuser\IBM\IIBT10\workspace_RESTRequest

- [x] HRDB
- [x] HRDB_project
- [x] HR_Service

3. When imported, you will see the HR_Service REST API project, and the HRDB shared library. The shared library has a library reference from the HR_Service REST API.

   You will see two subflows are present in HR_Service. This indicates that two operations have already been implemented in this REST API.

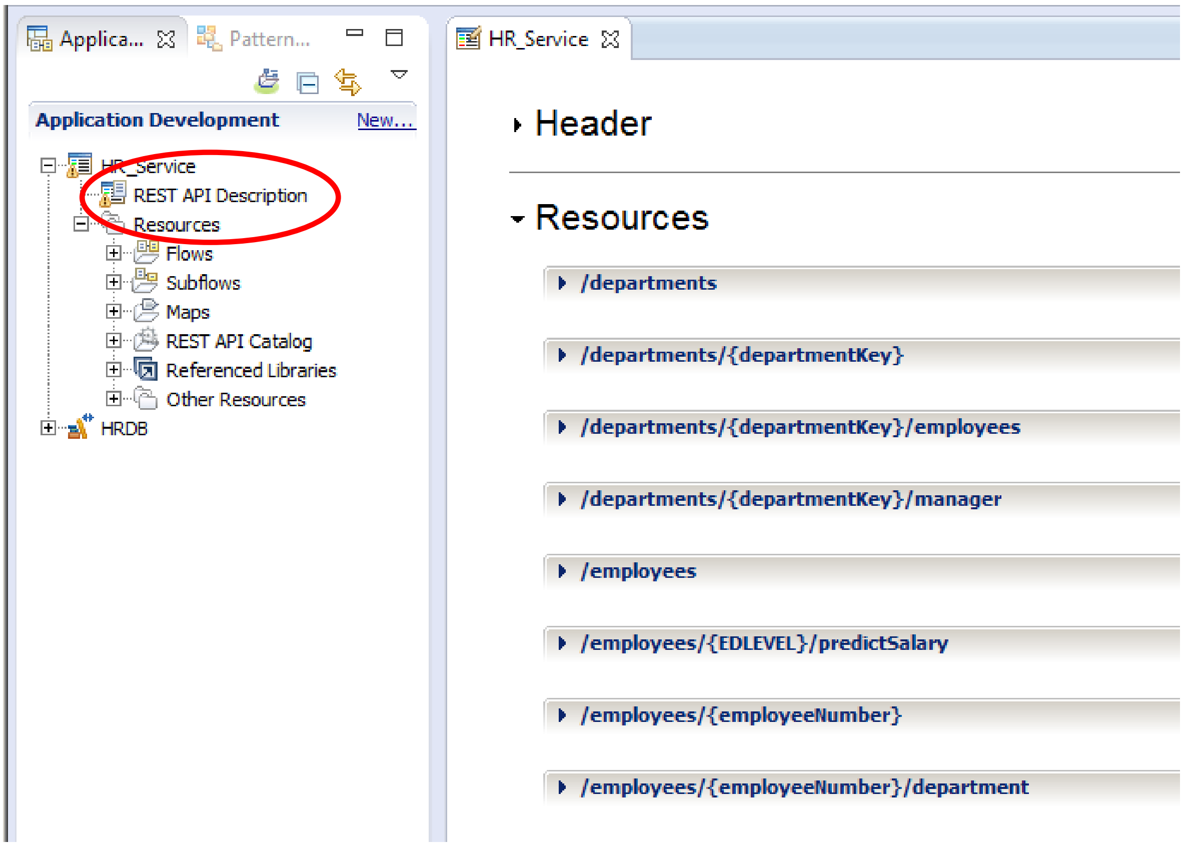
   Expanding the REST API Catalog will show the entire list of operations that are defined in this REST API. However, at this point, only two of these operations have been implemented.



```
HR_Service
    REST API Description
    Resources
        Flows
            gen
                HR_Service.msgflow
        Subflows
            getDepartment.subflow
            getEmployee.subflow
        Maps
        REST API Catalog
            HR Employee and Department Services 3.0.0
                createDepartment
                createEmployee
                deleteDepartment
                deleteEmployee
                getDepartment
                getDepartmentEmployees
                getDepartmentManager
                getDepartments
                getDetails
                getEmployee
                getEmployeeDepartment
                getEmployees
                predictSalary
                setDepartmentManager
                setEmployeeDepartment
                updateDepartment
                updateEmployee
        Referenced Libraries
        Other Resources
HRDB
    Other Resources
        HRDB_project
            HRDB.dbm
Independent Resources
```

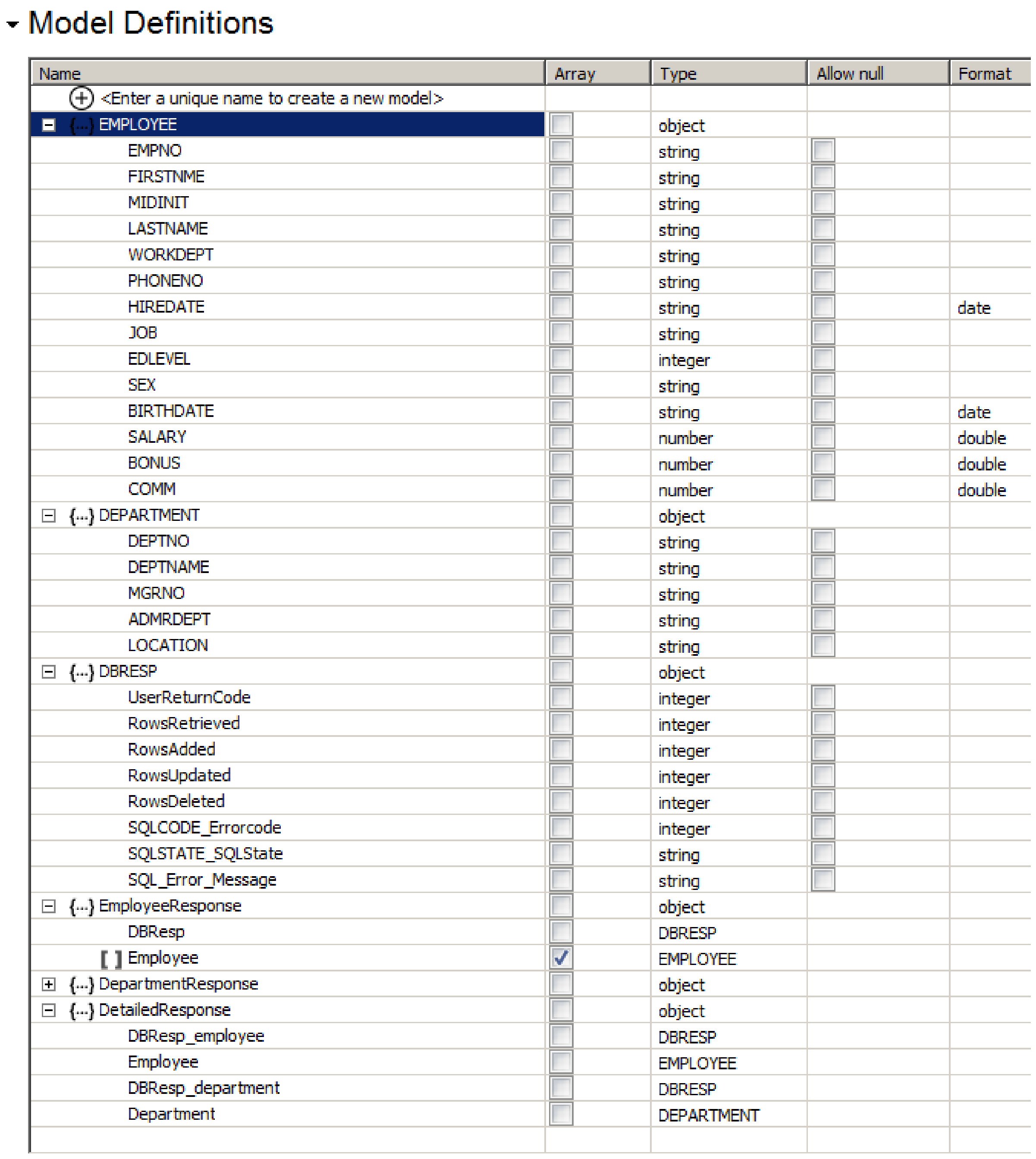
4. Open (double-click) the REST API Description. This will show the Header and Resources definitions in the REST API.

   Expand /departments/{departmentKey} or /employees/{employeeNumber}. The getDepartment and getEmployee operations have been implemented; the implementation of these operations can be opened by using the icon on the right side of the window for these operations (you will need to use the scroll bar to see these icons).



5. Further down the HR_Service editor, you will see the supplied Model Definitions.

   Expand the supplied models. These will be used later in the lab; in particular, the getDetails operation will use the DetailedResponse model.

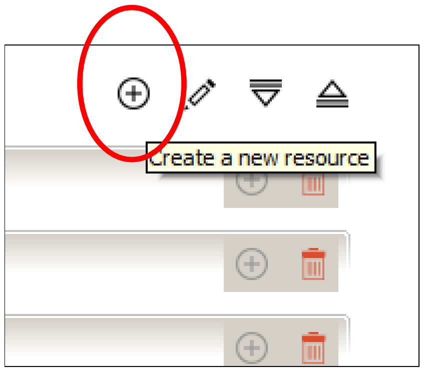


Model Definitions

Name	Array	Type	Allow null	Format
<Enter a unique name to create a new model>				
{...} EMPLOYEE	☐	object		
EMPNO	☐	string	☐	
FIRSTNME	☐	string	☐	
MIDINIT	☐	string	☐	
LASTNAME	☐	string	☐	
WORKDEPT	☐	string	☐	
PHONENO	☐	string	☐	
HIREDATE	☐	string	☐	date
JOB	☐	string	☐	
EDLEVEL	☐	integer	☐	
SEX	☐	string	☐	
BIRTHDATE	☐	string	☐	date
SALARY	☐	number	☐	double
BONUS	☐	number	☐	double
COMM	☐	number	☐	double
{...} 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	☐	integer	☐	
SQLSTATE_SQLState	☐	string	☐	
SQL_Error_Message	☐	string	☐	
{...} EmployeeResponse	☐	object		
DBResp	☐	DBRESP		
[ ] Employee	✓	EMPLOYEE		
{...} DepartmentResponse	☐	object		
{...} DetailedResponse	☐	object		
DBResp_employee	☐	DBRESP		
Employee	☐	EMPLOYEE		
DBResp_department	☐	DBRESP		
Department	☐	DEPARTMENT		

2.1 Add a new resource to HR_Service

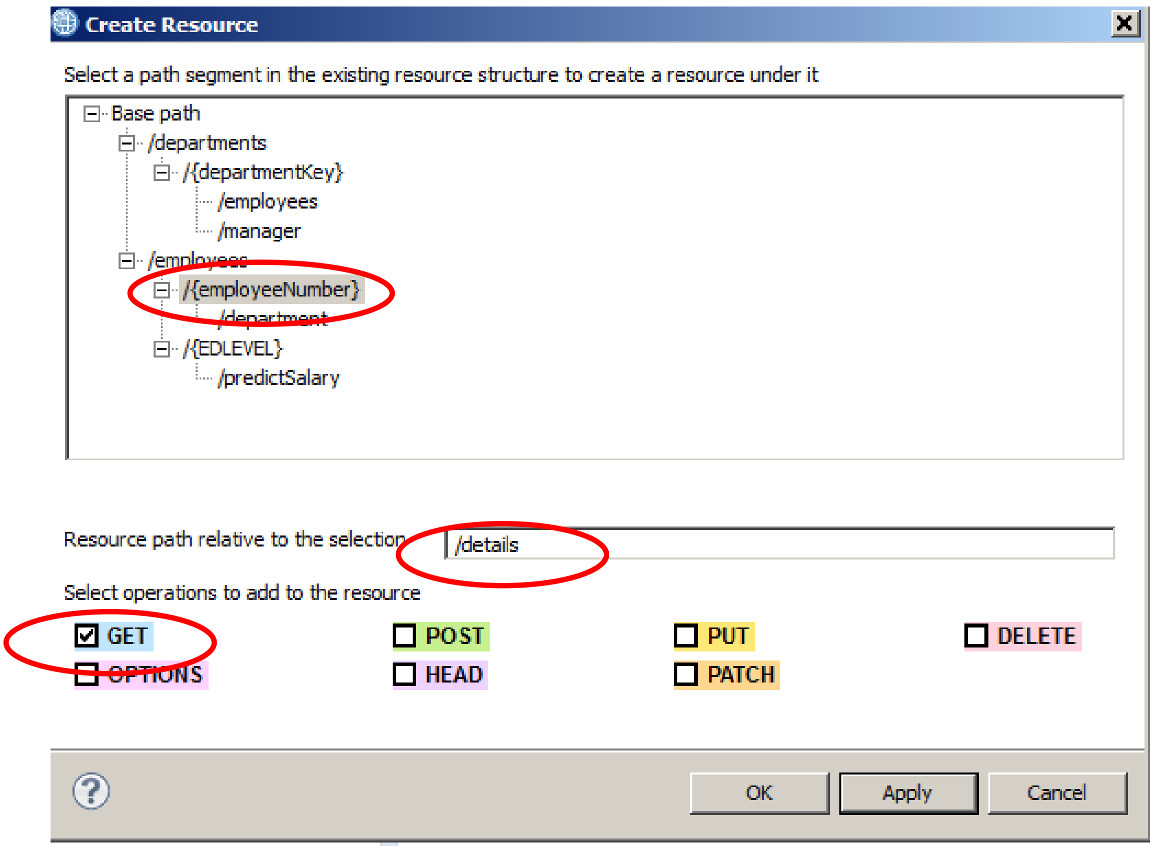
1. In the Integration Toolkit, In the Resources section of the REST API editor, scroll the window to the right, and click the "Create a new resource" icon.



2. In the Create Resource window:

   - Highlight the **/{employeeNumber}** path segment
   - Set the "resource path relative to selection" to **/details**
   - Select the GET operation

   Click OK.



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

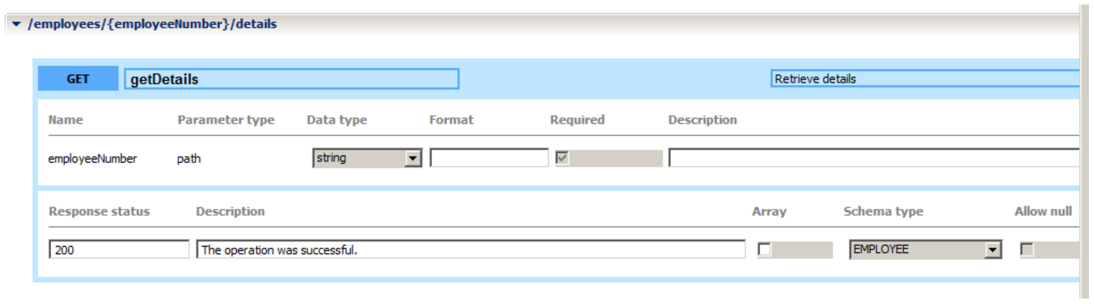
```
Base path
  /departments
    /{departmentKey}
      /employees
      /manager
  /employees
    /{employeeNumber}
      /department
    /{EDLEVEL}
      /predictSalary
```

Resource path relative to the selection: /details

Select operations to add to the resource

[✓ GET] POST PUT DELETE OPTIONS HEAD PATCH

3. The editor will create a new resource, as shown. Note that the schema type for the response message will be set to EMPLOYEE (because this is the first available model definition).



/employees/{employeeNumber}/details

GET getDetails Retrieve details

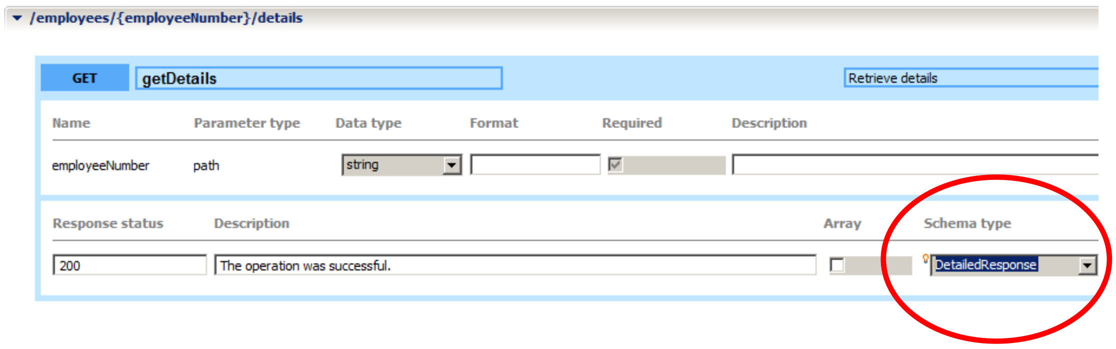
Name	Parameter type	Data type	Format	Required	Description
employeeNumber	path	string		✓	

Response status	Description	Array	Schema type	Allow null
200	The operation was successful.	☐	EMPLOYEE	☐

4. Change the schema type for the response to DetailedResponse.

Save the HR_Service REST API (Ctrl-S).

The getDetails operation is now ready to be implemented.



/employees/{employeeNumber}/details

GET getDetails Retrieve details

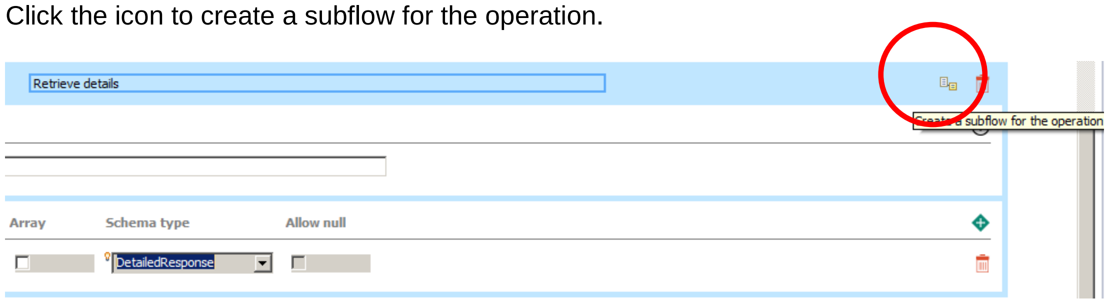
Name	Parameter type	Data type	Format	Required	Description
employeeNumber	path	string		✓	

Response status	Description	Array	Schema type
200	The operation was successful.	☐	DetailedResponse

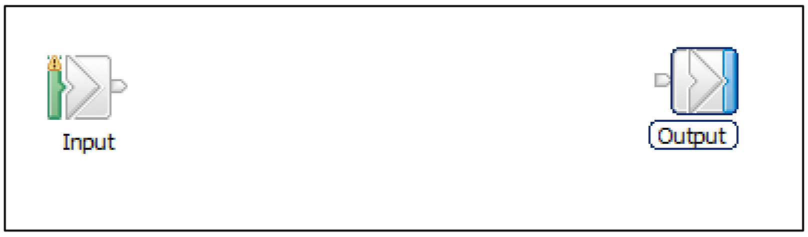
2.1.1 Implement the getDetails operation

1. In the editor for the getDetails operation (Resource), scroll to the right side of the editor.

   Click the icon to create a subflow for the operation.

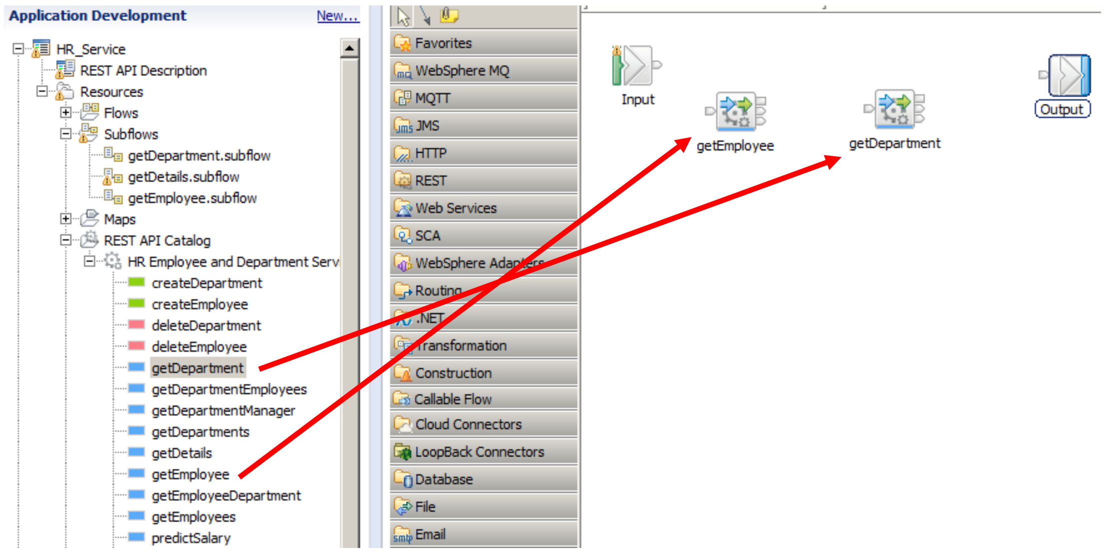


2. The subflow editor will open, and will be populated with Input and Output nodes, ready for the implementation.



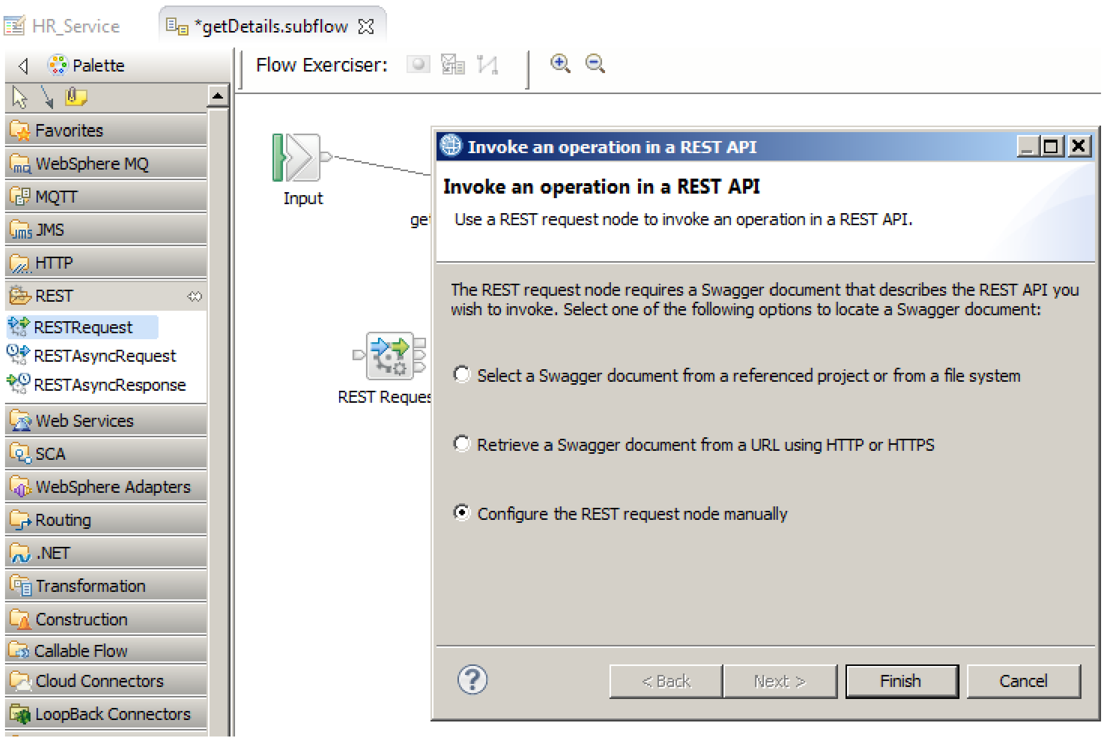
3. In the Application Development view, expand Resources, REST API Catalog, and "HR Employee and Department Services 3.0.0".

   Drag/drop an instance of the **getEmployee** operation onto the flow editor, followed by an instance of the **getDepartmen**t operation.



This will automatically add a REST Request node to the flow editor, and populate the node properties with values that are needed by the getEmployee and getDepartment operations.

(Alternatively, you can drop a native REST Request node onto the flow editor, specify a manual configuration, as shown below, and then drop the required operation onto the REST Request node. This will automatically set (or update) the node properties.)



Invoke an operation in a REST API

Use a REST request node to invoke an operation in a REST API.

The REST request node requires a Swagger document that describes the REST API you wish to invoke. Select one of the following options to locate a Swagger document:

- ○ Select a Swagger document from a referenced project or from a file system
- ○ Retrieve a Swagger document from a URL using HTTP or HTTPS
- [✓ Configure the REST request node manually]

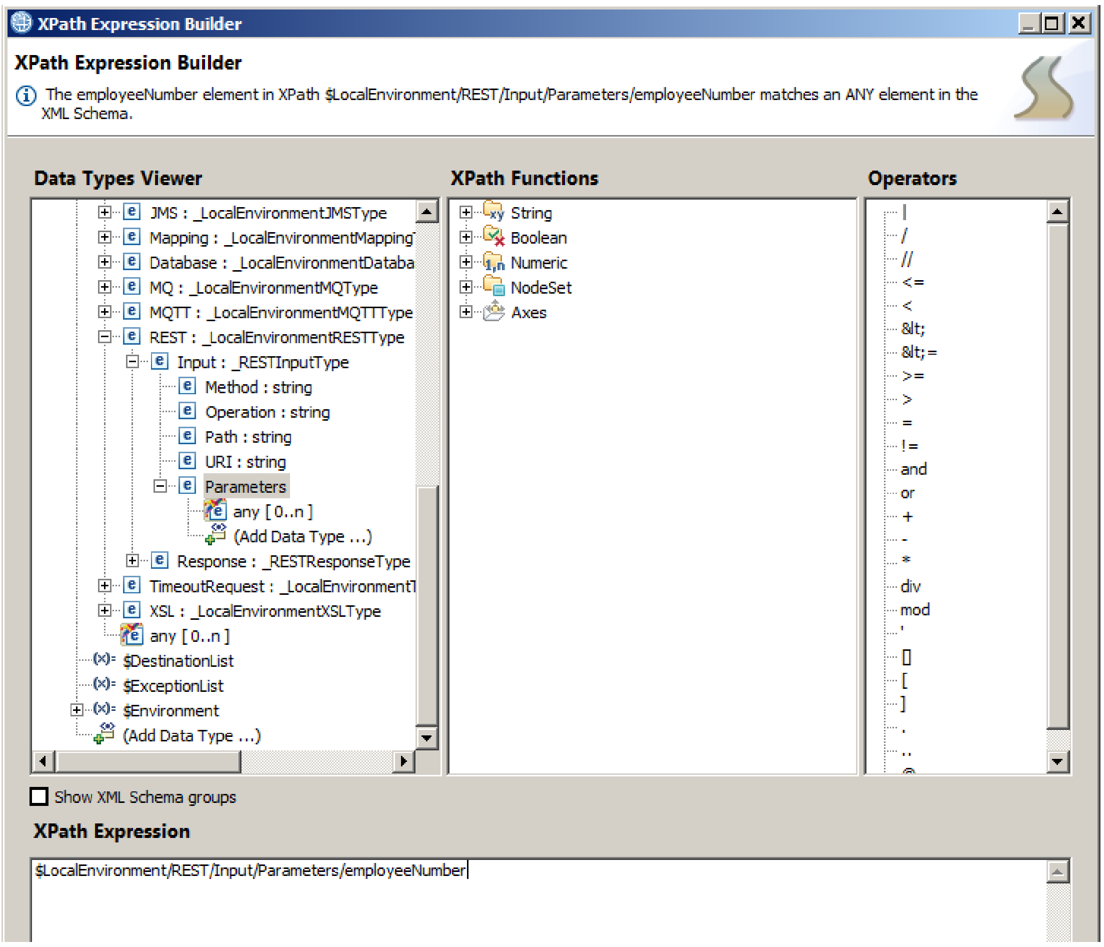
4. Highlight the getEmployee node and make the following changes to the node properties:

   Request tab

   - Set Expression for employeeNumber = **$LocalEnvironment/REST/Input/Parameters/employeeNumber**

   This value is required because the getDetails operation has the same input parameter as getEmployee, so the employeeNumber parameter is passed straight through to the getDetails operation.

   You can use the XPath Expression Builder to provide almost all the required path, and manually add the “employeeNumber” variable to the end of the generated path. To open the expression builder, highlight the expression line, and click the small box containing three dots.



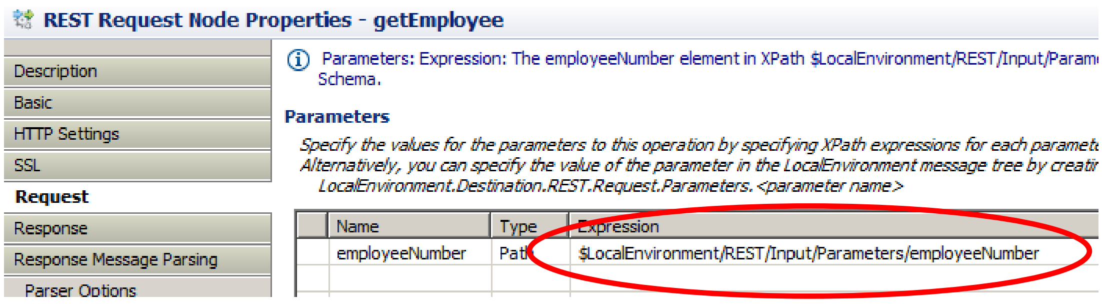
XPath Expression Builder

The employeeNumber element in XPath $LocalEnvironment/REST/Input/Parameters/employeeNumber matches an ANY element in the XML Schema.

XPath Expression

```
$LocalEnvironment/REST/Input/Parameters/employeeNumber
```

When complete, the expression should look like this:



REST Request Node Properties - getEmployee

Parameters: Expression: The employeeNumber element in XPath $LocalEnvironment/REST/Input/Param
Schema.

Parameters

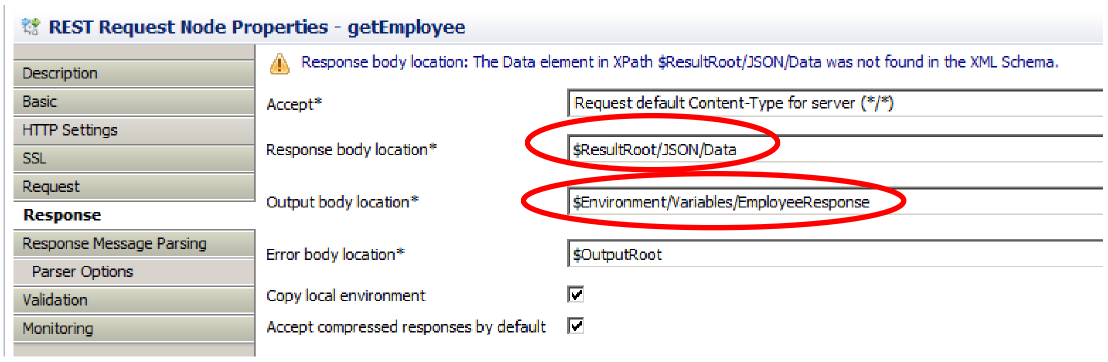
Specify the values for the parameters to this operation by specifying XPath expressions for each paramete
Alternatively, you can specify the value of the parameter in the LocalEnvironment message tree by creatir
LocalEnvironment.Destination.REST.Request.Parameters.<parameter name>

Name	Type	Expression
employeeNumber	Path	$LocalEnvironment/REST/Input/Parameters/employeeNumber

5. For the Response tab:

- Set Response body location = **$ResultRoot/JSON/Data** (this means that just this part of the response message will be used, not the whole message).
  Note the XPath Expression Builder may start automatically; the required JSON/Data element is not available using this tool, so ignore any warning messages at this point.

- Set Output body location = **$Environment/Variables/EmployeeResponse** (this means that the response message from getEmployee will be stored in the Environment tree, under a variable named EmployeeResponse).


REST Request Node Properties - getEmployee

Response body location: The Data element in XPath $ResultRoot/JSON/Data was not found in the XML Schema.

Accept*	Request default Content-Type for server (*/*)
Response body location*	$ResultRoot/JSON/Data
Output body location*	$Environment/Variables/EmployeeResponse
Error body location*	$OutputRoot
Copy local environment	✓
Accept compressed responses by default	✓

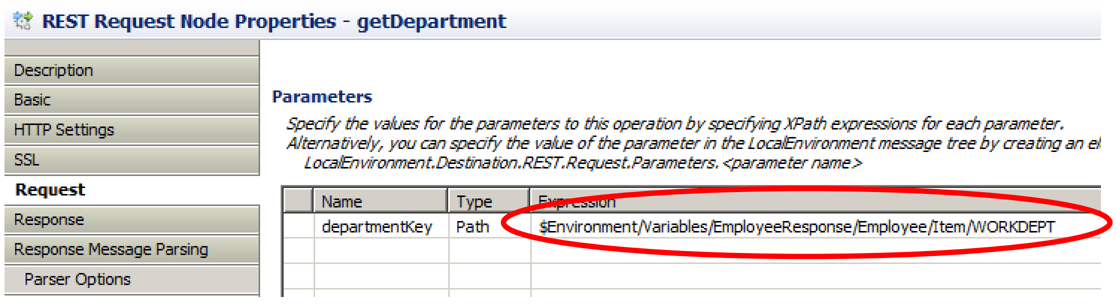
6. Highlight the getDepartment node and make the following changes to the node properties:

   Request tab

   - Set Expression for departmentKey = **$Environment/Variables/EmployeeResponse/Employee/Item/WORKDEPT**

     As above, ignore any input or text provided by the XPath expression builder.

   This means that the input to the getDepartment operation will be extracted from the response message of the getEmployee operation (which was stored in the Environment tree).



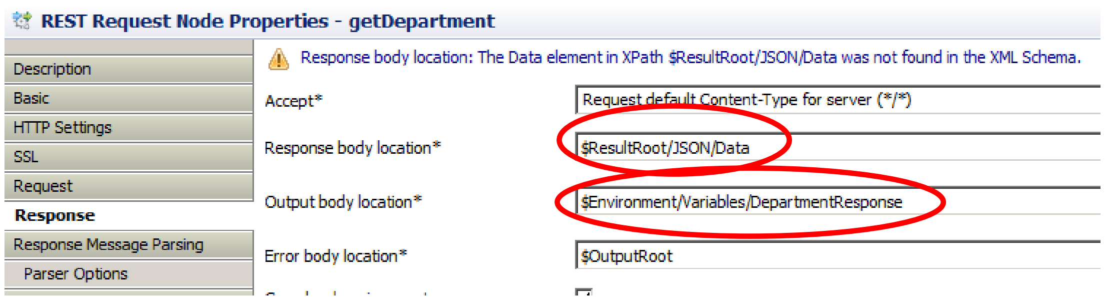
Specify the values for the parameters to this operation by specifying XPath expressions for each parameter. Alternatively, you can specify the value of the parameter in the LocalEnvironment message tree by creating an el LocalEnvironment.Destination.REST.Request.Parameters.<parameter name>

Name	Type	Expression
departmentKey	Path	$Environment/Variables/EmployeeResponse/Employee/Item/WORKDEPT

Response tab

- Set Response body location = **$ResultRoot/JSON/Data** (this means that just this part of the response message will be used, not the whole message).

- Set Output body location = **$Environment/Variables/DepartmentResponse** (this means that the response message will be stored in the Environment tree, under a variable named DepartmentResponse.



Response body location: The Data element in XPath $ResultRoot/JSON/Data was not found in the XML Schema.

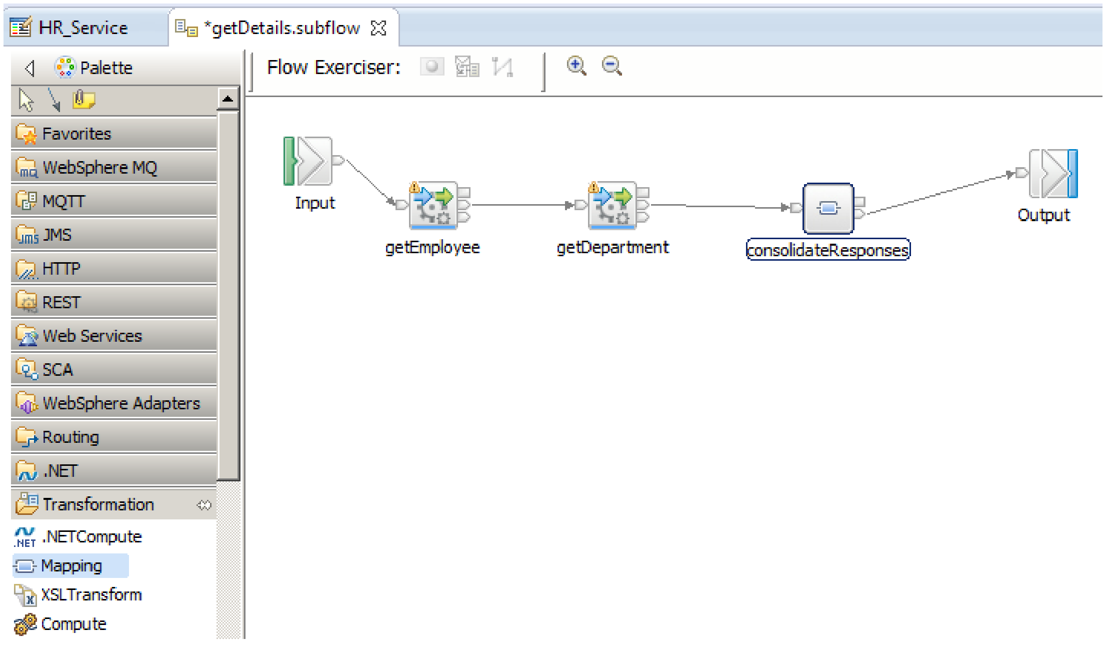
Accept*	Request default Content-Type for server (*/*)
Response body location*	$ResultRoot/JSON/Data
Output body location*	$Environment/Variables/DepartmentResponse
Error body location*	$OutputRoot

Save the subflow at this time (Ctrl-S).

2.1.2 Add Mapping Node to build response message

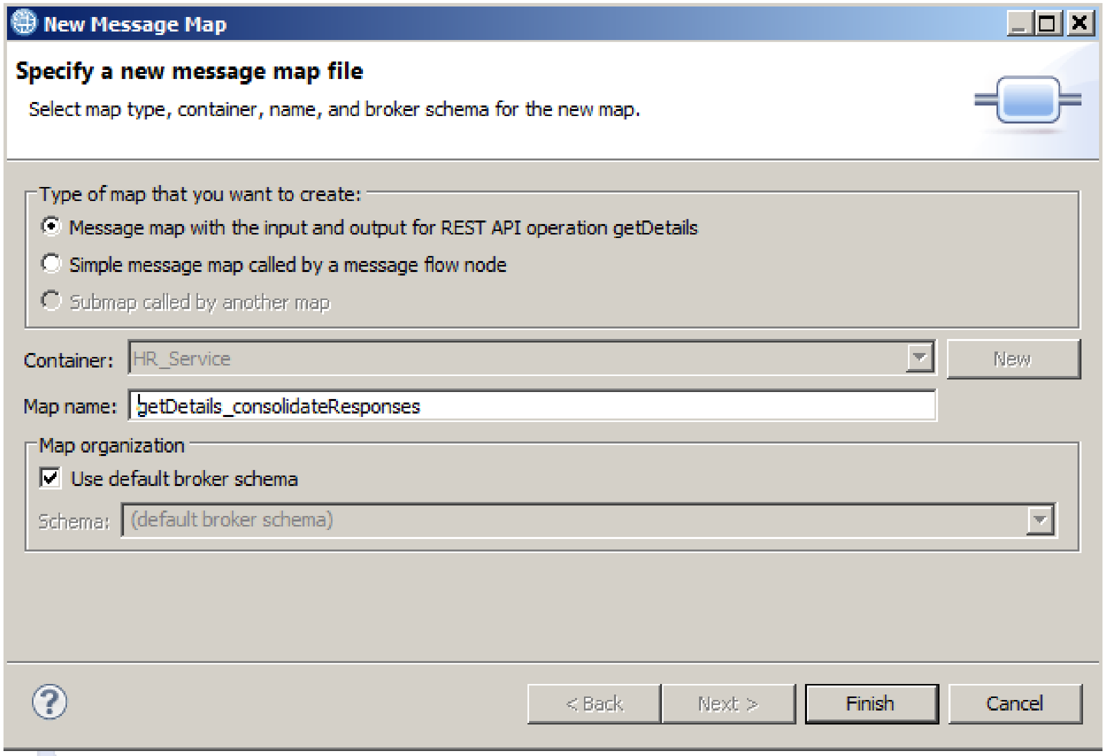
1. To complete the operation, add a new Mapping node to the subflow. Name it "consolidateResponses".

   Connect the nodes as shown.



2. Open (double-click) the consolidateResponses map node.

   In the New Message Map wizard, accept the default map type, and click Finish. This will automatically create a map where the output assembly will be generated as DetailedResponse, which is the required response message for the getDetails operation.



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 getDetails
- Simple message map called by a message flow node
- Submap called by another map

Container: HR_Service

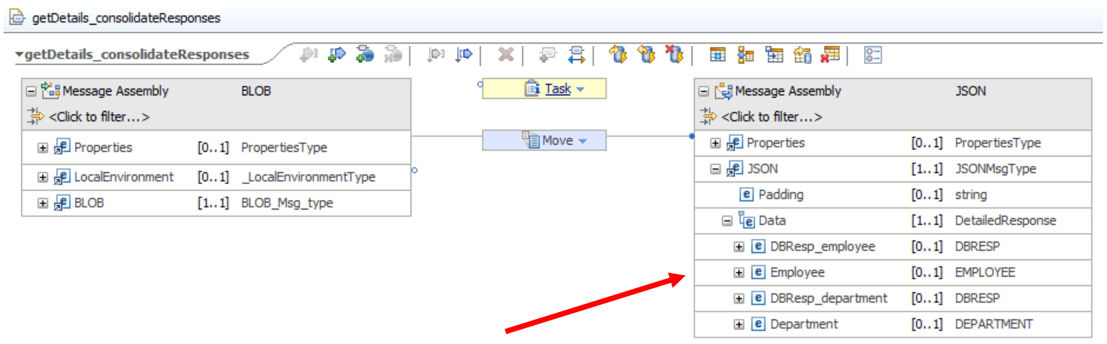
Map name: getDetails_consolidateResponses

Map organization
- [✓] Use default broker schema

Schema: (default broker schema)

3. In the map editor, expand the output JSON/Data message assembly. You will see that the assembly has been pre-populated with the DetailedResponse message.

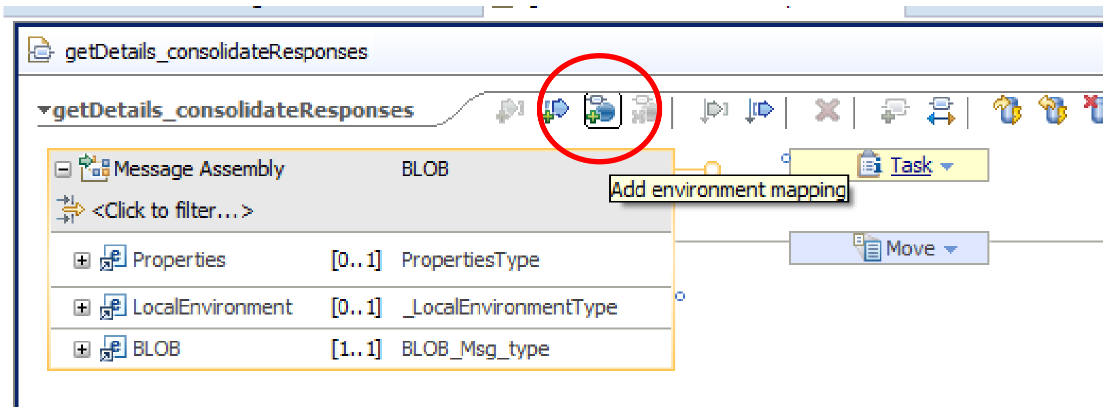
   Note that the DetailedResponse message contains just JSON object elements, not arrays.



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]	DetailedResponse
DBResp_employee	[0..1]	DBRESP
Employee	[0..1]	EMPLOYEE
DBResp_department	[0..1]	DBRESP
Department	[0..1]	DEPARTMENT

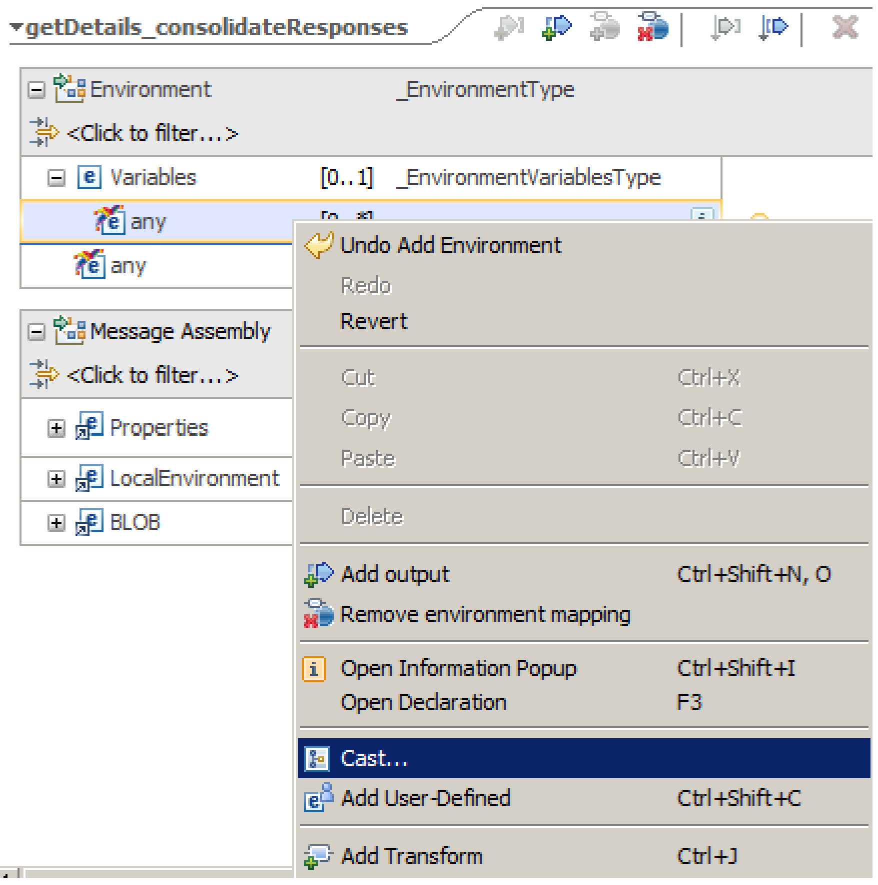
4. Add the Environment tree to the map editor. Do this by clicking the Add Environment icon as shown.



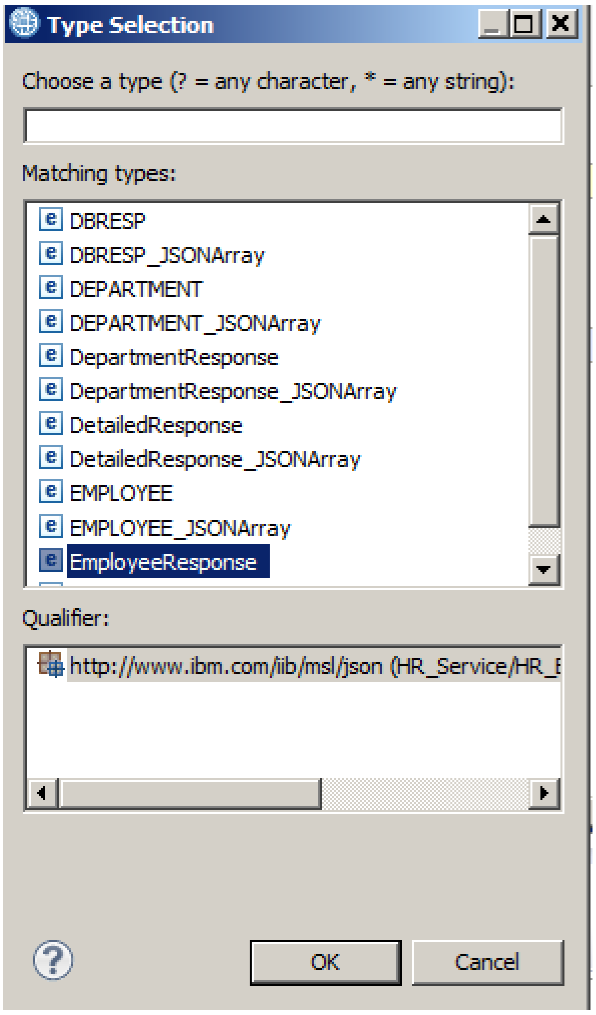
Message Assembly		BLOB
<Click to filter…>		
Properties	[0..1]	PropertiesType
LocalEnvironment	[0..1]	_LocalEnvironmentType
BLOB	[1..1]	BLOB_Msg_type

5. Expand the Environment tree, and expand Variables.

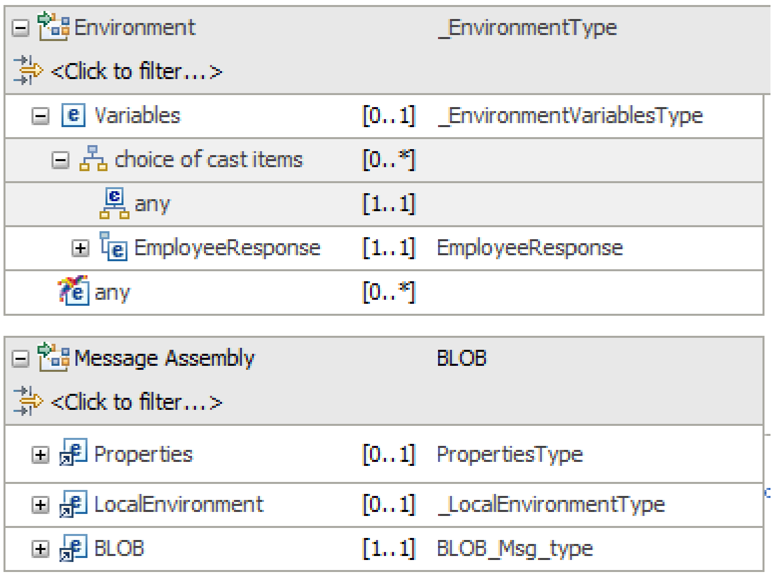
   Under Variables, right-click the “any” element, and select Cast from the context menu. **(Be sure to select the any element under Variables).**


getDetails_consolidateResponses
Environment _EnvironmentType
<Click to filter...>
Variables [0..1] _EnvironmentVariablesType
any
any
Message Assembly
<Click to filter...>
Properties
LocalEnvironment
BLOB
Undo Add Environment
Redo
Revert
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete
Add output Ctrl+Shift+N, O
Remove environment mapping
Open Information Popup Ctrl+Shift+I
Open Declaration F3
Cast...
Add User-Defined Ctrl+Shift+C
Add Transform Ctrl+J

6. From the Type Selection window, select EmployeeResponse and click OK.



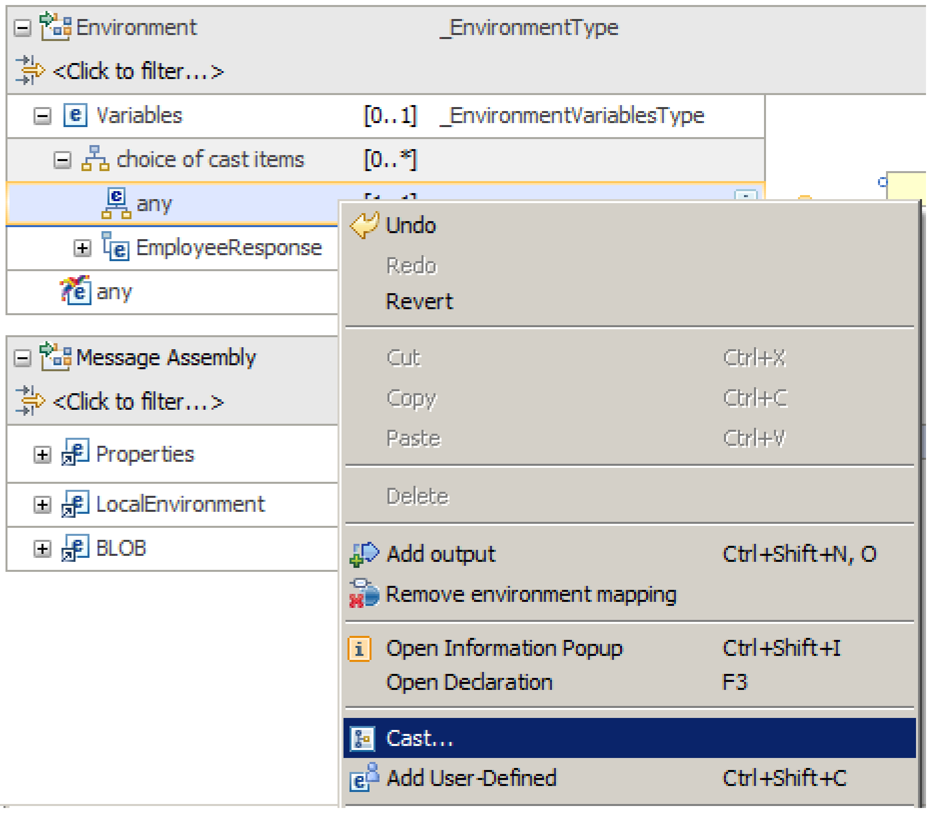
7. EmployeeResponse will have been added to the Environment/Variables folder. This was done automatically by the mapping editor, as a result of the selection of the getEmployee operation.



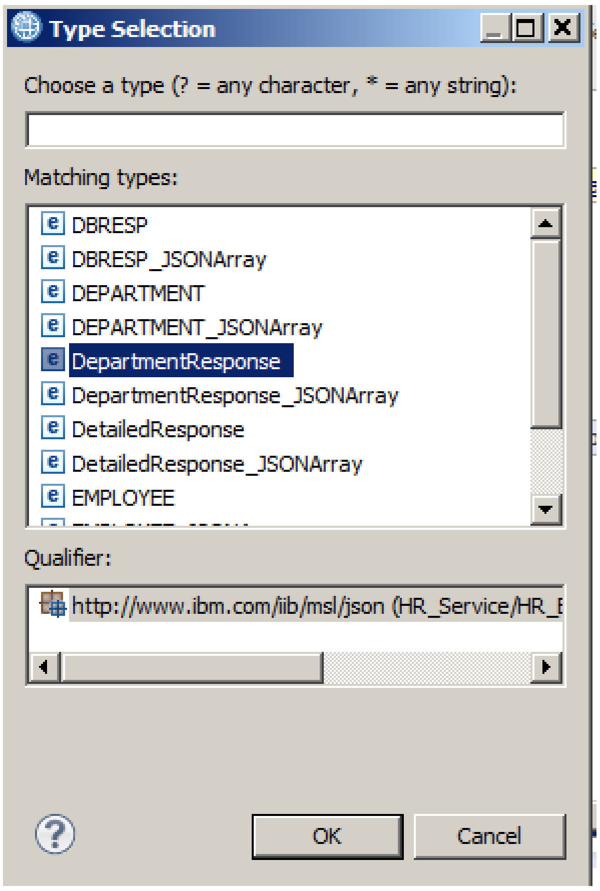
Environment		_EnvironmentType
<Click to filter…>		
Variables	[0..1]	_EnvironmentVariablesType
choice of cast items	[0..*]	
any	[1..1]	
EmployeeResponse	[1..1]	EmployeeResponse
any	[0..*]	

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

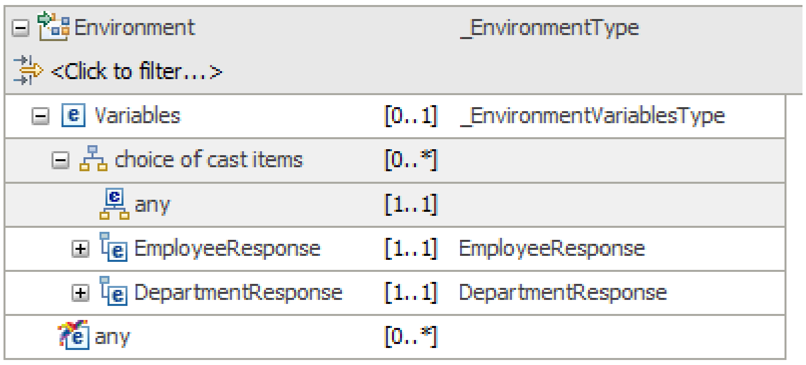
8. Right-click the “Variables/any” element again, and select Cast again.


Environment _EnvironmentType
<Click to filter...>
Variables [0..1] _EnvironmentVariablesType
choice of cast items [0..*]
any
EmployeeResponse
any
Message Assembly
<Click to filter...>
Properties
LocalEnvironment
BLOB
Undo
Redo
Revert
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete
Add output Ctrl+Shift+N, O
Remove environment mapping
Open Information Popup Ctrl+Shift+I
Open Declaration F3
Cast...
Add User-Defined Ctrl+Shift+C

9. Select DepartmentResponse and click OK.


Type Selection
Choose a type (? = any character, * = any string):
Matching types:
DBRESP
DBRESP_JSONArray
DEPARTMENT
DEPARTMENT_JSONArray
DepartmentResponse
DepartmentResponse_JSONArray
DetailedResponse
DetailedResponse_JSONArray
EMPLOYEE
Qualifier:
http://www.ibm.com/iib/msl/json (HR_Service/HR_
OK
Cancel

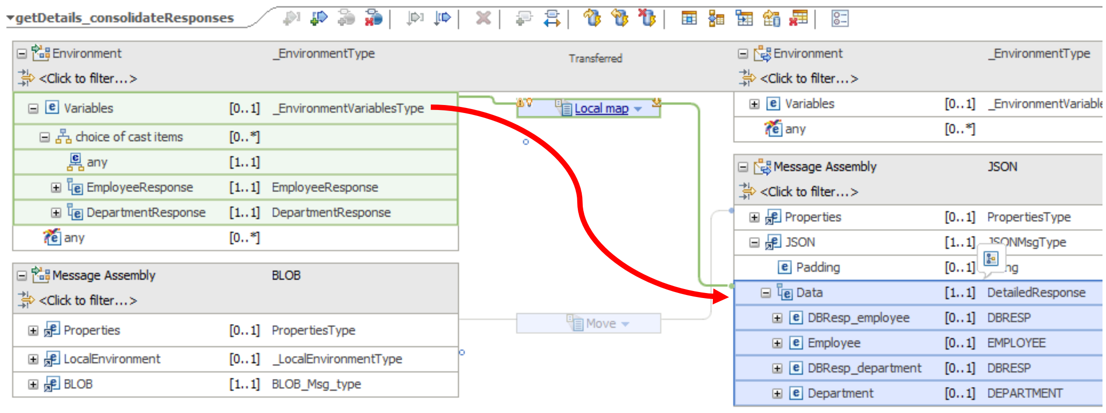
10. The Environment/Variables folder will now contain references to the response data from both the getEmployee and getDepartment REST Requests.



Environment		_EnvironmentType
<Click to filter…>		
Variables	[0..1]	_EnvironmentVariablesType
choice of cast items	[0..*]	
any	[1..1]	
EmployeeResponse	[1..1]	EmployeeResponse
DepartmentResponse	[1..1]	DepartmentResponse
any	[0..*]	

11. Connect the input Environment/Variables to the output JSON/Data. This will result in a Local Map transform.

Make sure you connect these elements as directed, and not the lower level elements. Although connecting lower-level elements is possible in some situations, the logical structure of the message assemblies in this example (including arrays for EmployeeResponse and DepartmentResponse mean that this is easier within a Local Map.

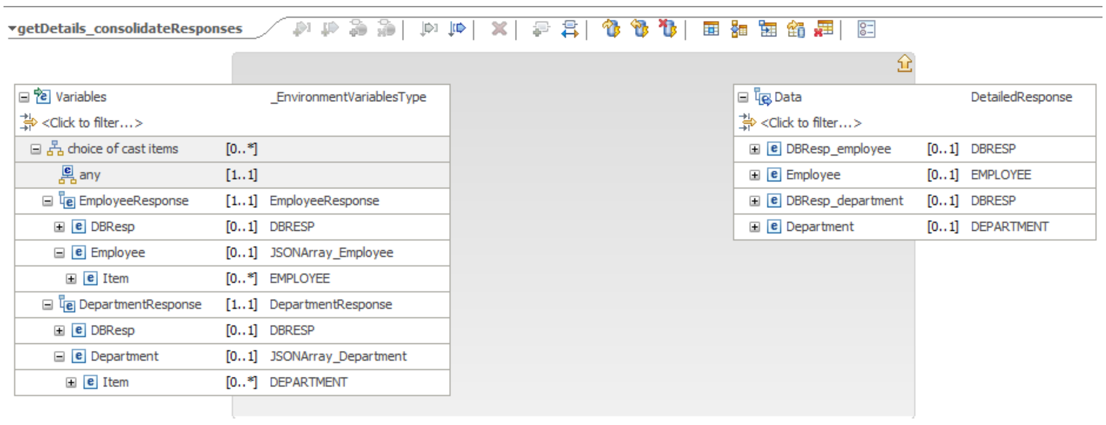


Environment		_EnvironmentType
<Click to filter…>		
Variables	[0..1]	_EnvironmentVariablesType
choice of cast items	[0..*]	
any	[1..1]	
EmployeeResponse	[1..1]	EmployeeResponse
DepartmentResponse	[1..1]	DepartmentResponse
any	[0..*]	
Message Assembly		BLOB
<Click to filter…>		
Properties	[0..1]	PropertiesType
LocalEnvironment	[0..1]	_LocalEnvironmentType
BLOB	[1..1]	BLOB_Msg_type

Environment		_EnvironmentType
<Click to filter…>		
Variables	[0..1]	_EnvironmentVariabl
any	[0..*]	
Message Assembly		JSON
<Click to filter…>		
Properties	[0..1]	PropertiesType
JSON	[1..1]	JSONMsgType
Padding	[0..1]	ng
Data	[1..1]	DetailedResponse
DBResp_employee	[0..1]	DBRESP
Employee	[0..1]	EMPLOYEE
DBResp_department	[0..1]	DBRESP
Department	[0..1]	DEPARTMENT

12. Click “Local Map” to specify the precise element mappings.

When in the Local Map, expand the input assembly as shown. Make sure the Employee and Department elements show the “Item” elements (remember, these are returned from their respective operations as arrays).

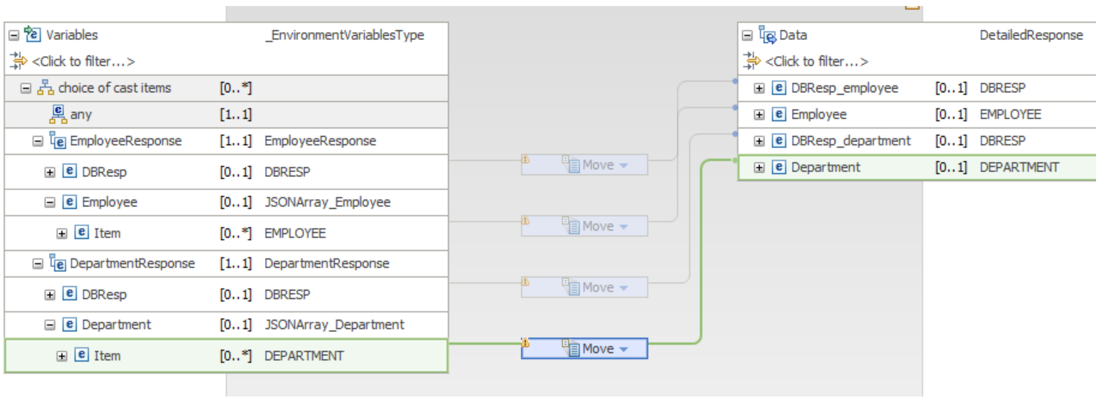


Variables		_EnvironmentVariablesType
<Click to filter…>		
choice of cast items	[0..*]	
any	[1..1]	
EmployeeResponse	[1..1]	EmployeeResponse
DBResp	[0..1]	DBRESP
Employee	[0..1]	JSONArray_Employee
Item	[0..*]	EMPLOYEE
DepartmentResponse	[1..1]	DepartmentResponse
DBResp	[0..1]	DBRESP
Department	[0..1]	JSONArray_Department
Item	[0..*]	DEPARTMENT

Data		DetailedResponse
<Click to filter…>		
DBResp_employee	[0..1]	DBRESP
Employee	[0..1]	EMPLOYEE
DBResp_department	[0..1]	DBRESP
Department	[0..1]	DEPARTMENT

13. Connect the inputs/outputs as follows:

- EmployeeResponse/DBResp ---> DBResp_employee
- EmployeeResponse/Employee/Item ---> Data/Employee
- DepartmentResponse/DBResp ---> DBResp_department
- DepartmentResponse/Department/Item --> Data/Department

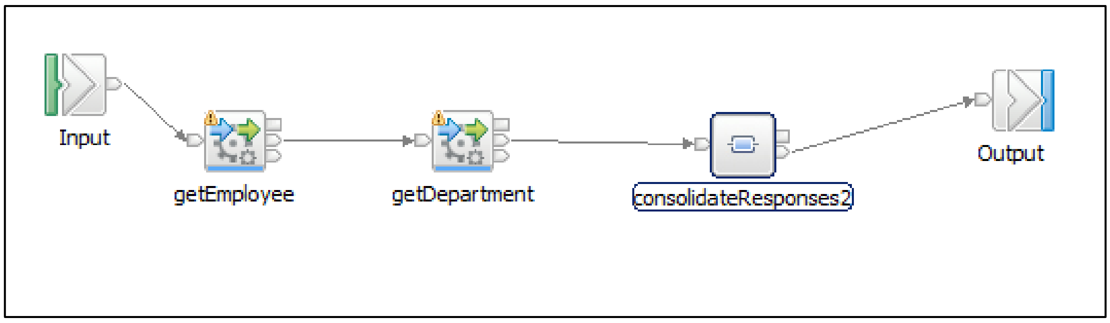


Variables		_EnvironmentVariablesType
<Click to filter...>		
choice of cast items	[0..*]	
any	[1..1]	
EmployeeResponse	[1..1]	EmployeeResponse
DBResp	[0..1]	DBRESP
Employee	[0..1]	JSONArray_Employee
Item	[0..*]	EMPLOYEE
DepartmentResponse	[1..1]	DepartmentResponse
DBResp	[0..1]	DBRESP
Department	[0..1]	JSONArray_Department
Item	[0..*]	DEPARTMENT

Data		DetailedResponse
<Click to filter...>		
DBResp_employee	[0..1]	DBRESP
Employee	[0..1]	EMPLOYEE
DBResp_department	[0..1]	DBRESP
Department	[0..1]	DEPARTMENT

Save and close the map.

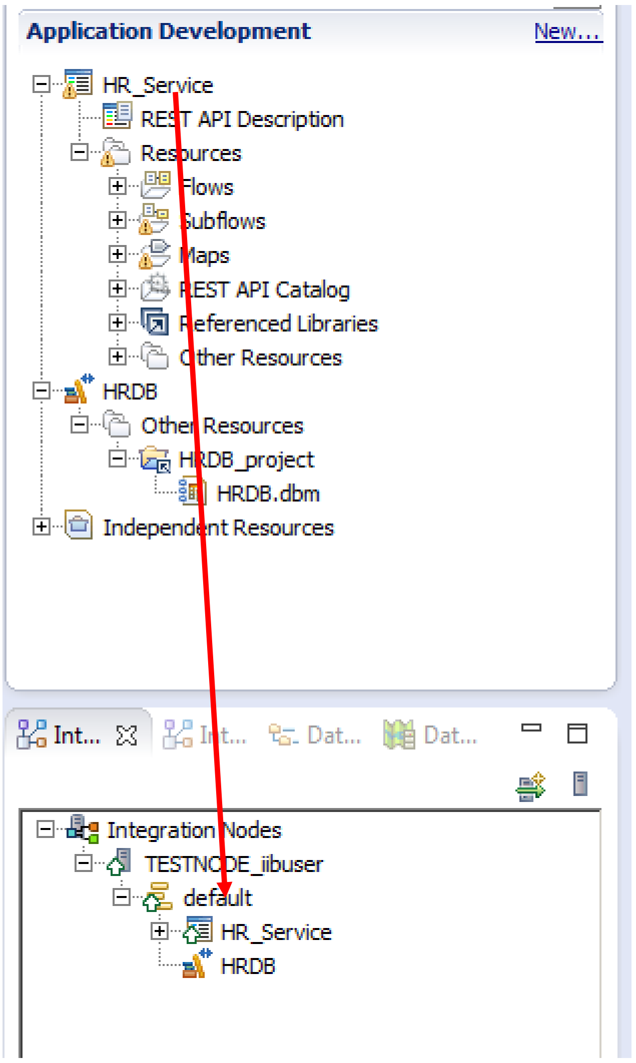
The subflow should now look like this. Make sure all connections have been made as shown.



Save the updated subflow.

2.2 Deploy and test the updated REST API

1. Deploy the HR_service_REST API onto the default server.



2. Use a simple browser request to test the new operation.

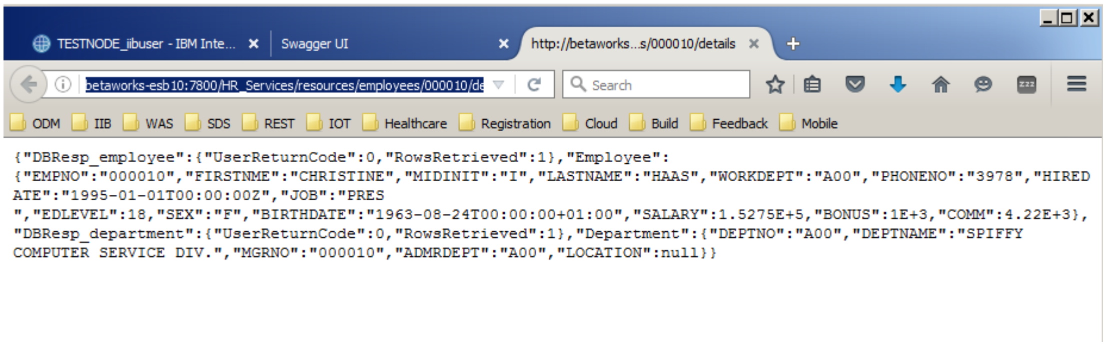
   Use the following URL:

   http://betaworks-esb10:7800/HR_Services/resources/employees/000010/details

   (Ensure the hostname and port parts of the URL are set correct for your own system).

   On the provided workshop VMWare image, this URL is bookmarked in the REST folder.

   The response message will look something like this. Note that the response message contains both the Employee and Department data for the specified employee (000010 in this example).



TESTNODE_iibuser - IBM Inte...
Swagger UI
http://betaworks...s/000010/details
betaworks-esb10:7800/HR_Services/resources/employees/000010/de
Search
ODM
IIB
WAS
SDS
REST
IOT
Healthcare
Registration
Cloud
Build
Feedback
Mobile

```
{"DBResp_employee":{"UserReturnCode":0,"RowsRetrieved":1},"Employee":
{"EMPNO":"000010","FIRSTNME":"CHRISTINE","MIDINIT":"I","LASTNAME":"HAAS","WORKDEPT":"A00","PHONENO":"3978","HIRED
ATE":"1995-01-01T00:00:00Z","JOB":"PRES
","EDLEVEL":18,"SEX":"F","BIRTHDATE":"1963-08-24T00:00:00+01:00","SALARY":1.5275E+5,"BONUS":1E+3,"COMM":4.22E+3},
"DBResp_department":{"UserReturnCode":0,"RowsRetrieved":1},"Department":{"DEPTNO":"A00","DEPTNAME":"SPIFFY
COMPUTER SERVICE DIV.","MGRNO":"000010","ADMRDEPT":"A00","LOCATION":null}}
```

END OF LAB GUIDE