L97

IBM Integration Bus

Implementing Analytics with the R Node using REST API

Featuring:

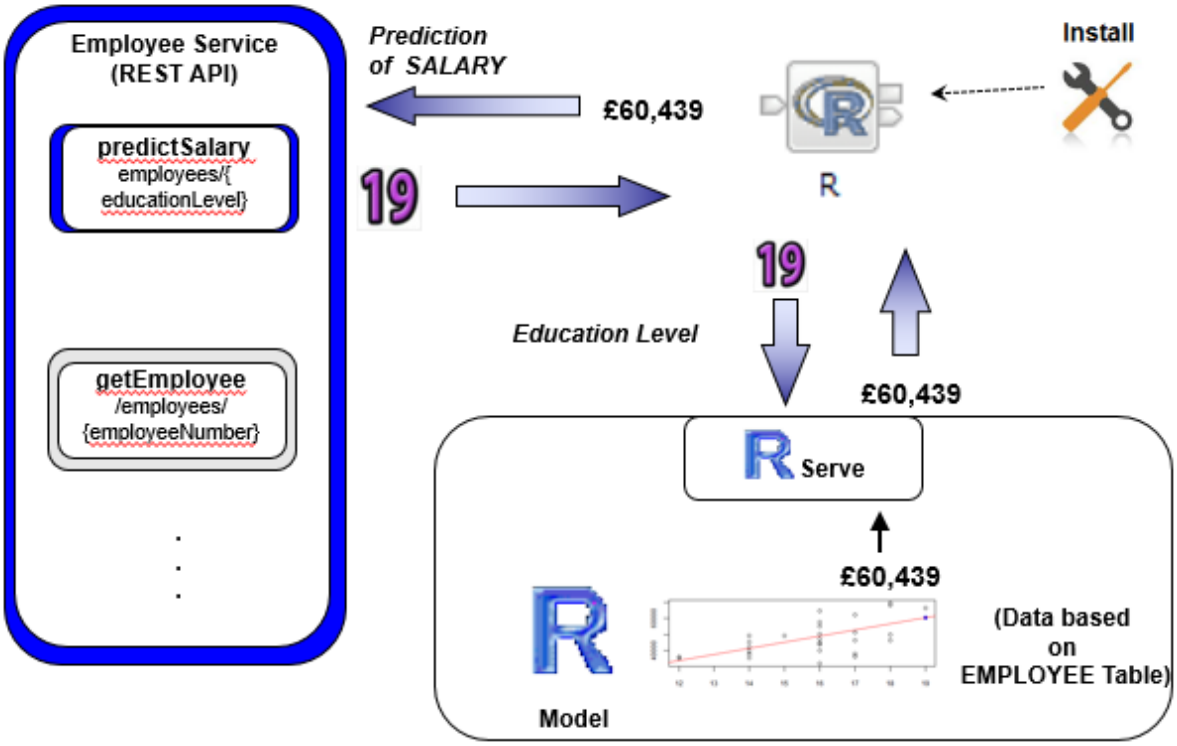
Analytics using R
REST API

June 2015
Hands-on lab built at product
Version 10.0.0.0

Contents

1. Introduction

This lab guide provides instructions on how to set up an R node in IIB and use it in a REST service. Data was extracted from the EMPLOYEE table in the DB2 SAMPLE database and an R model was created with the data. The simple application that you will create, will predict the value of Salary given a value for Education level. The following diagram outlines at a high level what you will do:



This lab guide shows you how to do the following tasks:

1) How to add R node support to your environment using the R node IIB Extension.
2) How to configure a simple message flow containing an R node using a REST service.
3) How to configure an R node in a message flow.
4) How to test a message flow with an R node and interface with R serve.
5) How to understand the number returned from R.

2. Prepare the Environment

2.1 Create and configure a new node for REST

The IIB support for the REST API requires some special configuration for the IIB node and server. To ensure this scenario does not interfere with other scenarios, you will use a separate IIB node.

1. **Do this section only if RESTNODE, a dedicated IIB node for REST API, does not already exist on your system.**

   Support for REST requires the use of the embedded HTTP listeners, rather than the node-wide listener.

   Additionally, the IIB node has to be configured for Cross-Origin Resource Sharing (CORS). For details on this, please read: http://www.w3.org/TR/cors/.

   To configure these items, we have provided a script. In an IIB Command Console, change directory to c:\student10\REST_service\install.

   Run the script file:

   ```
   Create_RESTNODE
   ```

   Accept the default values for the IIB node name (RESTNODE).

   This script will create the new node, and run two key commands:

   - Enable HTTP embedded listeners. The REST support is only provided for embedded listeners, not the node-wide listener:

     ```
     mqsichangeproperties RESTNODE
           -e default
           -o ExecutionGroup
           -n httpNodesUseEmbeddedListener -v true
     ```

   - Enable Cross-Origin Resource Scripting for REST. This is required when testing with the SwaggerUI test tool. See http://en.wikipedia.org/wiki/Cross-origin_resource_sharing for further information.

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

   The script will also configure the JDBC parameters for connection to the SAMPLE database, which will be used in this scenario.

   When the node is fully restarted, you will see the following messages on the IIB Event Log viewer:

   ```
   BIP3132I: ( RESTNODE ) The HTTP Listener has started listening on port
   ''4418'' for ''WebAdmin http'' connections. [08/04/2015 09:16:21]
   BIP2152I: ( RESTNODE.default ) Configuration message received from
   integration node. [08/04/2015 09:16:22]
   BIP2153I: ( RESTNODE.default ) About to ''Start'' an integration server.
   [08/04/2015 09:16:22]
   BIP2154I: ( RESTNODE.default ) Integration server finished with
   Configuration message. [08/04/2015 09:16:22]
   ```

2.2 Install the IIB R Node

The Analytics node "R" is not part of the standard IIB V10 toolkit and runtime installation. The R node is being made available as an IIB extension from GitHub and is available from here:

https://github.com/ot4i/integrate-R.

In order to do this lab, you will need to install the R node. Two files need to be copied into the IBM Integration Bus installation. They are in `C:\student10\Analytics\Resources` and are:

- A toolkit component (currently `RNodeToolkit_1.0.0.20150417-1641.jar`): to be copied to `C:\IBM\IIB\10.0.0.0\tools\plugins.` (Once this has happened, the Integration Toolkit must be restarted).

- A runtime component (currently `RNodeRuntime-1.0.0.20150417-1641.par`): to be copied to `C:\IBM\IIB\10.0.0.0\server\bin.` (Once this has happened, any integration servers to which the R node will be deployed, must be restarted).

You can do this manually, if you wish. Alternatively, run a script we written for you, as follows:

1. Open an IIB Console and navigate to "`C:\student10\Analytics\Commands`" and run the command `RnodeInstall.cmd`

2. The script defaults to copying the files named above from `C:\student10\Analytics\Resources` to their respective destinations.
   In addition, it stops and starts IB10NODE by default.
   Change this to RESTNODE and accept the other defaults by pressing Enter.

3. The output from the script should be like this:

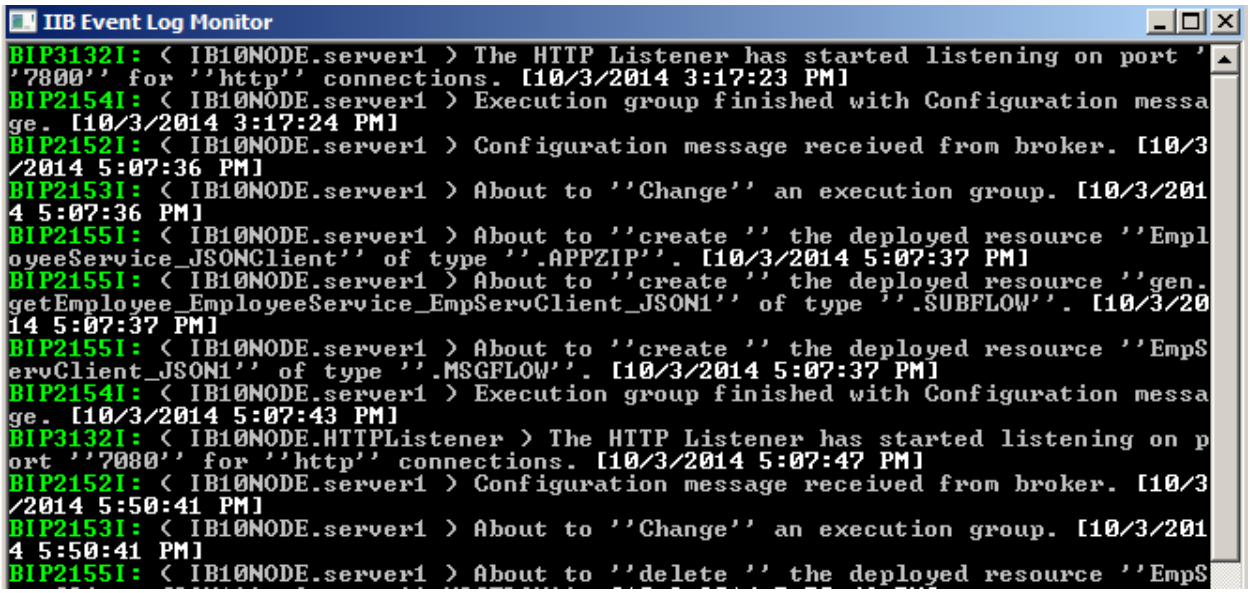
```
C:\student10\Analytics\Commands>RnodeInstall.cmd
This command file must be run within an Integration Bus Command Console.
Betaworks Analytics Lab guide Rnode setup.
Enter IIB Node name (default is IB10NODE): RESTNODE
Enter R Node runtime file name (default is RNodeRuntime-1.0.0.20150417-1641.par)
:
Enter R Node runtime from folder (default is C:\student10\Analytics\Resources):

Enter R Node runtime to folder (default is C:\IBM\IIB\10.0.0.0\server\bin):
Enter R Node toolkit file name (default is RNodeToolkit_1.0.0.20150417-1641.jar)
:
Enter R Node toolkit from folder (default is C:\student10\Analytics\Resources):

Enter R Node toolkit to folder (default is C:\IBM\IIB\10.0.0.0\tools\plugins):
-
Thankyou, using values "IB10NODE", "RNodeRuntime-1.0.0.20150417-1641.par", "C:\s
tudent10\Analytics\Resources", "C:\IBM\IIB\10.0.0.0\server\bin", "RNodeToolkit_1
.0.0.20150417-1641.jar", "C:\student10\Analytics\Resources", "C:\IBM\IIB\10.0.0.
0\tools\plugins"
 Ok to proceed? Use Ctrl-C to terminate.
Press any key to continue . . .
-
Stopping node "RESTNODE"
BIP8019E: Integration node 'RESTNODE' stopped.
This integration node is stopped; the command you issued cannot be processed whe
n an integration node is stopped.
A previous command has been issued to stop this integration node, or this integr
ation node has never been started.
This integration node can be started, changed, or deleted.
-
About to remove R Node code from IIB. Failure messages can be ignored.
Could Not Find C:\IBM\IIB\10.0.0.0\server\bin\RNodeRuntime-1.0.0.20150417-1641.p
ar
Could Not Find C:\IBM\IIB\10.0.0.0\tools\plugins\RNodeToolkit_1.0.0.20150417-164
1.jar
About to copy across R Node code.
Ok to proceed? Use Ctrl-C to terminate.
        1 file(s) copied.
        1 file(s) copied.
-
About to Start "RESTNODE"
BIP8096I: Successful command initiation, check the system log to ensure that the
 component started without problem and that it continues to run without problem.
```

4. Check that the files have been copied across and the integration node has been started.

2.3 Open the Windows Log Monitor for IIB

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

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

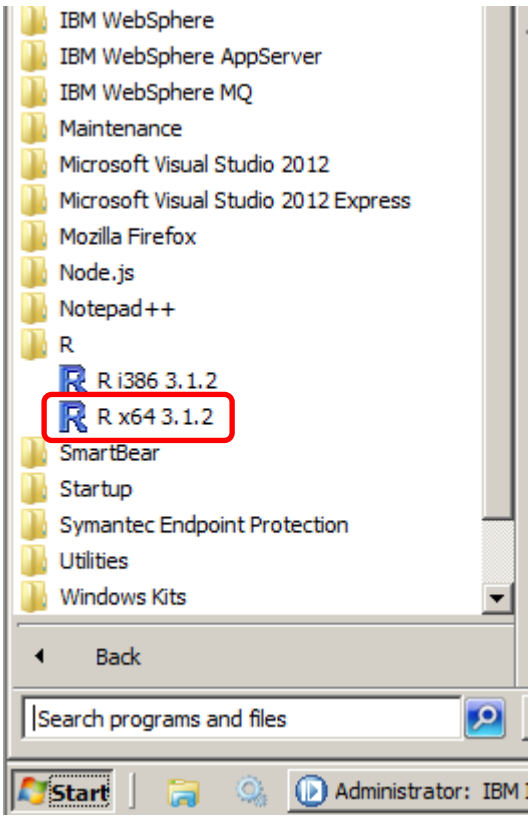


This tool is not shipped as part of the IIB product; please contact us directly if you would like a copy.

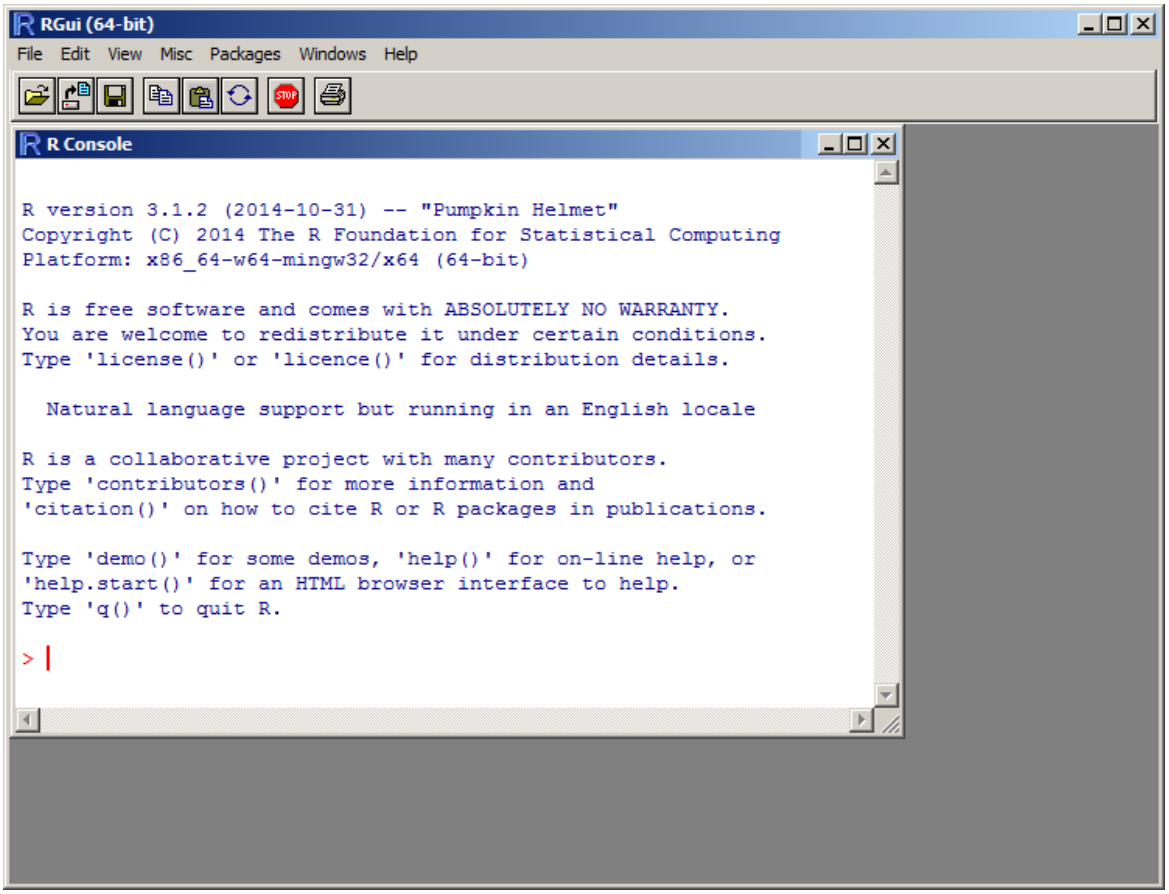
2.4 Start R server

In order to run a message flow with an R node, the R server needs to be started. In this section you will start R Server from the R GUI.

1. From the Windows Start menu start the R Gui, (click **Start > All programs > R > "R x64 3.1.2")**



2. The **RGui(64-bit)** with an **R Console** will open:



3. In the “R console” type:

```
library(Rserve)
```

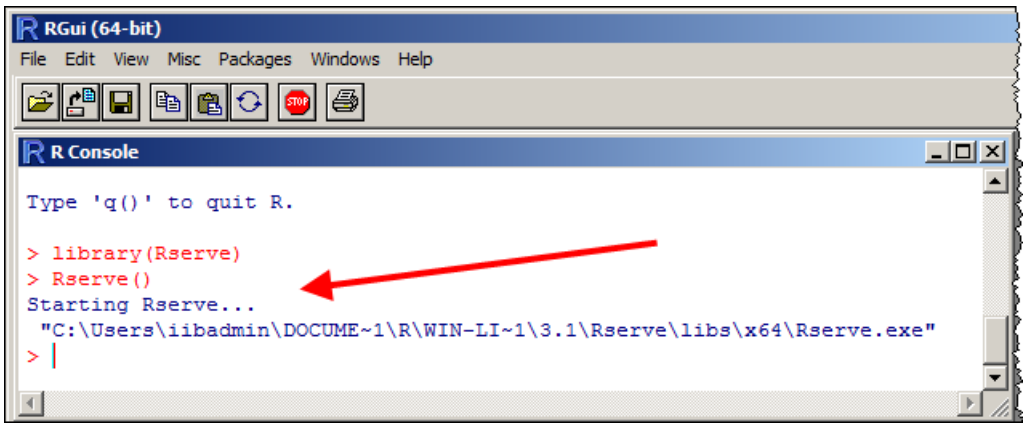
and press enter.

4. In the same console type:

```
Rserve()
```

and press enter.

The R Console will look like this:



The IIB R node can now use Rserve to interface with R.

2.5 Import pre-requisites

1. First, to ensure there are no conflicts with other components, switch to a new IIB workspace in the IIB Toolkit, for example name it c:\workspaces\IIB_R_REST.

2. In the IIB Toolkit, import these Project Interchange files:
   c:\student10\Integration_service\solution\EmployeeServiceInterface.V10.zip and
   c:\student10\Integration_service\solution\ EmployeeService.V10.zip
   c:\student10\REST_service\solution\EmployeeService_REST.V10.zip

3. Update REST API to call R

3.1 Examine predictSalary operation in Swagger document (optional)

In Windows Explorer, locate the file:

c:\student10\REST_service\resources\EmployeeService.json

Open the file with the Notepad++ editor (right-click, select Edit with Notepad++).

We have installed a JSON document plugin into Notepad++, so this JSON document will be formatted for easy reading.

You will see a section defining the analytics operation. Key details are highlighted below.

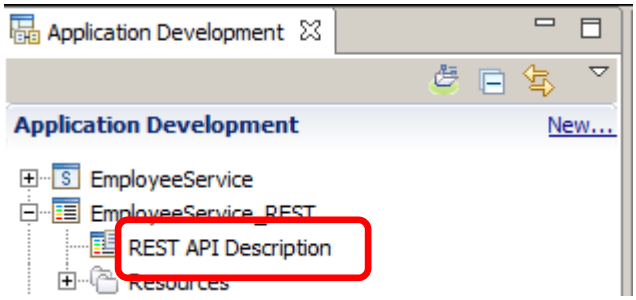
```
            "/employees/{educationLevel}/predictSalary": {
                  "get": {
                        "tags": ["employees"],
                        "summary": "Retrieve the predicted salary for an
employee",
                        "description": "Retrieve the predicted salary
for an employee",
                        "operationId": "predictSalary",
                        "produces": ["application/json"],
                        "parameters": [{
                              "name": "educationLevel",
                              "in": "path",
                              "description": "The educationLevel of the
employee",
                              "required": true,
                              "type": "integer"
                        }],
                        "responses": {
                              "200": {
                                    "description": "OK"
                              },
                              "500": {
                                    "description": "Something wrong in
Server"
                              }
                        }
                  }
            }
```

Key details here are:

- a new path, *employees/{educationLevel}/…* to reflect that education level is passed as a parameter
- suitable summary and description
- a new operation ID, *predictSalary* and associated description
- education level as a parameter
- parameter type as an integer

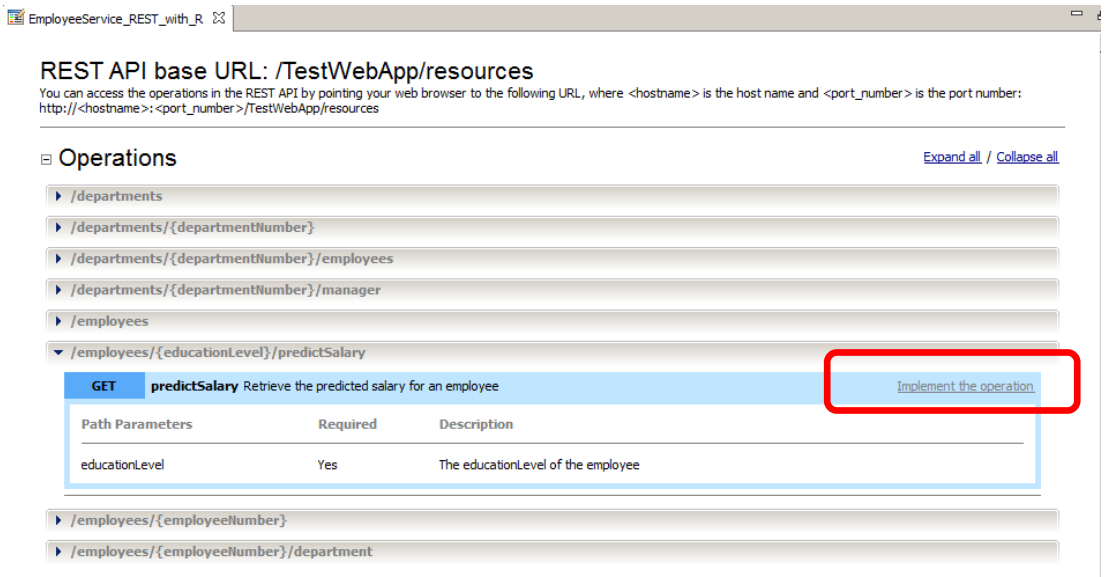
3.2 Implement the predictSalary operation

1. Double click on 'REST API Description' under EmployeeService_REST in the Application Development pane.



2. In the Operations list, locate predictSalary and click "Implement the operation".

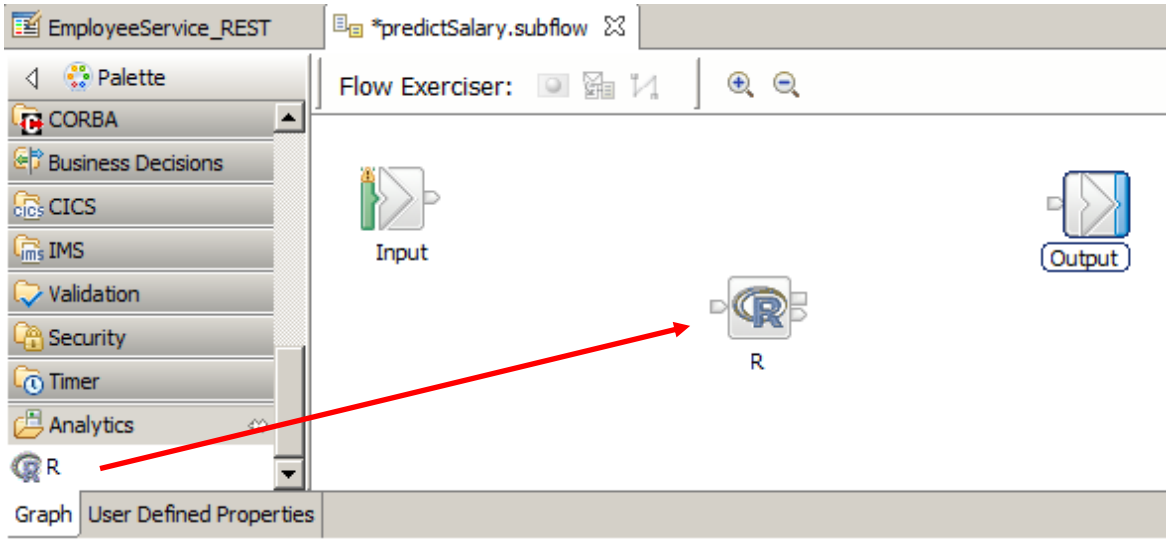
   This will open the subflow editor. Each operation is implemented with a separate subflow.



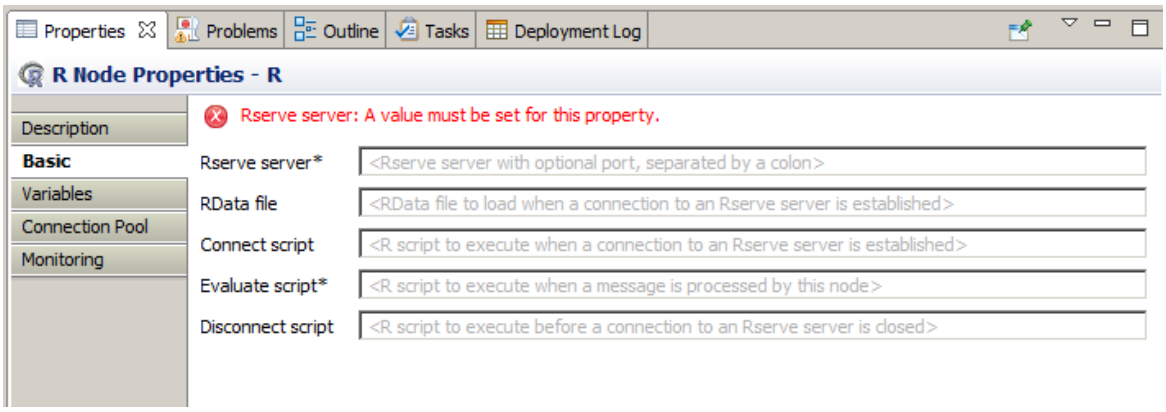
3.3 Configure an R Node

3.3.1 Configure the Basic properties

1. Open the Analytics folder in the message flow palette, and drag the R node onto the canvas:



2. Highlight the R node so that the properties tab shows the R node properties:



3. Configure the R node properties as follows:

Basic Tab:

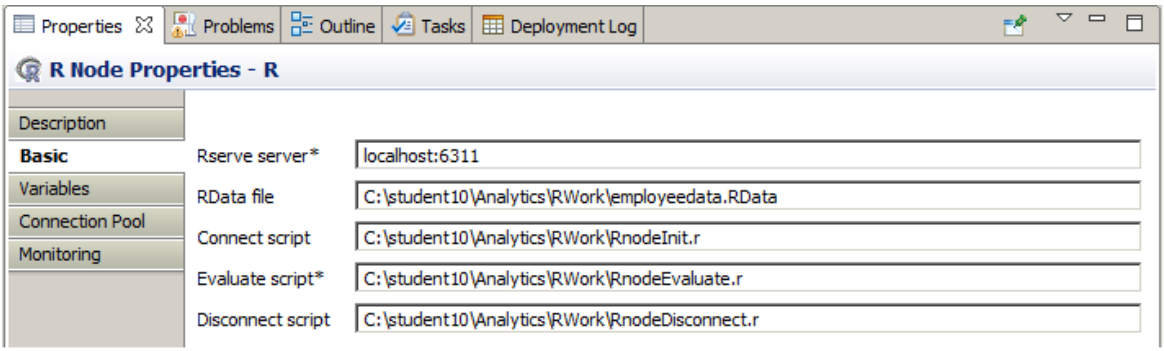
Rserve server : **`localhost:6311`**

RData file: **`C:\student10\Analytics\RWork\employeedata.RData`**

(note:* in a distributed environment *- the RData file needs to exist on the same machine as IIB toolkit)

Connect script: **`C:\student10\Analytics\RWork\RnodeInit.r`**
Evaluate script: **`C:\student10\Analytics\RWork\RnodeEvaluate.r`**
Disconnect script: **`C:\student10\Analytics\RWork\RnodeDisconnect.r`**

(note:* in a distributed environment- *the scripts need to exist on the same machine as IIB Run time environment, they also reference locations on the server where R is installed)



3.3.2 Configure Variables

The Variables tab in the R node is where you configure the variables to be used with your R environment. A Data Frame, in R, is a used to store a collection of data in rows and columns, (similar to the concept of a matrix in mathematics, however the types of a data frame do not necessarily need to be numeric).

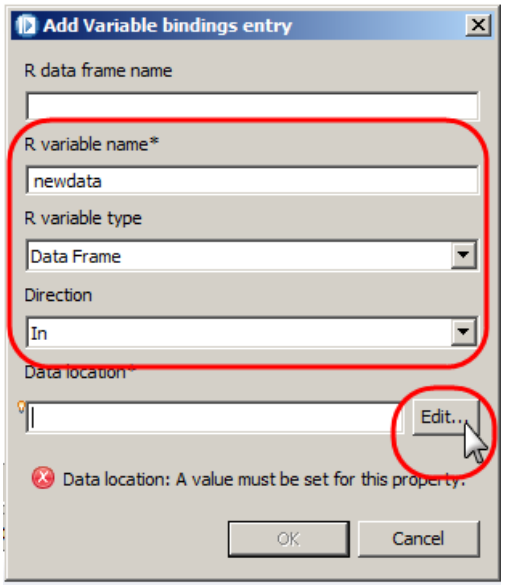
In the following section you will define the variables that will be used in R and how IIB will use them.

1. **Variables Tab:**

   Click the “Add…” button to add a variable, the “Add Variable bindings entry” window will appear: Specify the following (case sensitive):

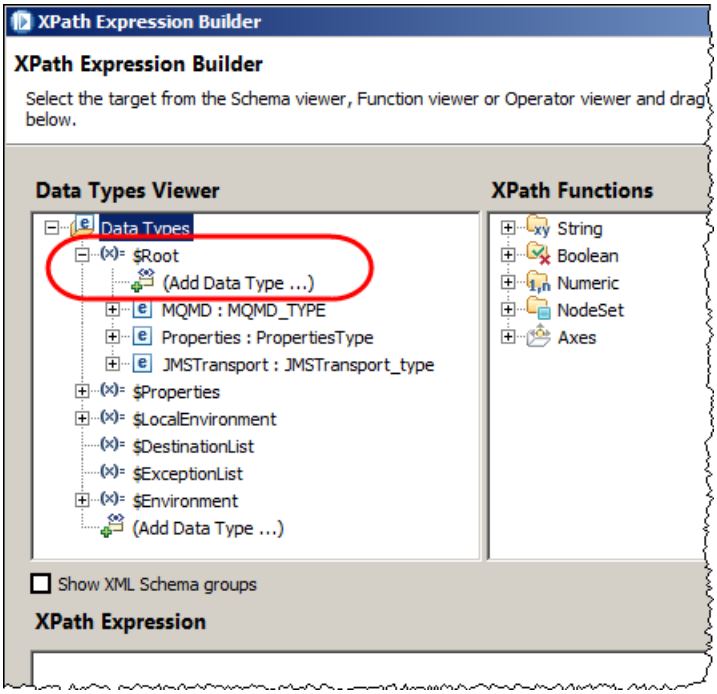
   R variable name: **newdata**
   R variable type : **Data Frame**
   Direction : **In**

   For Data location click the Edit button:

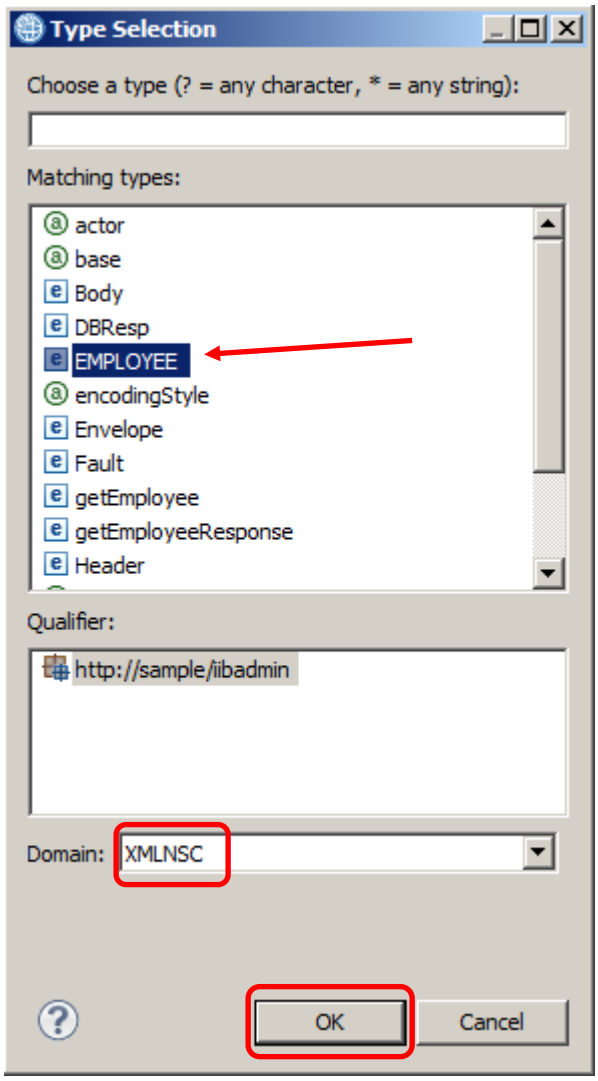


2. The XPath Expression Builder will open.

   In the **Data Types Viewer** section, expand **$Root** and click "**(Add Data Type)**"



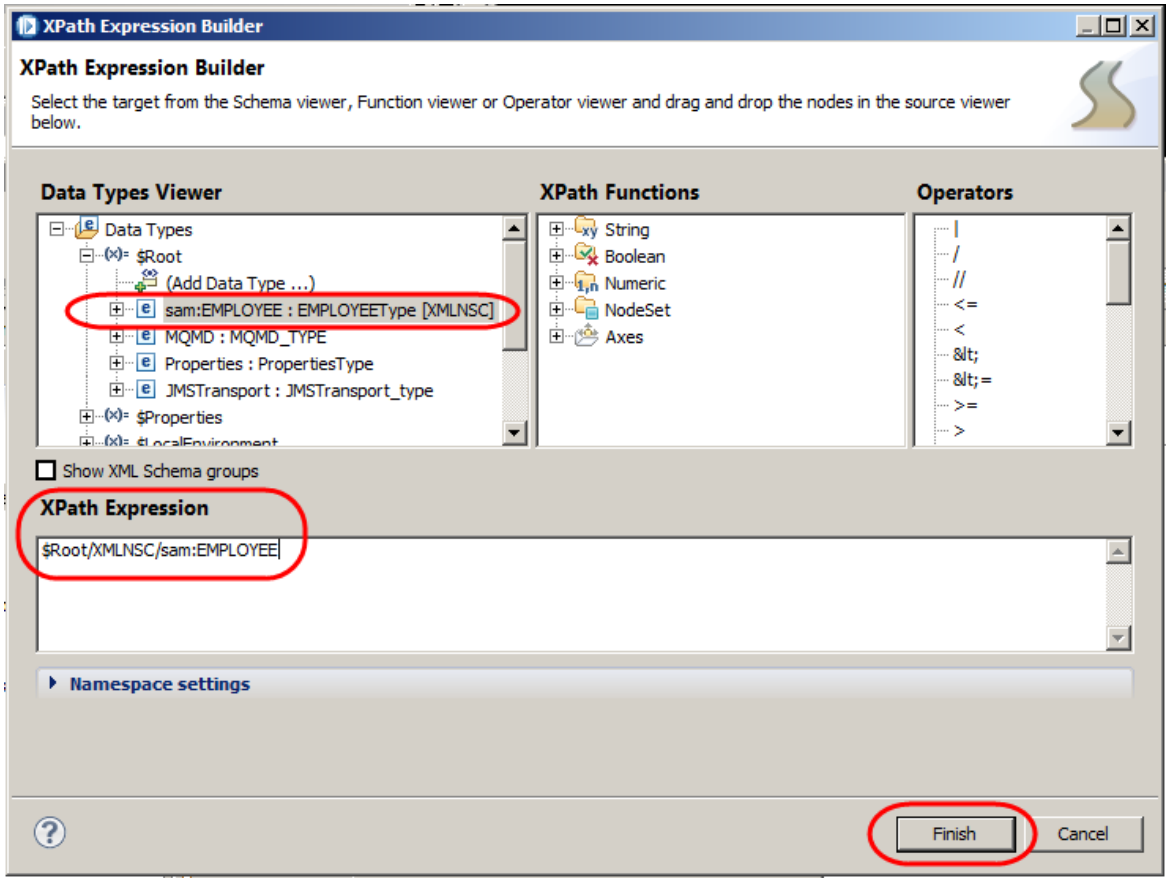
3. In the "**Type Selection**" window,
   1) highlight **EMPLOYEE**
   2) Change the Domain to **`XMLNSC`**
   3) and click **OK**:



4. The EMPLOYEE data type appears under $Root.

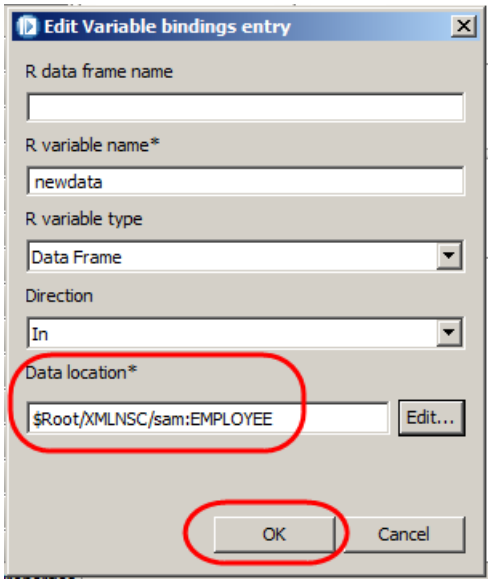
   Double click on **sam:EMPLOYEE:EMPLOYEEType [XMLNSC] to set the value in the XPath Expression**

   Click Finish to save the expression:

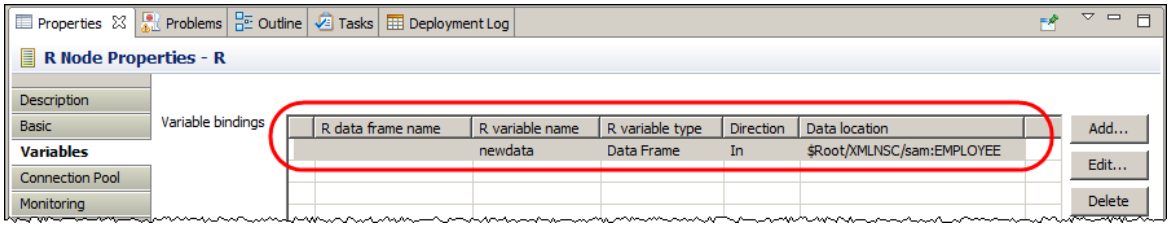


5. The Data location field will be automatically completed with the XPath expression.

   Click OK to add the variable:



6. The variable bindings table will be updated with the definition for the newdata “Data Frame”:



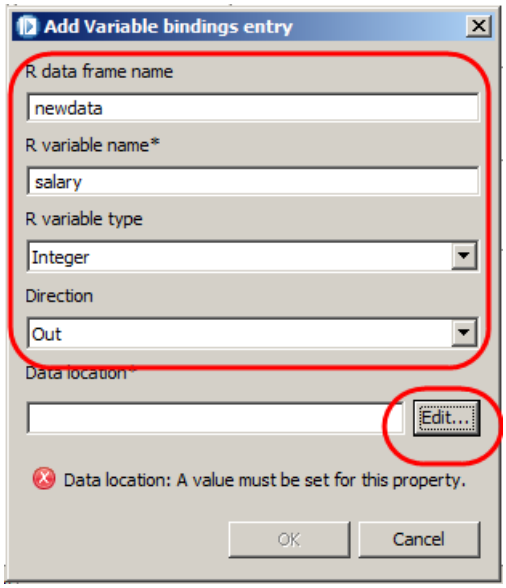
7. Click the Add button to add a second variable:

Click the “Add…” button to add a **second variable**, the “Add Variable bindings entry” window will appear.

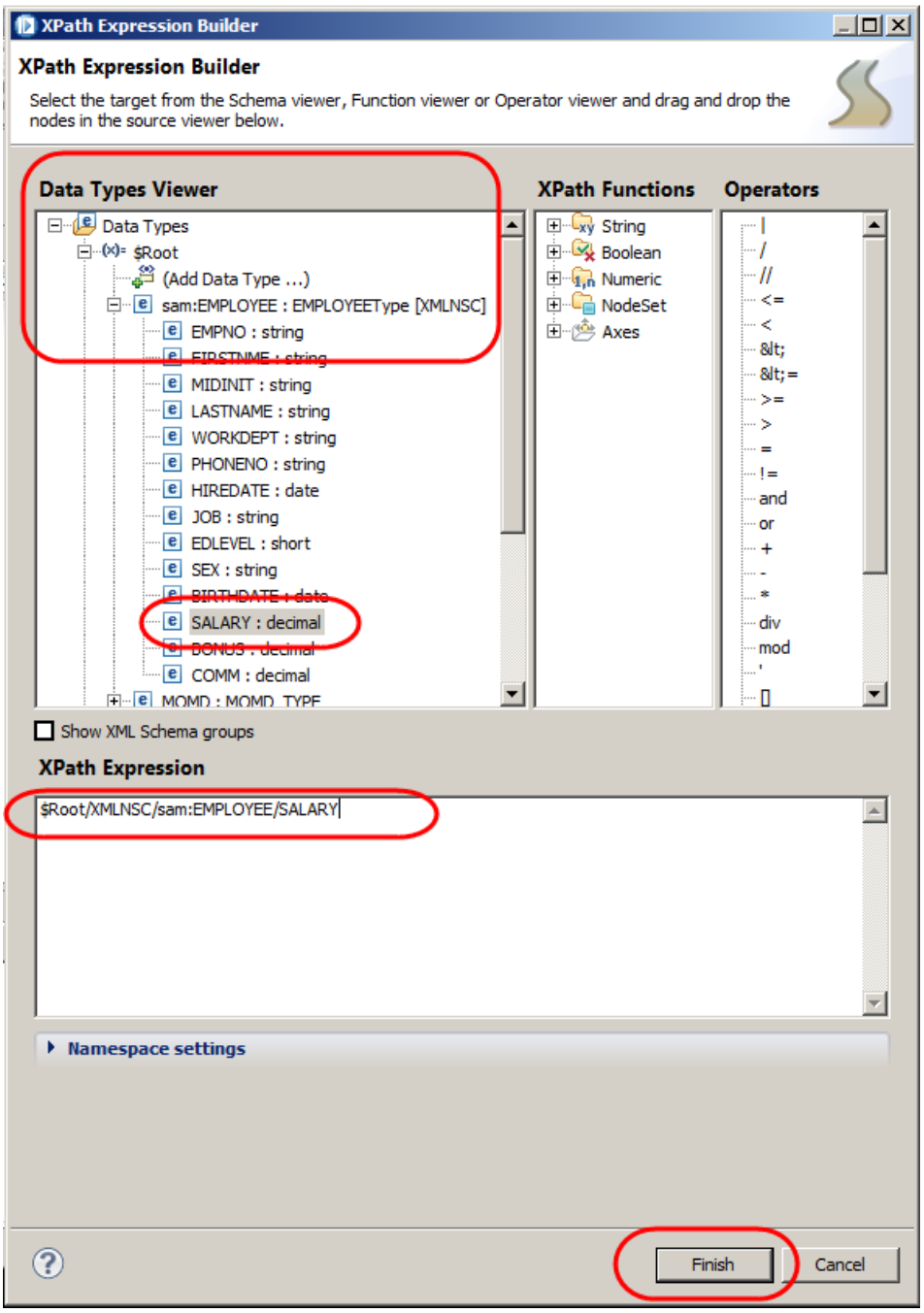
Specify the following (case sensitive):

R data frame name : **newdata**
R variable name : **salary**
R variable type : **Integer**
Direction : **Out**

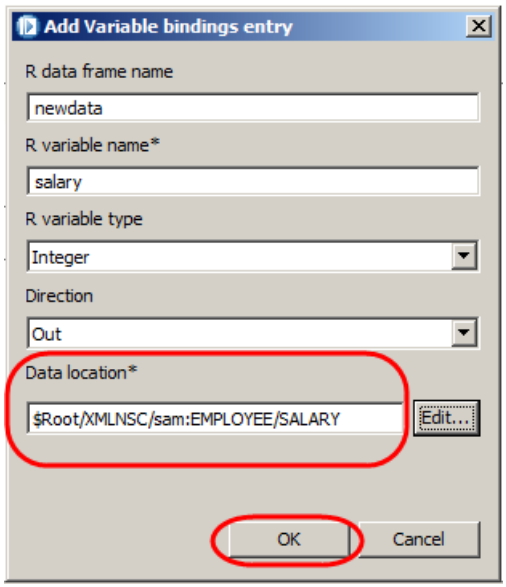
For Data location click the Edit button:



8. In the XPath Expression Builder expand $Root until you can select SALARY. Double click on Salary to formulate the expression. Click Finish:



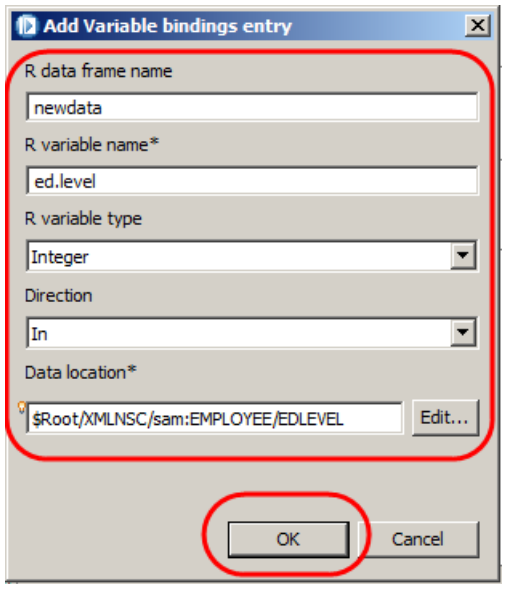
9. The Data location will automatically be filled with the location of SALARY in the message tree, Press OK to save the variable:



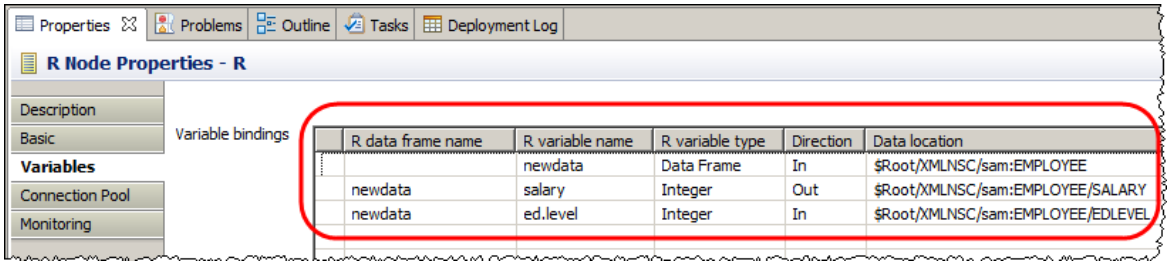
10. Click the "Add…" button to add a **third variable**, using the same method you used for the previous two variables.

Specify the following (case sensitive):

R data frame name: **newdata**
R variable name : **ed.level**
R variable type : **Integer**
Direction : **In**
Data location : **$Root/XMLNSC/sam:EMPLOYEE/EDLEVEL**



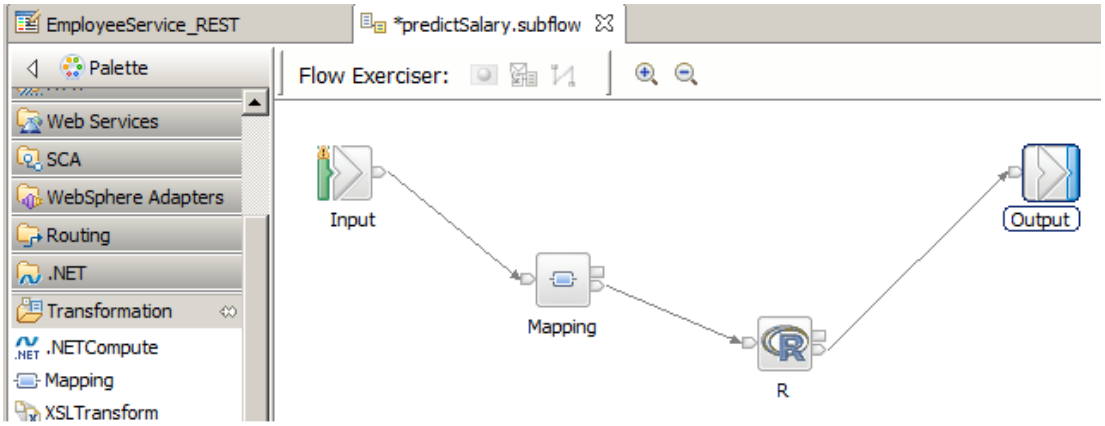
11. The variables table will look like the following when all required variables for the model we are using are defined:



12. Save the message flow <ctrl s>, keeping the message flow open in the Integration Toolkit.

3.4 Add Mapping Node

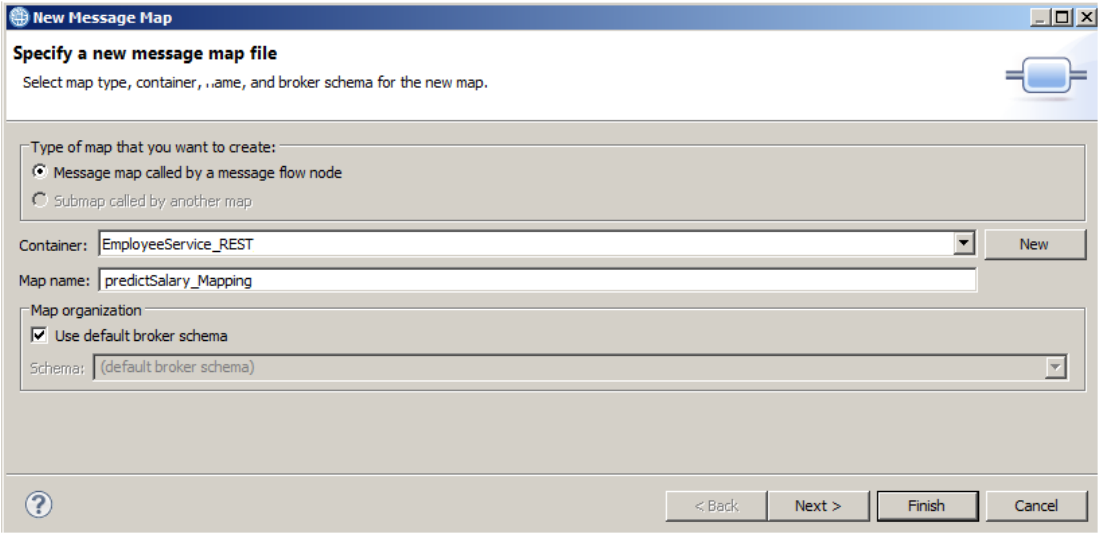
1. Drop a Mapping node onto the flow editor and connect as shown.



REST GET operations will be contained within the HTTP header part of the incoming message. When IIB receives such an incoming message, it extracts the REST parameters and places them into the Local Environment. IIB v10 has introduced a new part of the LocalEnv called REST.

The R Node requires its input in the form defined by the EMPLOYEE schema. To construct the message in this format, the new mapping node needs to obtain the incoming parameter for the predictSalary operation. The mapping node will use this information to construct a message in the format required by the R Node.

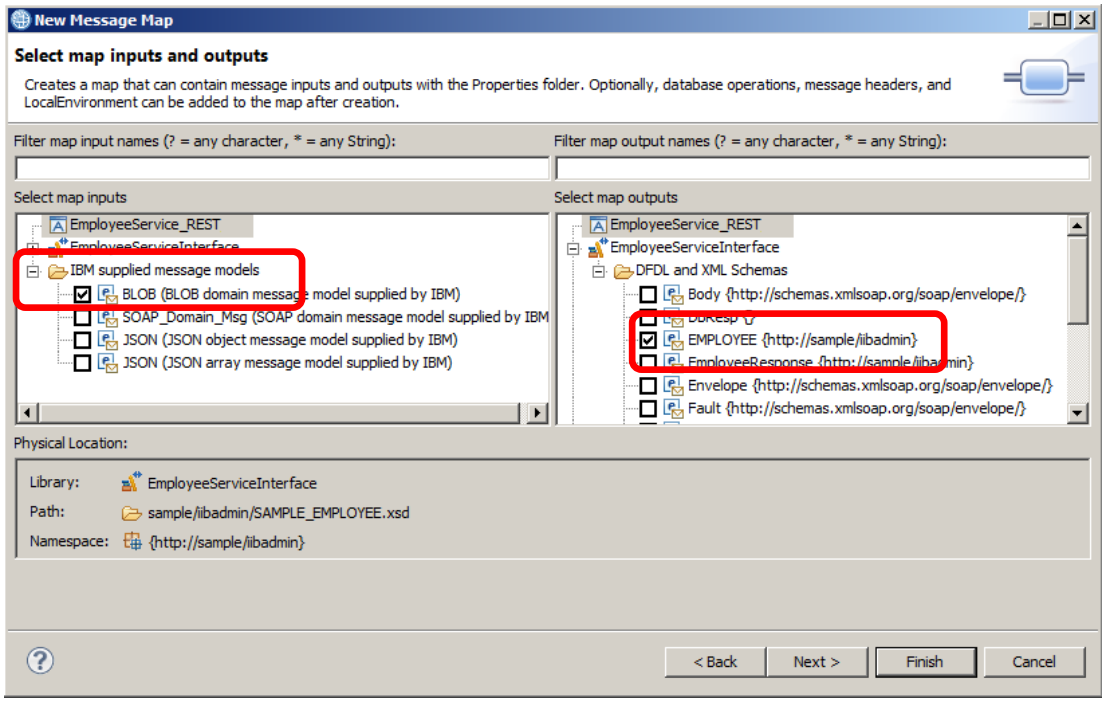
2. Double-click on the mapping node and click Next.



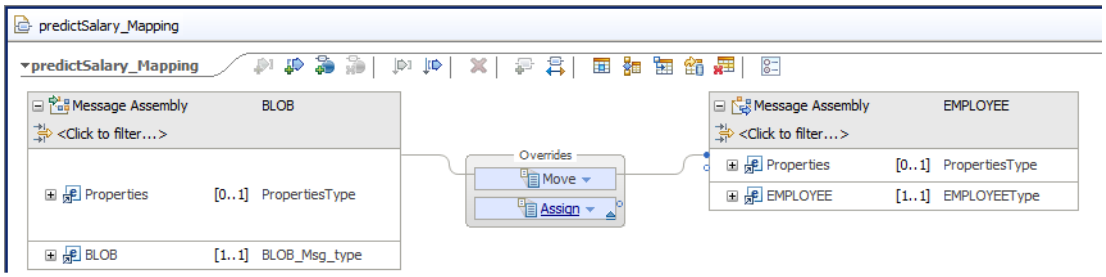
3. For the map inputs and outputs, make the following selections:

   - Input
     - IBM supplied message models - BLOB (the subflow will obtain its information from the LocalEnvironment, and there is no need to parse the message payload)
   - Output
     - EmployeeServiceInterface, XML schemas - EMPLOYEE

   Click Finish.

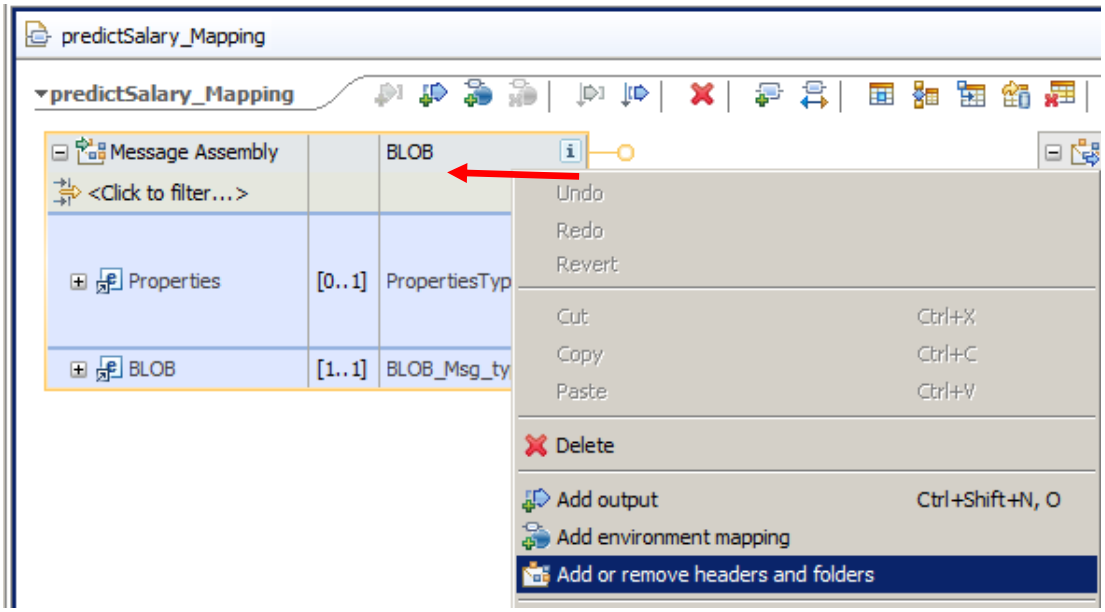


4. The basic mappings will be shown.

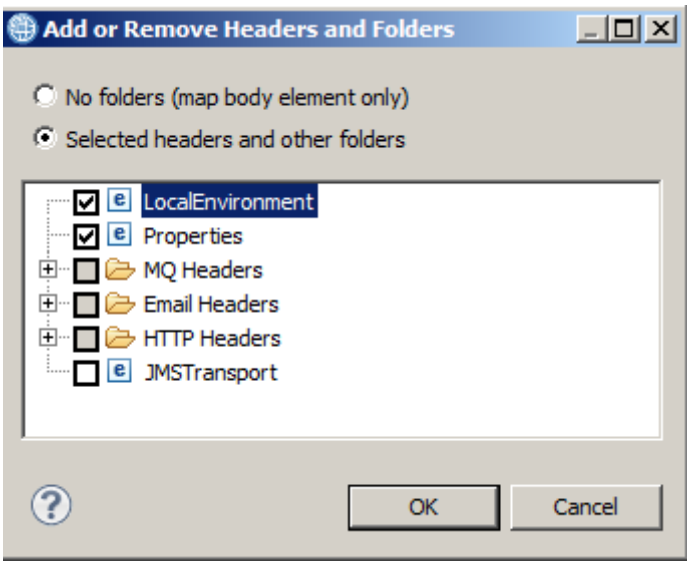


5. For a REST GET operation, the education level will be available in the Local Environment. For the map to access the Local Environment, you must explicitly add this header to the Message Assembly.

   On the input Message Assembly, right-click and select "Add or remove headers and folders".



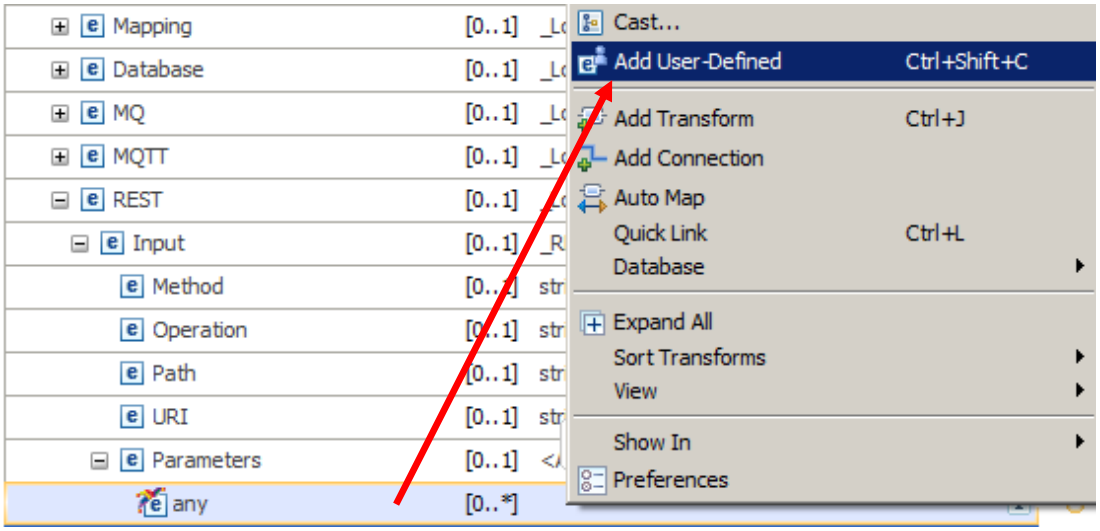
   Select the LocalEnvironment and click OK.



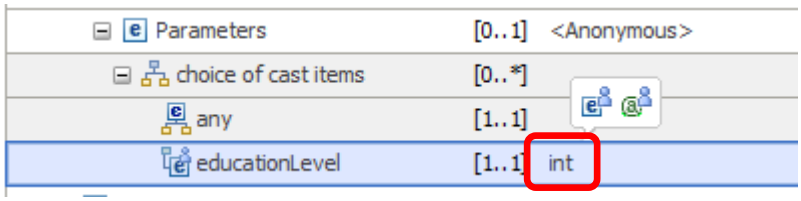
6. Expand the Local Environment and the REST section (located near the bottom of the Local Environment).

   Incoming REST parameters will appear under the REST/Input/Parameters element, so the definition of this element needs to be added here.

   Right-click the "any" element and select "Add User-Defined".

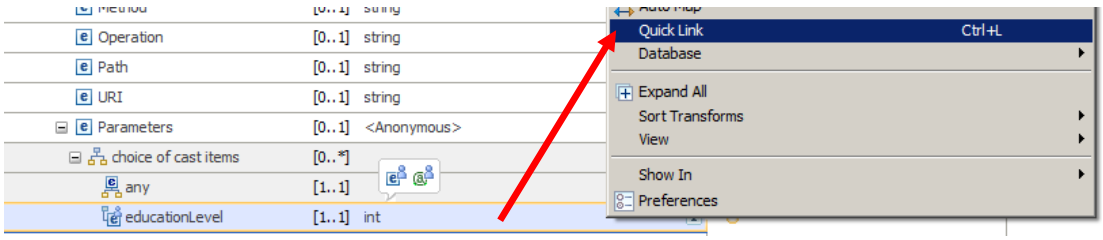


7. Name the new element educationLevel. Note, this has to match the precise name and case of the element in the JSON document. Change the type to *int*, as this takes integer values.



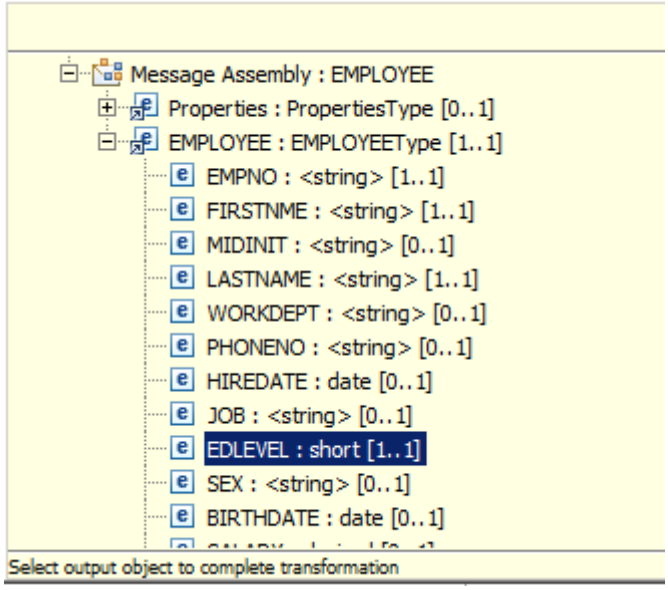
8. The new element needs to be mapped to the output EMPLOYEE/EDLEVEL message.

   Unfortunately, expanding the Local Environment has probably meant that the output message has disappeared from the map display. To handle this, perform a "Quick Link". Right-click educationLevel, and select Quick Link.

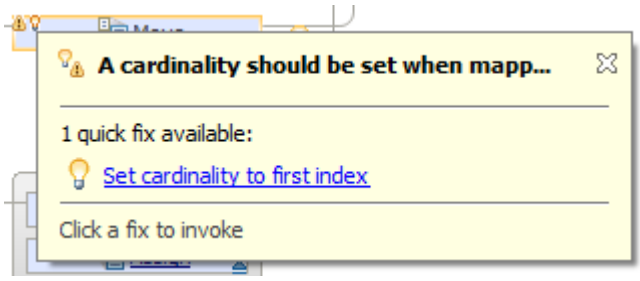


9. A pop-up window will show the available output message.

   Collapse the Properties element, and in the EMPLOYEE element, select EDLEVEL.

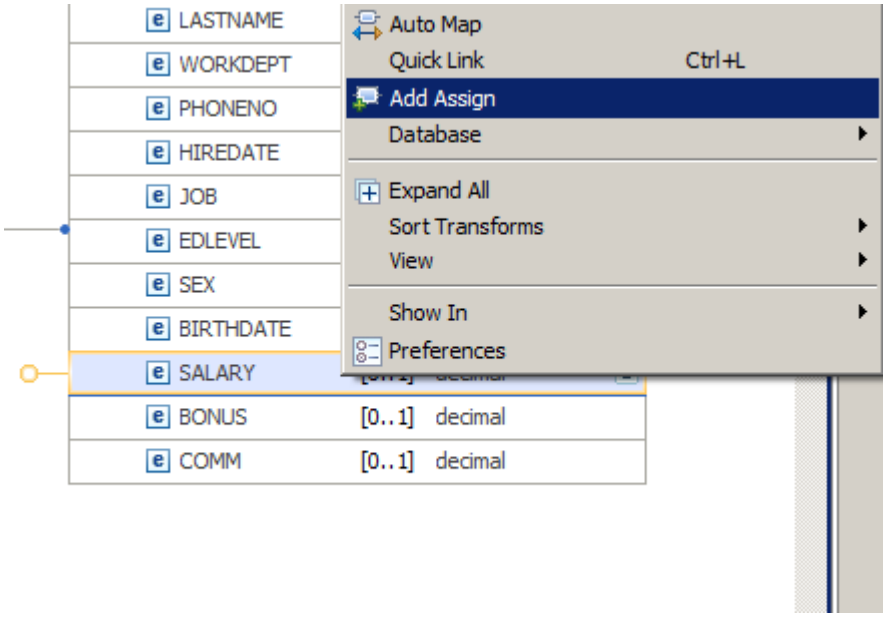


10. A Move transform will be generated. If you wish, you can use the QuickFix to correct the cardinality warning message (hover over the small light on the Move transform, then select "Set cardinality to first index").

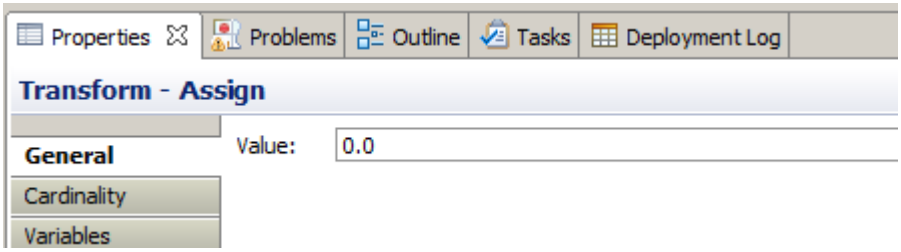


11. The output predicted salary will need to be initialized. This is done by assigning a value of 0.0 to SALARY in the output map.

    Right click SALARY and select 'Add Assign'.



    The Properties tab in the lower pane will show the value has been assigned automatically.



12. Save and close the map.

    Save the subflow.

4. Testing

4.1 Deploy the service

1. In the navigator, deploy the EmployeeServiceInterface shared library to RESTNODE/default.

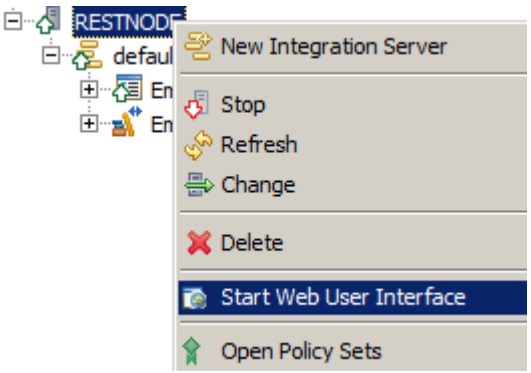
   Then deploy new REST API to RESTNODE/default (drag/drop or right-click, deploy).

4.2 Test the service with the Swagger UI

2. In the navigator, deploy the EmployeeServiceInterface shared library to RESTNODE/default.

   Then deploy new REST API to RESTNODE/default (drag/drop or right-click, deploy).

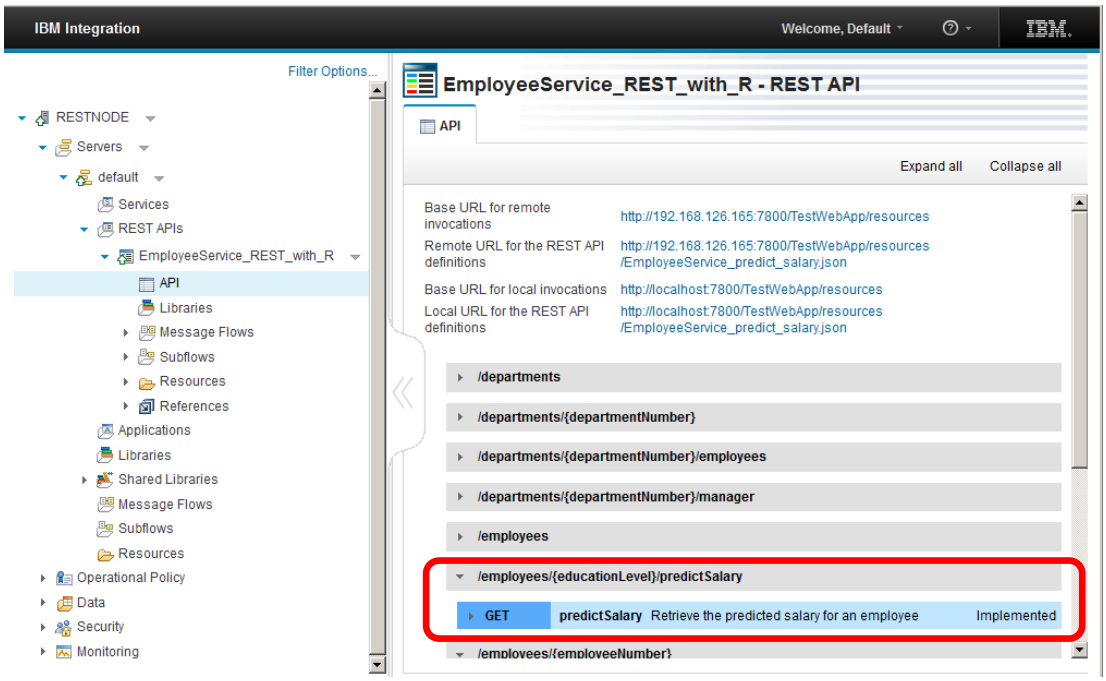
3. Open the IIB web UI by right-clicking RESTNODE and selecting Start Web User Interface.



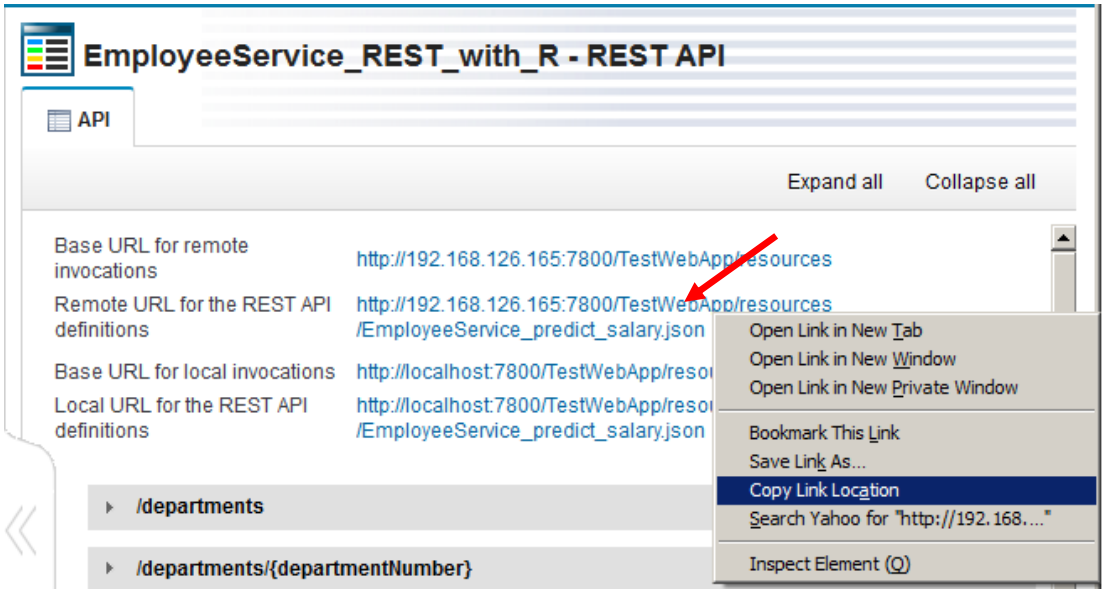
4. You will be switched to the default browser. Fully expand RESTNODE, down to the new REST API as shown below and click the API link under it.

   This will show you the available operations in IIB and whether they have been implemented. Check that you have implemented the correct operation.

   It will also show you the URLs for local and remote invocations, and the REST API definitions (the .json file).

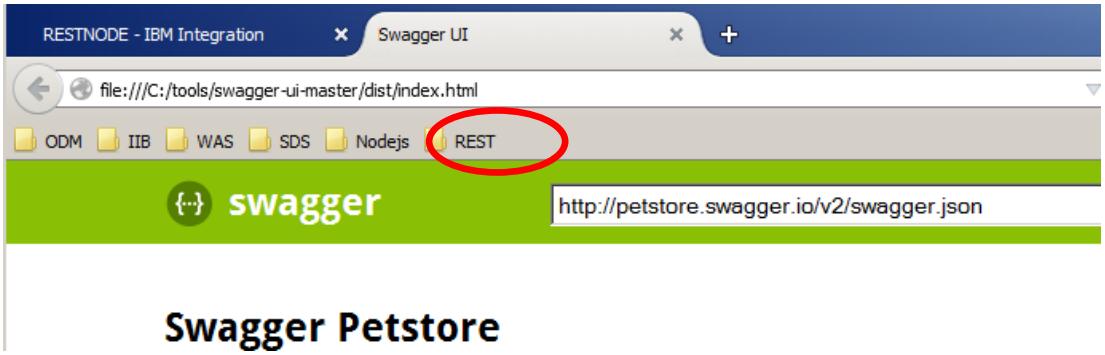


5. On the "Remote URL for REST API definitions", right-click and select "Copy Link Location".



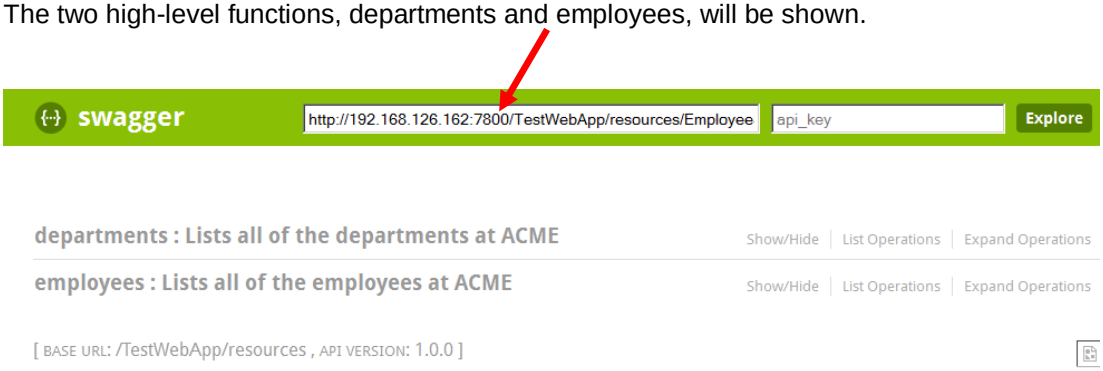
6. In Firefox, open a new tab, and open the SwaggerUI tool (using the bookmark in the REST folder).

   By default, this will open the Petstore Swagger document.



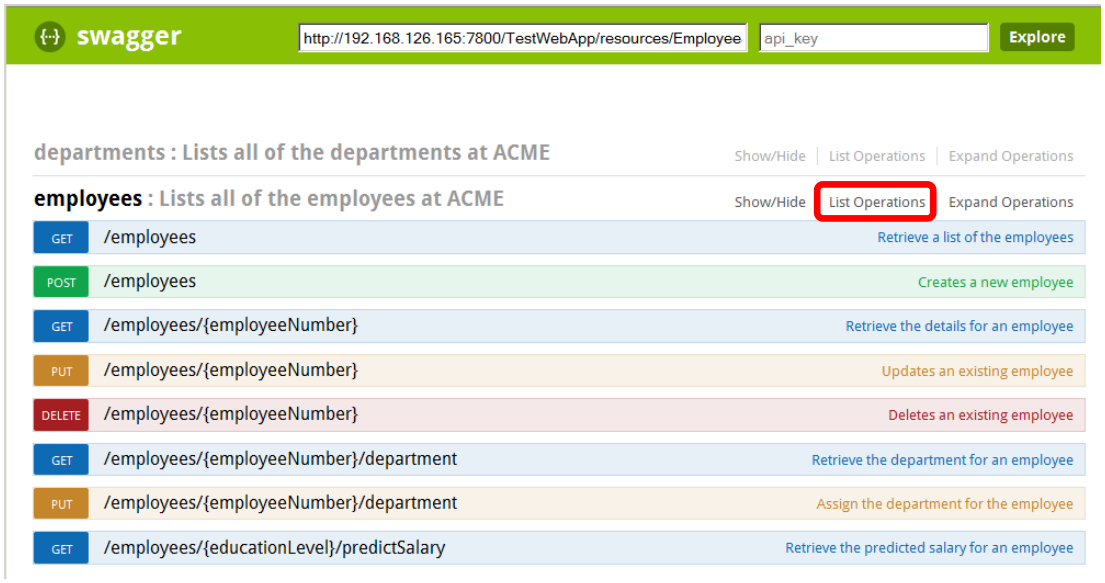
7. In the entry field, paste the contents of the clipboard and click Return (or click Explore).

   The two high-level functions, departments and employees, will be shown.



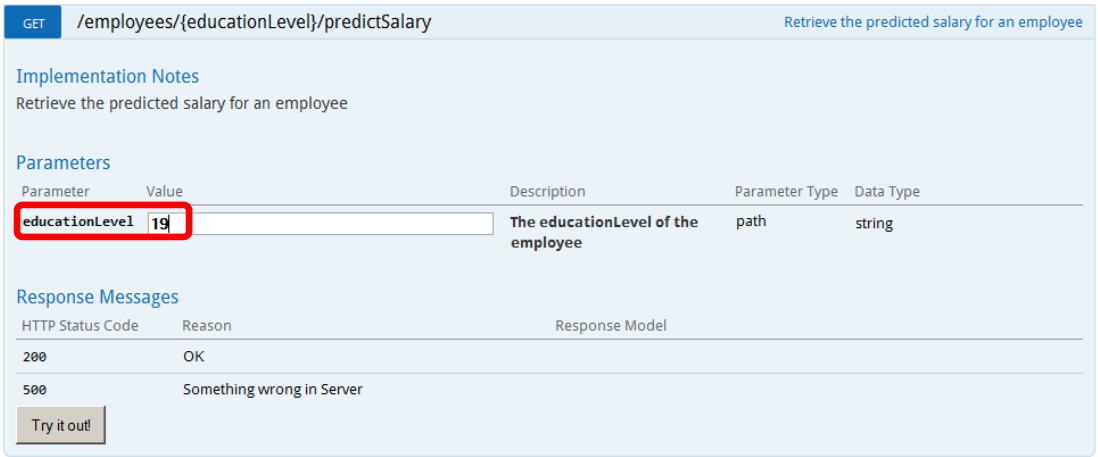
8. We are concerned with the predictSalary operation so click "List Operations" to show the operations related to employees.

   Note that SwaggerUI does not have any knowledge at this point of whether the operation has been implemented.



9. Expand the GET employees/{educationLevel}/predictSalary operation by clicking it.

   The educationLevel will show the expression (required). Replace this with a suitable value, say *19* and then click *Try it out!*.

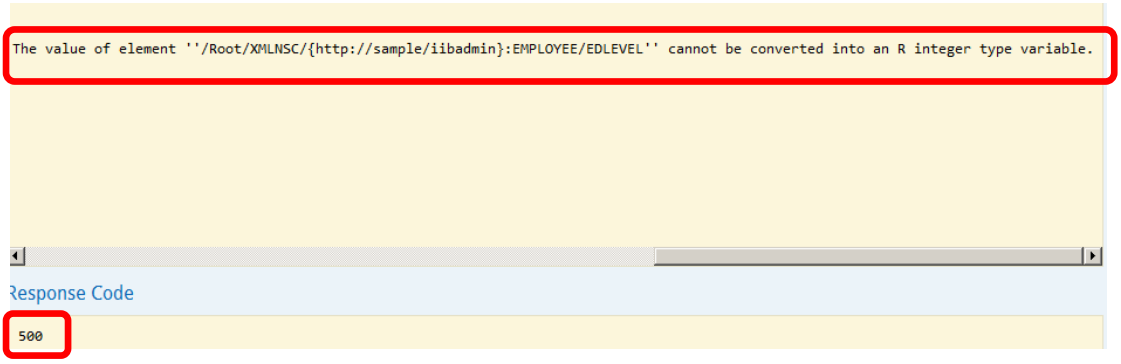


10. If successful, the returned data will look something like this. Note the predicted salary based on the supplied education level, in the response body.

    Note also the response code indicating normal completion.

11. Now use an invalid parameter containing non-numeric characters, for example *abc*.

    In this case, the request fails with a message indicating that the value is invalid and with an appropriate response code to indicate failure.



4.3 Understanding the value returned from R

The R node support in IIB V10 provides an interface for IIB to have the facility to call out to an R environment. Three scripting based interfaces are provided with the R node:

1) Connection script: Run when IIB initially connects to RServe.
2) Evaluate script: Run (per message) when a message reaches the R node.
3) Disconnect script: Run when the Integration Server stops or the application is redeployed.

We have used a very simple R model that predicts a value for SALARY based on a value for EDLEVEL. The data contained in the R model you used above, was based on an extract from the EMPLOYEE table in the DB2 SAMPLE database.

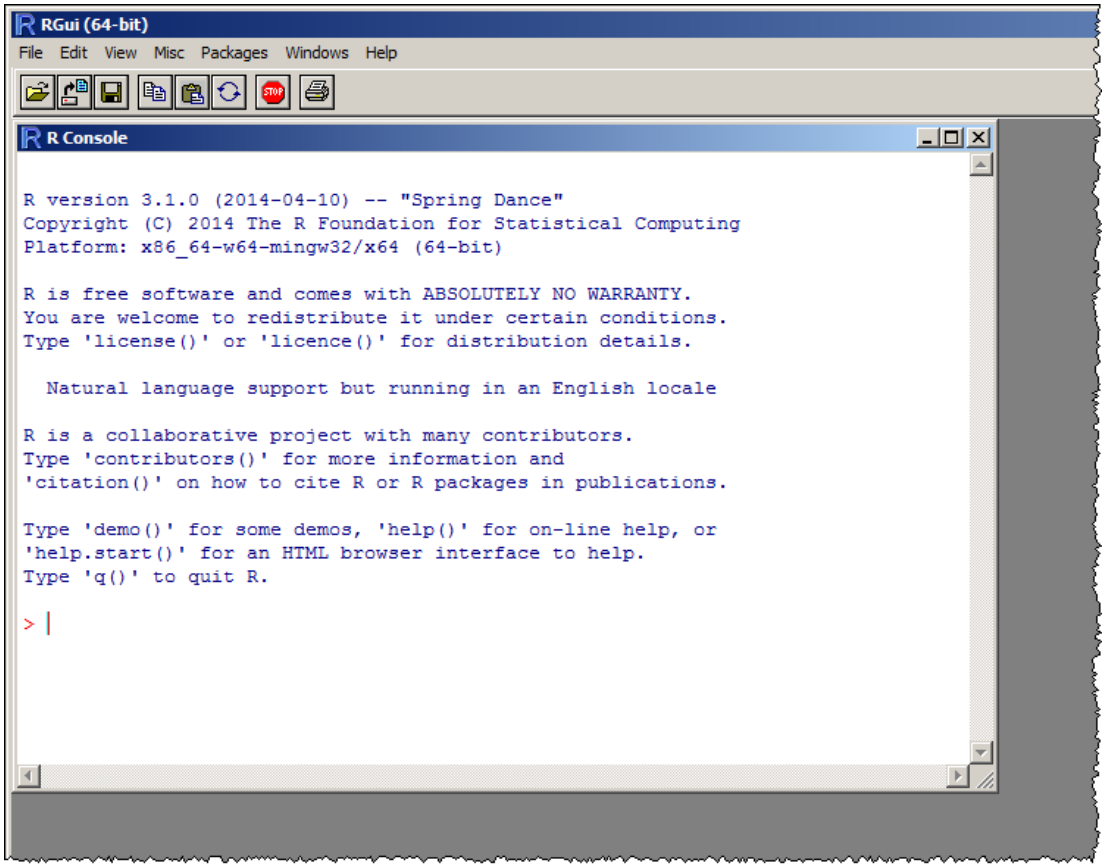
You will now use some provided R scripts to graphically represent how the model works and how the SALARY number was derived by the R model.

4.4 Analyze the model before and after running the application

You will now use R to show a graphical representation of the model before and after you run the application.

1. Open the R Gui interface (Start > All Programs > R > R x64 3.1.0).

   RGui(64-bit) will open with an R Console.

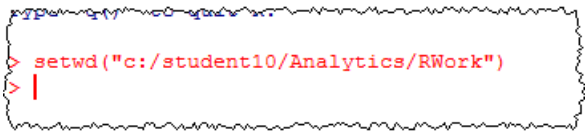


2. In the console type "**`setwd("c:/student10/Analytics/RWork")`**"

   (Note the **Forward slashes**).

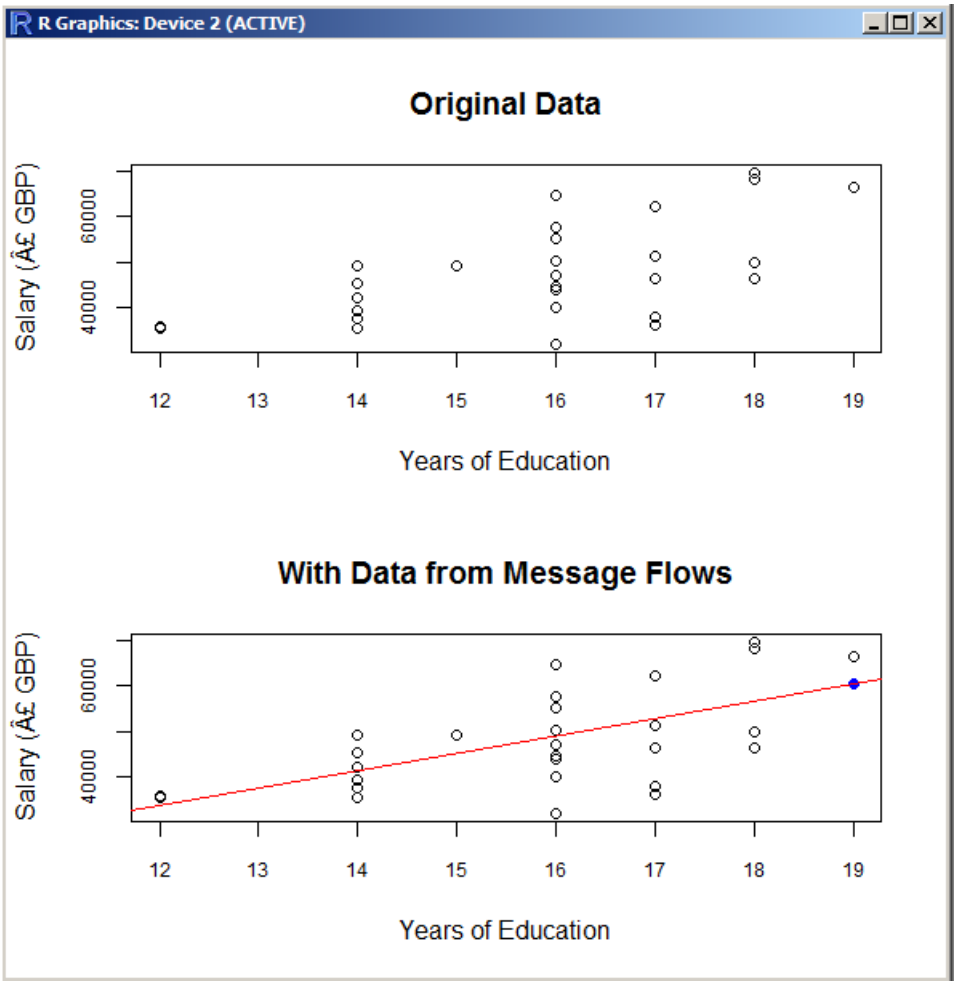
   The command sets the working directory for the R console session, you are setting the value of this working directory to where it can find scripts to run in the student10 directory.

   If the command works there will be no response:



3. Type source "**`source("DrawGraphs.r")`**" and press enter

4. A window with two graphs will open in RGui :



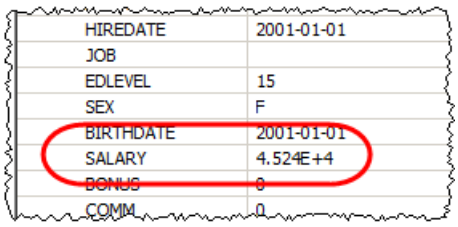
5. The first graph shows the original data that was built with the model.

   The second graph shows a (Red) line. This is a (very simple) line of best fit to the data.

   The (blue) mark at the red line shows how the value for SALARY was predicted given the EDLEVEL value of 19 (the SALARY value is taken from the Y (axis).

6. Go back to the Swagger UI and send a message with EDLEVEL=15.

   Click the green arrow to execute the predictSalary.
   A value of 4.524E+4 will be returned in the SALARY field:

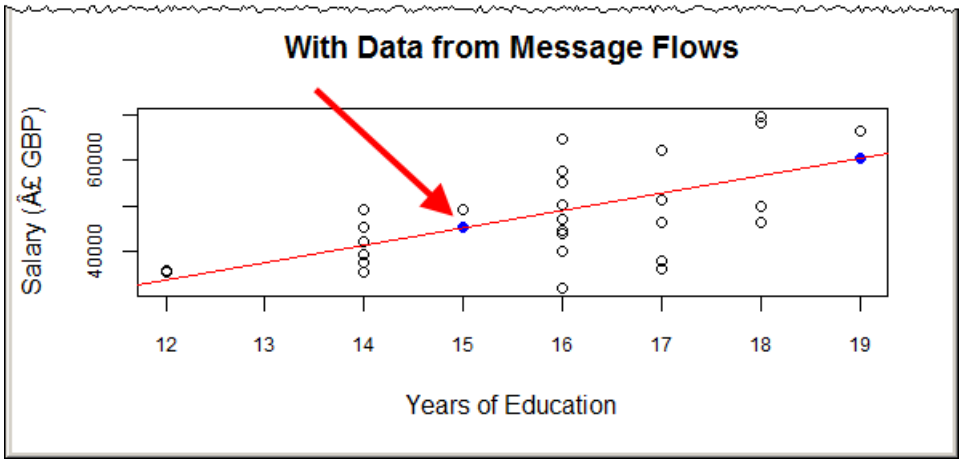


HIREDATE	2001-01-01
JOB	
EDLEVEL	15
SEX	F
BIRTHDATE	2001-01-01
SALARY	4.524E+4
BONUS	0
COMM	0

7. Switch back to the R console and re run the DrawGraphs script

   The command is **`source("DrawGraphs.r")`**
   (highlight the R console and press the up arrow to retrieve previous commands)

8. A blue mark will now appear on the line (above 15 on the X axis).



Again the value passed back to the message flow was taken from the Y axis of the graph.

END OF LAB GUIDE