16L03

IBM Integration Bus

Accessing the Global Cache using a Mapping Node

Featuring:

The REST API tools for IIB
Mapping node Cache Transforms

September 2016
Hands-on lab built at product
Version 10.0.0.6

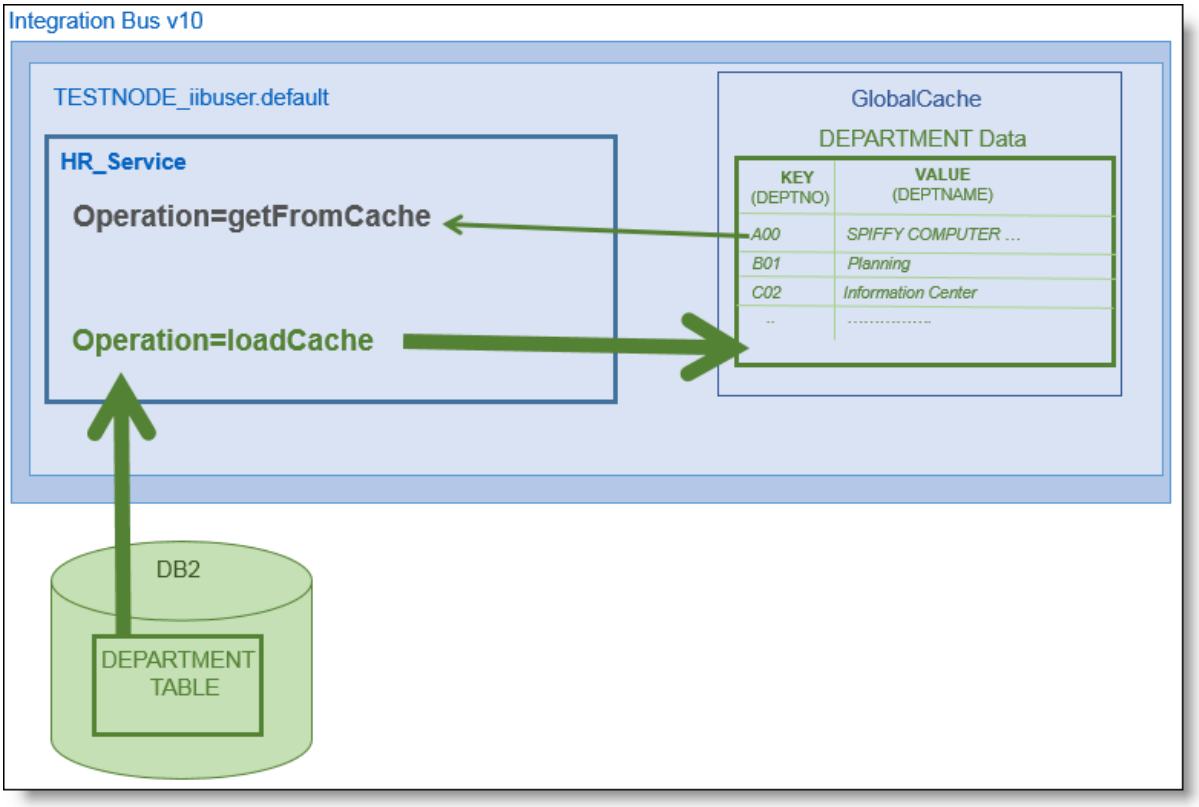
1. Objectives

In this lab, you will implement a REST operation to retrieve data from the Global Cache.

To allow the lab to be completed in the appropriate time, we have provided a REST application that has already implemented the operation to read a database table, and to load the data from the table into the global cache. You will perform the following tasks:

- Import the partially-built REST application
- Investigate the supplied "loadCache" operation
- Configure the IIB node for Global Cache operations
- Deploy and test HR_Service with the supplied "loadCache" operation implemented to obtain data from the DEPARTMENT table in HRDB
- Extend the loadCache operation to add Department Name stored as key/value pairs to the Global Cache using a mapping node.
- Deploy and test HR_Service with the extended loadCache operation so that DEPARTMENT table data is written to the Global Cache
- Create a getFromCache operation to obtain the key/value pair data from the Global Cache.

The following diagram provides a simple outline of the high level components used in this lab guide:



2. Prepare the IIB Node

2.1 Configure TESTNODE_iibuser for REST applications

The IIB support for the REST API requires some special configuration for the IIB node and server. If you have already done the REST API lab in this series of labs, you can proceed straight to the next section.

> **In Windows, switch user and log in as "iibuser", password = "passw0rd".**
>
> **Start the IIB Toolkit from the Start menu.**

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://en.wikipedia.org/wiki/Cross-origin_resource_sharing 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 to a higher-level folder in a DOS window).

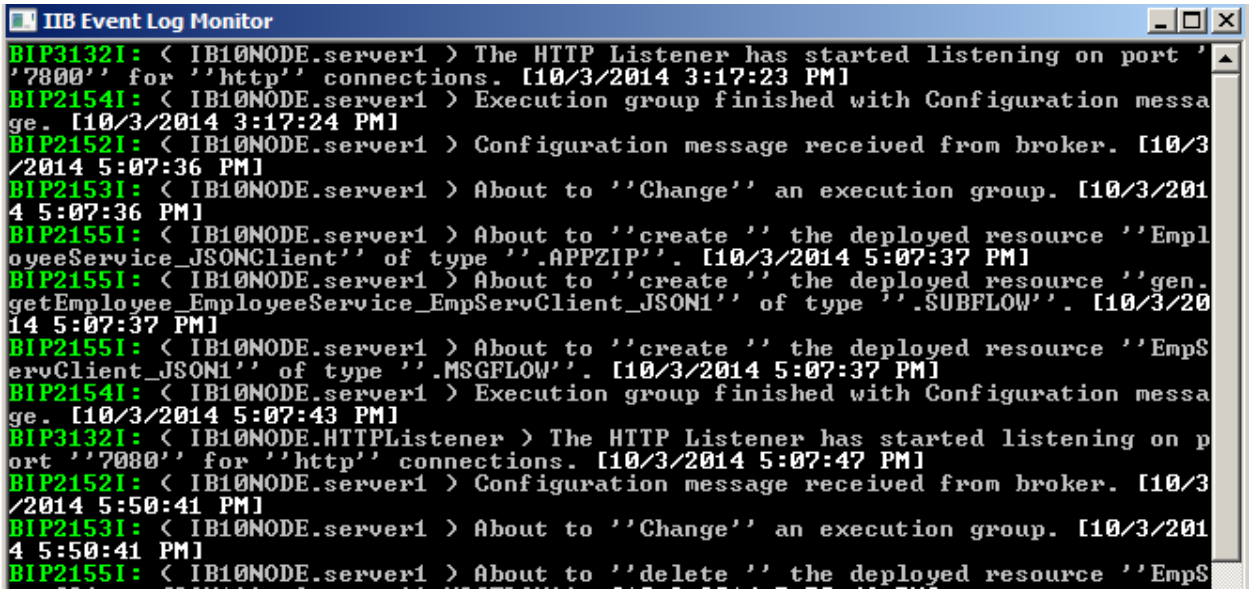
In an IIB Command Console (shortcut on the Start menu), run the command:

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

2.2 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.



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

2.3 Configure TESTNODE_iibuser to work with DB2

If you have already done Lab 1 in this series (create a REST API without an Swagger.json file), you can skip to the next page.

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.	To enable the above definitions to be activated a Stop and restart of the will need to be performed – you will do this in the next section

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

2.4 Configure TESTNODE_iibuser for Global Cache

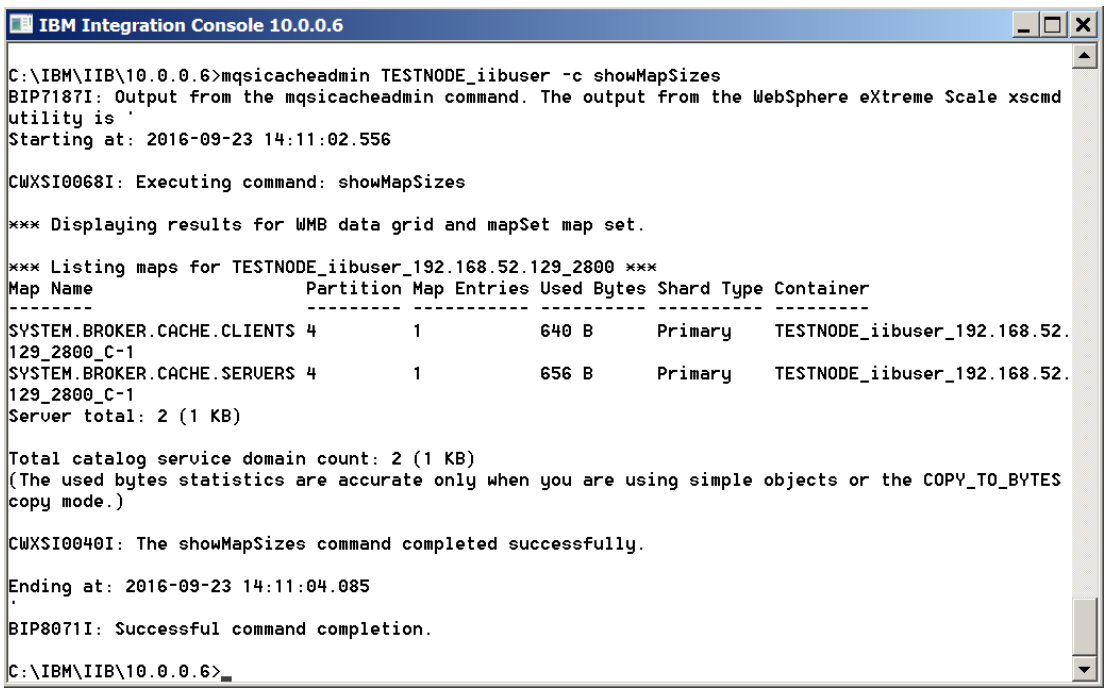
1. In an IIB Command Console, run the commands

```
mqsistop TESTNODE_iibuser

mqsichangebroker TESTNODE_iibuser -b default

mqsistart TESTNODE_iibuser
```

2. Check the current state of the global caches maps by running the command

```
mqsicacheadmin TESTNODE_iibuser -c showMapSizes
```

The output of this command will be similar to below, and will show that the global cache has no user data entries, and just contains the default cache maps.

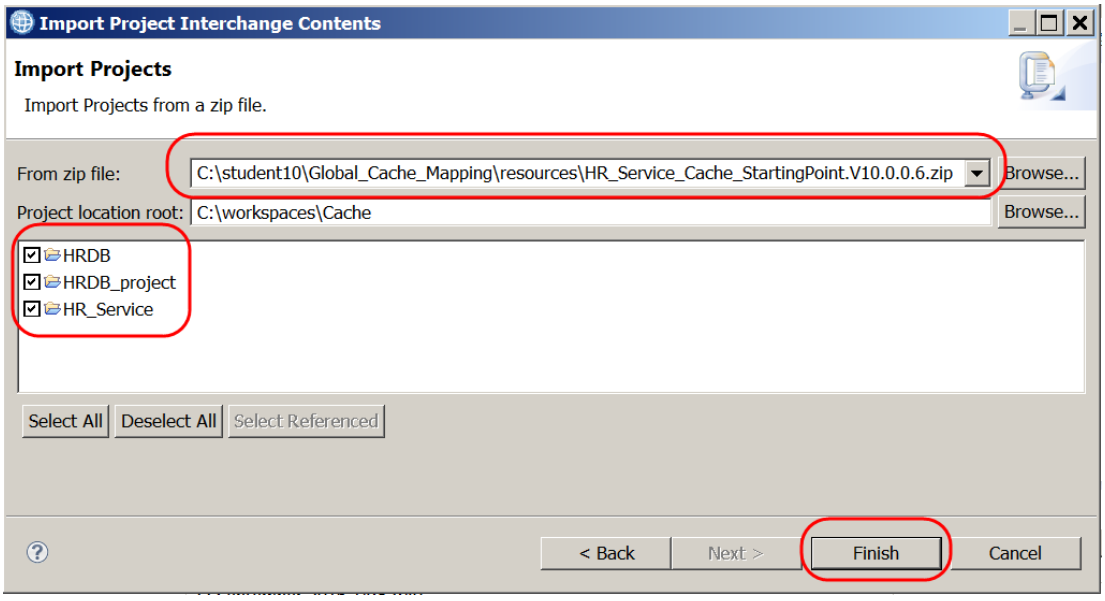


Leave this console open you will be using it later in the lab guide.

Note that the only maps available are the system defaults, each of which currently has 1 entry.

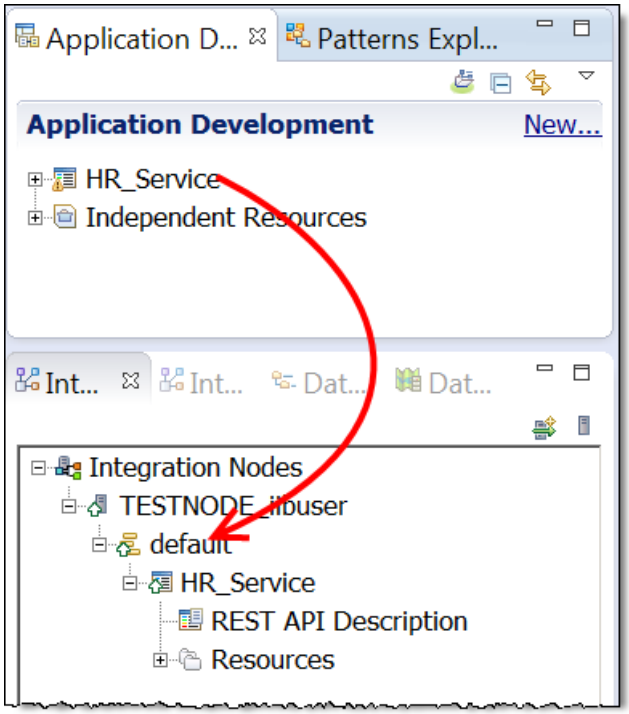
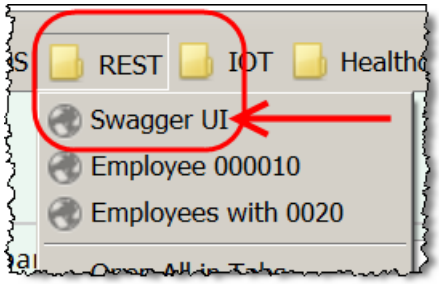
3. Import and test the partial solution

3.1 Import the partial solution

1.	To avoid Toolkit workspace conflicts, create a new IIB workspace. If you already have a workspace open, click File, Switch Workspace. Give the new workspace the name **c:\Workspaces\Cache**
2.	In the Application Development window, right-click in an open area (on the white background), and select Import.
3.	Select the Project Interchange file **c:\student10 \ Global_Cache_Mapping \ resources \ HR_Service_Cache_StartingPoint.V10.0.0.6.zip** This PI file contains three projects. Ensure all projects are selected, and click Finish to import them.  HRDB and HRDB_project contain database definitions for the HRDB database. HR_Service is the partially implemented REST API that you will configure to add DEPARTMENT information to the Global Cache.

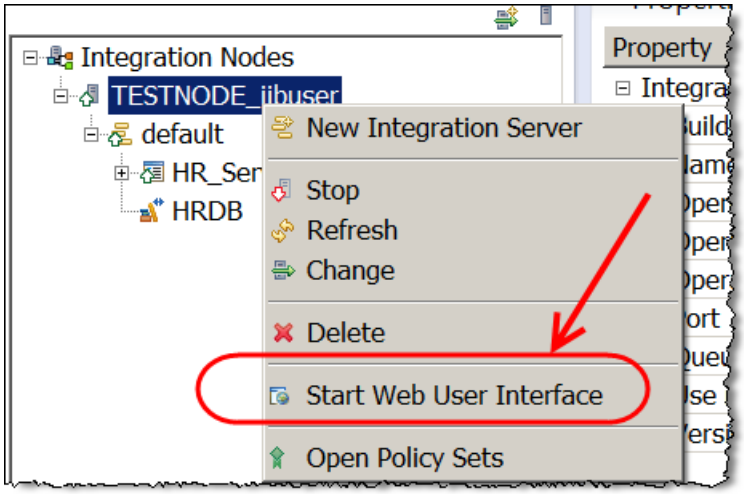
3.2 Deploy and test the HR_Service REST API

In this section you will deploy and test the HR_Service REST API to ensure it works successfully before you implement adding data from the DEPARTMENT table to the Cache.

1.	Deploy the HR_Service REST API to the default Integration Server:  Note: the HRDB Shared Library does not need to be deployed as the Database definition in this file is used at development time only.
2.	Open the Firefox browser and navigate to SwaggerUI (there is a bookmark in the supplied vm): 

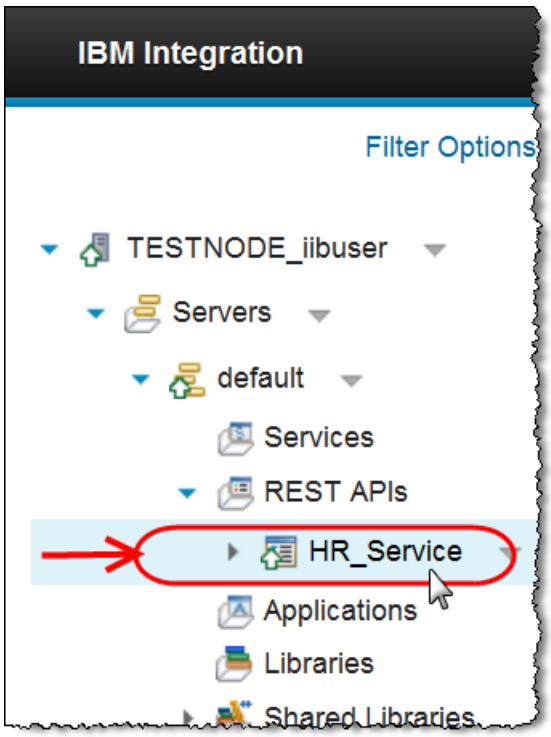
3. Switch to the Integration Toolkit.

   Right click on TESTNODE_iibuser and click “Start Web User interface”:

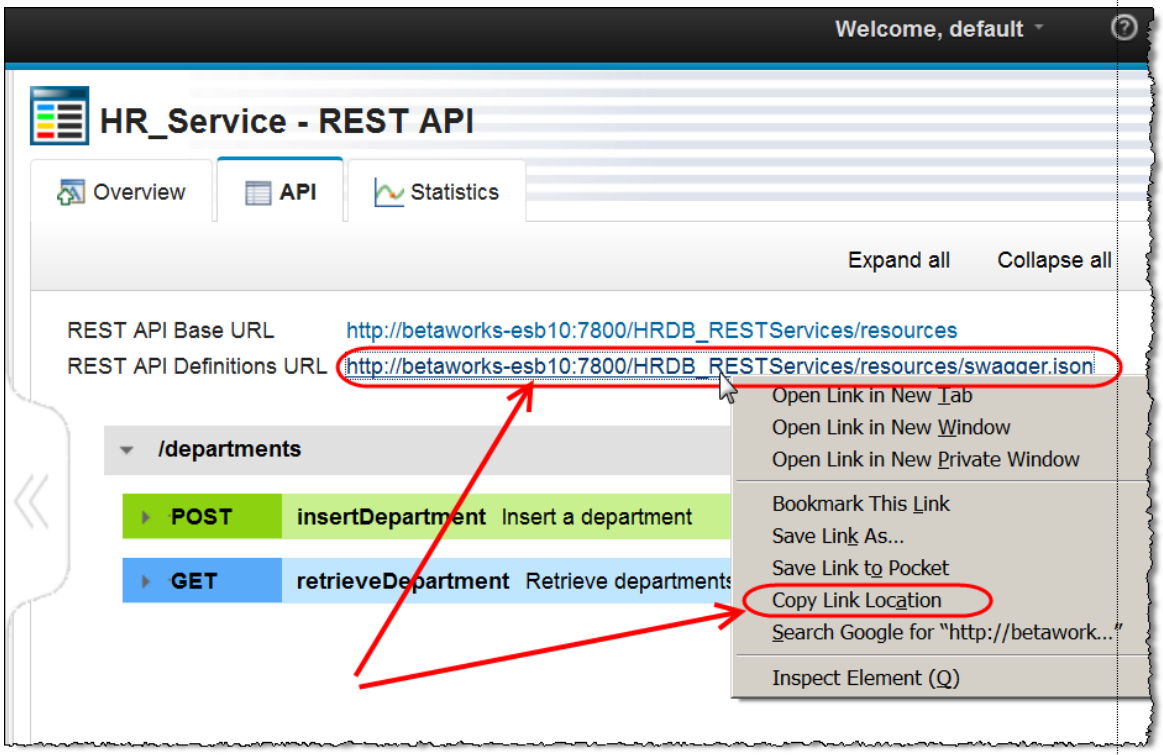


4. After a few seconds the Integration Web UI for TESTNODE_iibuser will open in a browser window.

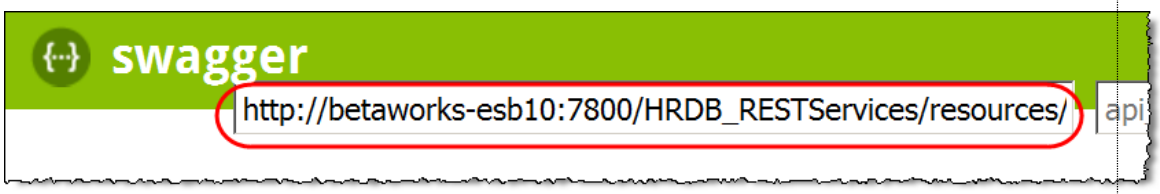
   In the Web UI navigate to the HR_Service (TESTNODE_iibuser > Servers > default > REST APIs). Click the HR_Service name.



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



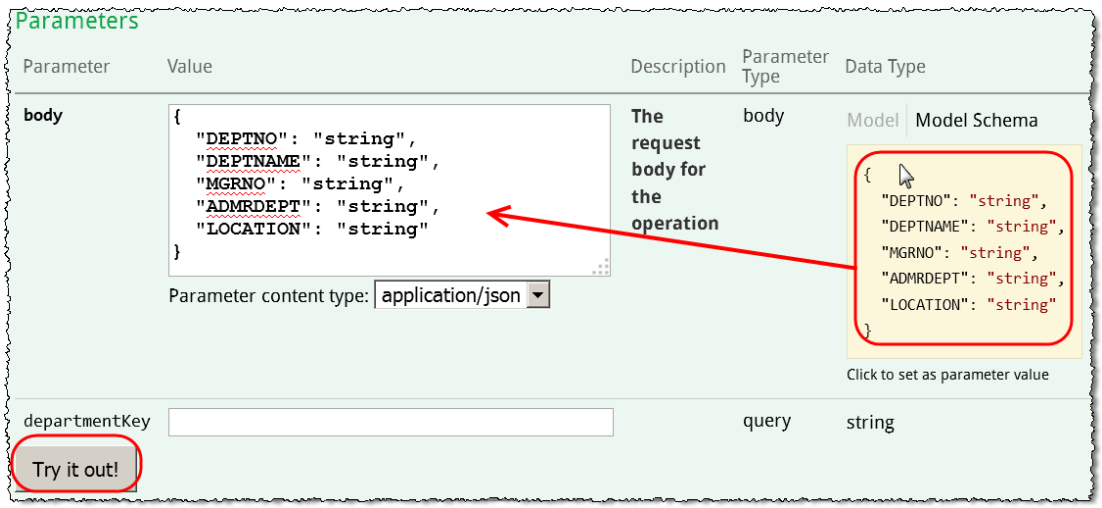
6. Switch back in the SwaggerUI tool, paste the value into the URL field and press Enter:



7. Scroll down to the parameters section.

Click the yellow background of the Model Schema to copy the required fields into the value of the body parameter.

Click the “Try it Out” button:

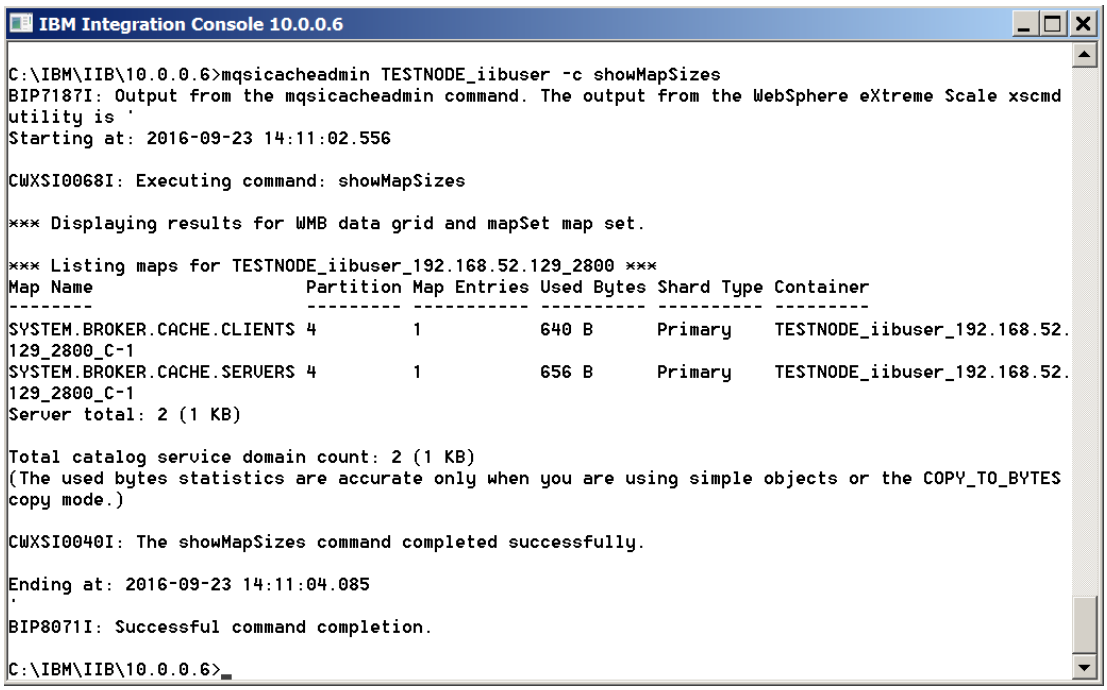


8. After a few seconds you will receive a response containing the data in the DEPARTMENT table:

9. In an IIB Command Console, issue the command

```
mqsicacheadmin TESTNODE_iibuser -c showMapSizes
```

The output of this command will be similar to below, and will show that the global cache has no user data entries, and just contains the default cache maps. There is no map with the name “SYSTEM.BROKER.DEFAULTMAP”.



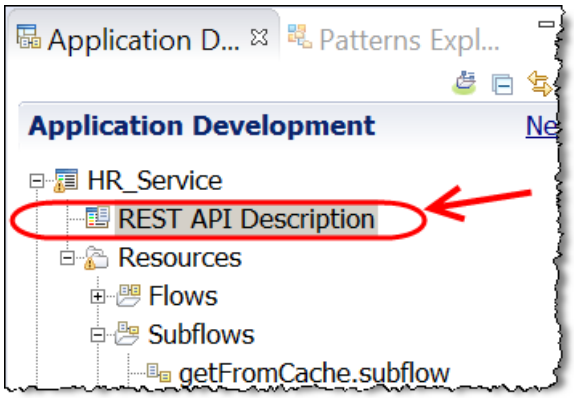
Leave this console open you will be using it later in the lab guide.

In the next section you will configure a map to add data from the DEPARTMENT table to the global cache.

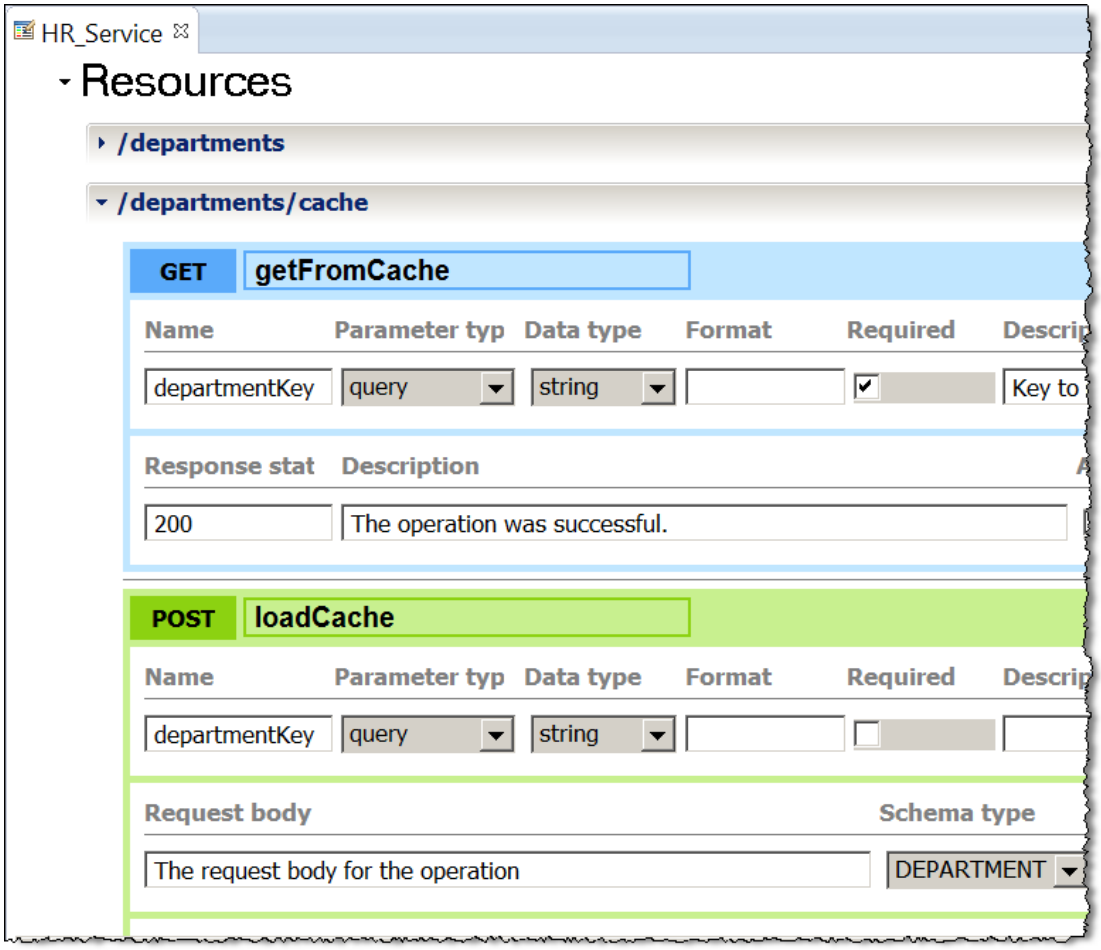
4. Implement the Load Cache operation

You will now extend the loadCache operation to add the rows returned from the DEPARTMENT table (by the SmartRetrieve map) into the Global Cache. Earlier, you activated the Global Cache for the TESTNODE_iibuser node; this scenario will use all of the supplied defaults for the global cache, such as cache grid name, map name, and time to live.

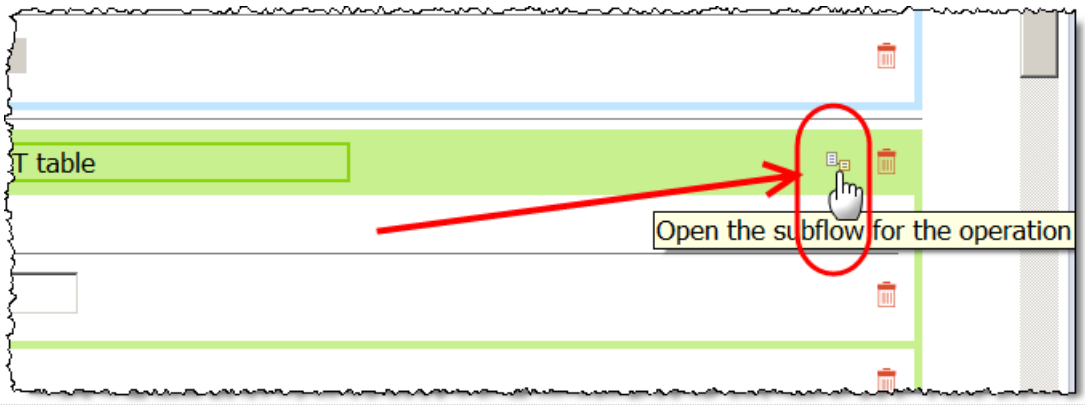
4.1 Investigate the loadCache subflow

1.	In the Integration Toolkit, Expand the HR_Service REST API and double click on the REST API Description: 

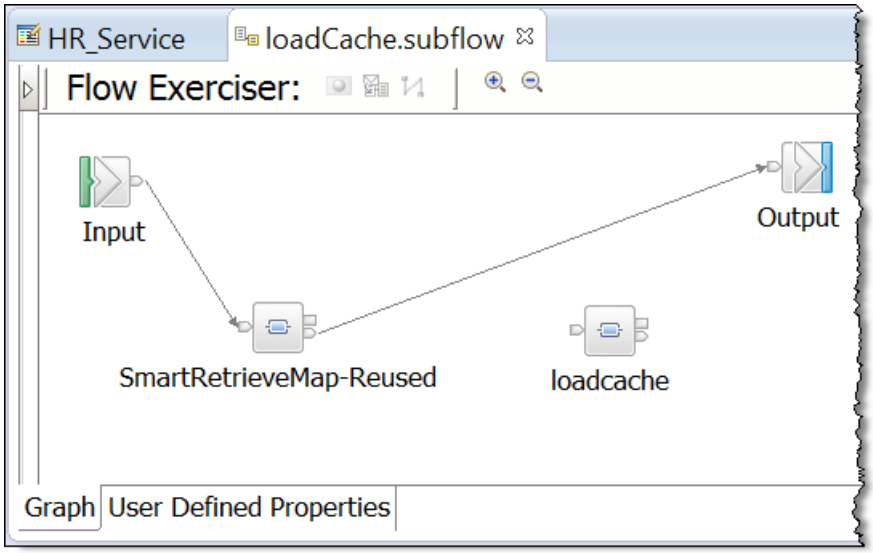
2. In the HR_Service REST API Description, locate the Resource **/departments/cache** and click on it to show the operations, you will see a GET (getFromCache) and a POST operation (loadCache):



3. In the loadCache operation click the "**open the subflow for the operation**" button:



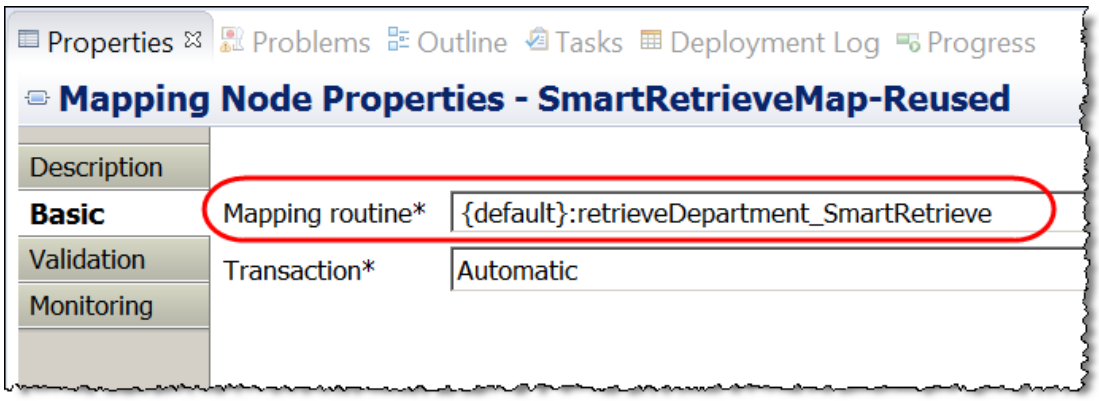
4. You will see the following subflow:



5. (Single Click) the SmartRetrieveMap-Reused mapping node and view the properties for this node (properties tab).

Note that this node has been configiured to reuse the mapping node (***retrieveDepartment_SmartRetrieve***) that was created in Lab 1 of this series (Creating a RESP API without and existing swagger.json file).

When the map is passed a departmentKey with no data, it will retrieve all data in the DEPARTMENT table. This is exactly what we want in order to add DEPARTMENT data to the Global Cache. To preserve the existing function instead of modifying this map, in the section you will configire a separate map that will process the response data from the **retrieveDepartment_SmartRetrieve** map and add it to the global cache.



4.2 Configure the loadcache mapping node

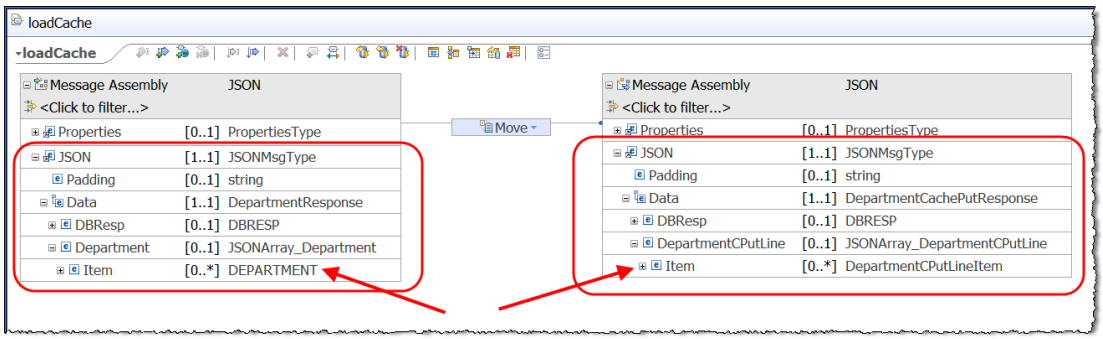
1. Open the loadCache map.

   Expand:

   1) the Input message assembly until you see the array Item with a type of **DEPARTMENT** (the array of department data obtained from the SmartRetreive map).

   2) The Output Message Assembly until you see the Item array element with a type of "**DepartmentCPutLineItem**"

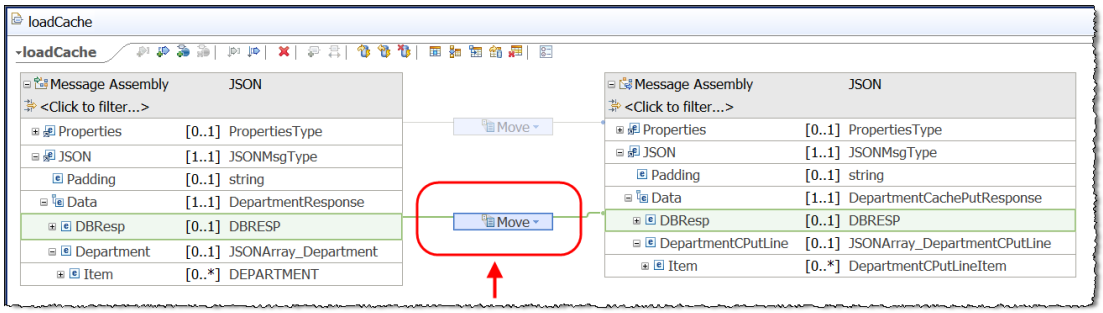
   The loadCache map will look like this:



2. There will be no Database calls in this map so we want to pass the database status response from the SmartRetrieve map in the response "as is".

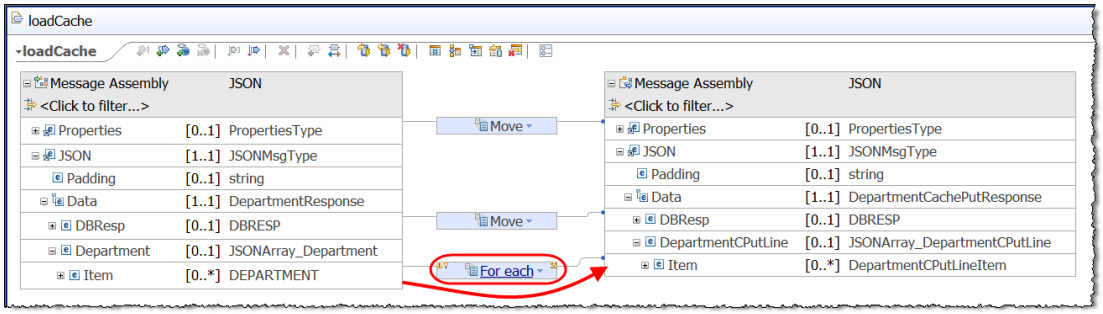
   Connect DBRESP on the input Message Assembly to DBRESP on the output Message Assembly.

   This will create a move operation between the two elements:



3. The data we need to add to the cache is stored in the input Message Assembly in the Department Array.

   In the input Message Assembly connect the **Item** array (of type **DEPARTMENT**) to the **Item** array in the output Message Assembly (of type **DepartmentCPutLineItem**). This will create a "**For each**" transform:



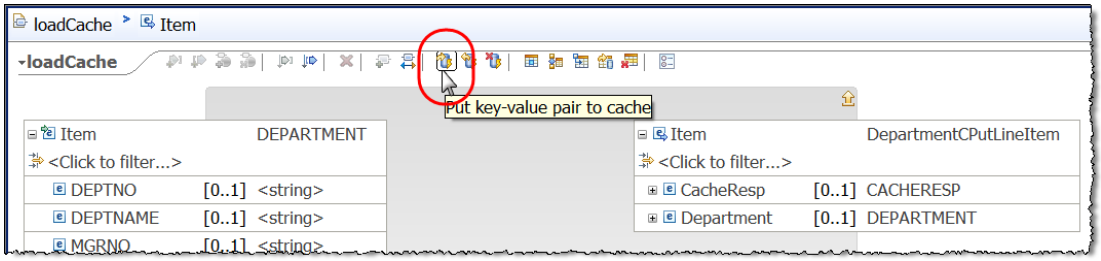
4.2.1 Configure Cache Transforms

In this section you will configure the Cache transforms that will:

a) Add the data to the cache using the Cache Put transform.
b) Process returned data using the Cache Return transform
c) Process what to do in the event of a failure response from the Cache using the Cache Failure transform.

1. Click the "For each" transform txt to show the nested map that will be executed for each array item in the Department array.

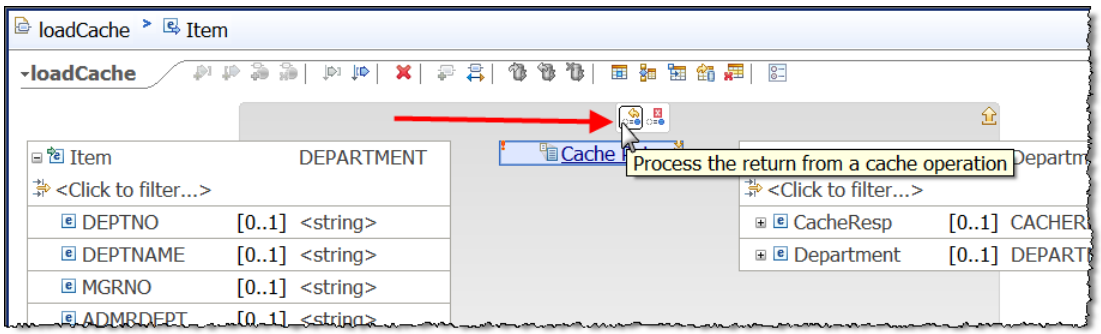
   Click the "Put Key-value pair to cache" icon:



   This will add a Cache Put transform to the map.

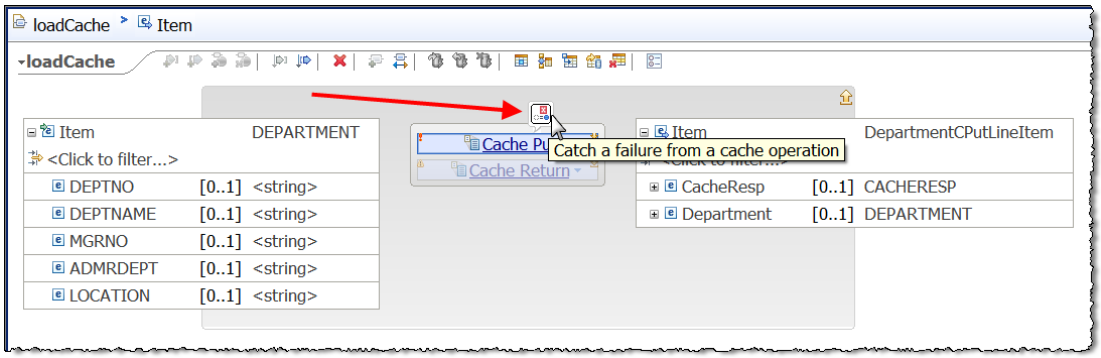
2. Click the blue box of the Cache Put transform and hover over the box until two icons appear.

   Click the "Process the return from cache operation" icon:



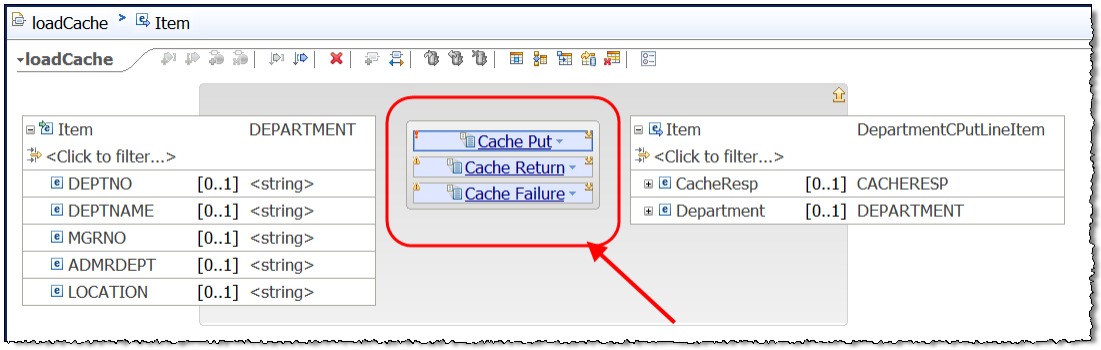
   This will add a Cache Return transform to the map.

3. Repeat the previous step, this time adding the icon to “Catch a Failure from a cache operation”:



This will add a Cache Failure transform to the map.

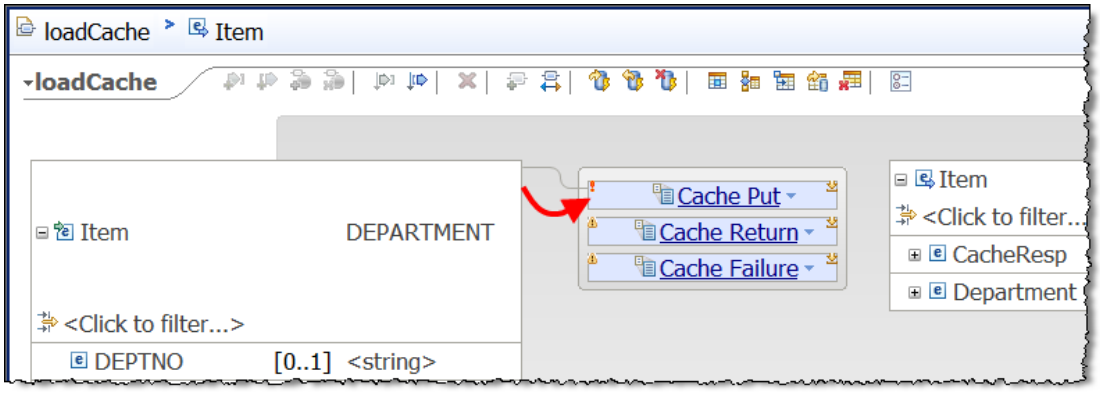
4. When all three transforms have been added your map will look like this:



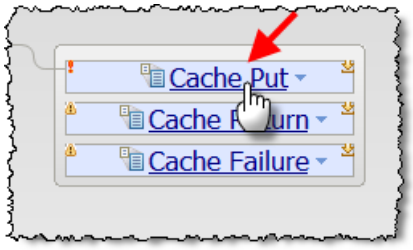
4.2.2 Configure the Cache Put transform

In this next section you configure the Cache Put transform. This is the transform that will add the Department data (retrieved from the HRDB database) to the Cache.

1. Connect the input Item (DEPARTMENT) to the Cache Put Transform:

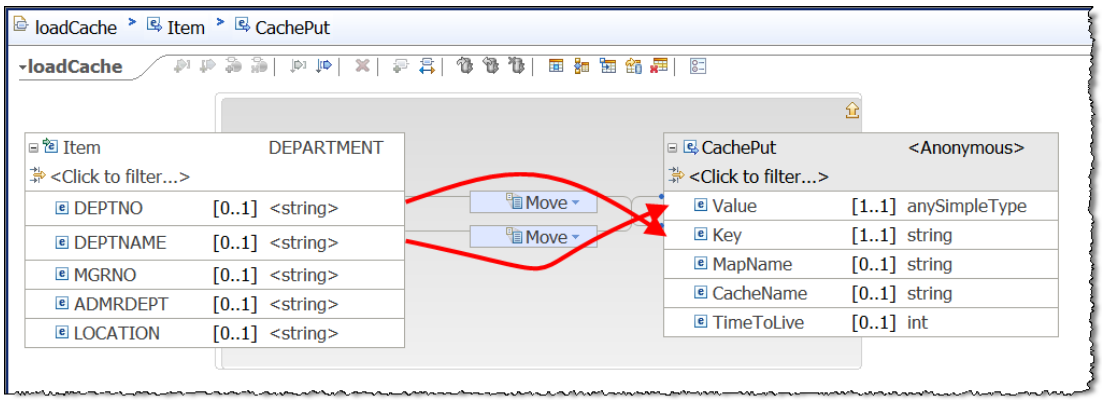


2. Click the **Cache Put** text to enter the nested map for the Cache Put transform:



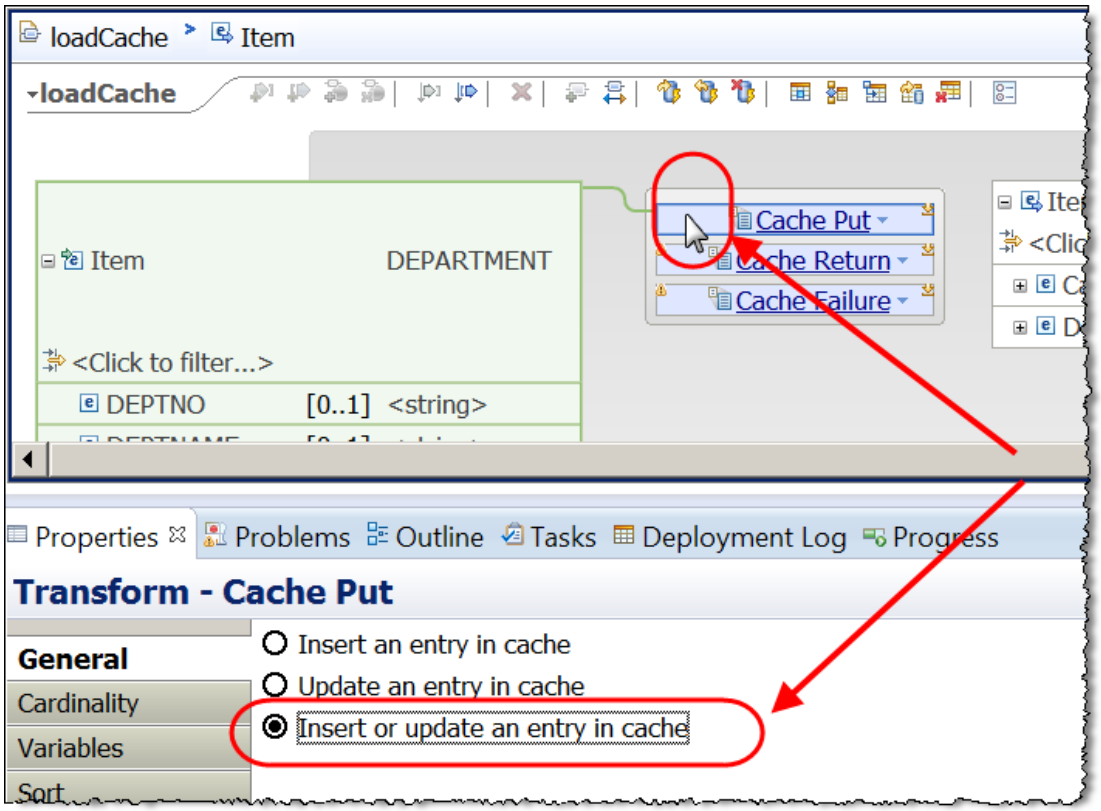
3. In the CachePut element Connect

   1) Item.DEPTNO to CachePut.Key
   2) Item.DEPTNAME to CachePut.Value (ignore the Cast Assist prompt)



4. Click the Yellow arrow to return to the previous level.

   Highlight the blue box of the Cache Put transform and change the properties of the transform to "**Insert or update and entry in the cache**":

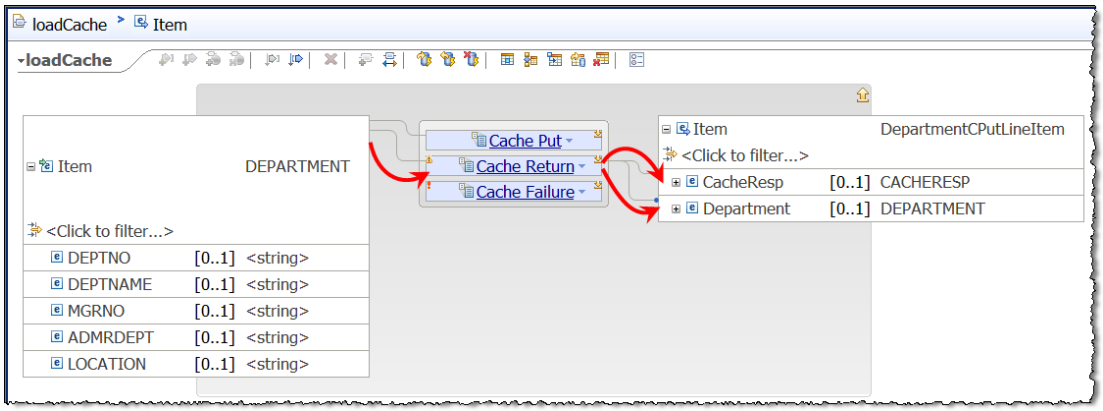


4.2.3 Configure the Cache Return transform

In this next section you configure the Cache Return transform. When used in conjunction with a Cache Put transform the Cache Return transform provides the number of entries added to the cache.

1. Connect the following:

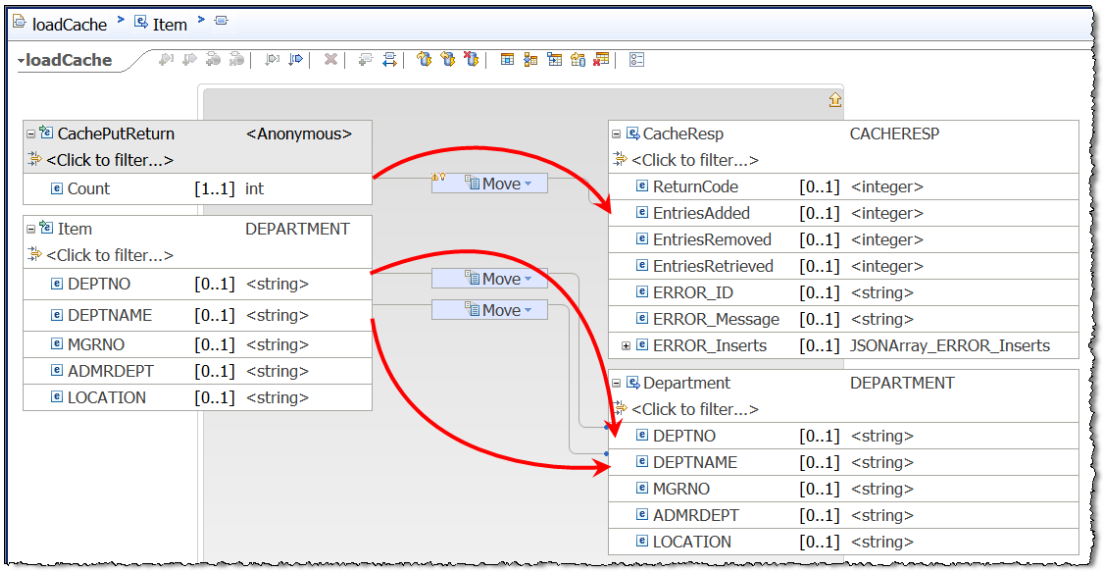
   1) **Item** (DEPARMENT) to **Cache Return**
   2) **Cache Return** to Item.CacheResp (of type CACHERESP)
   3) **Cache Return** to Item.Department (of type DEPARTMENT)



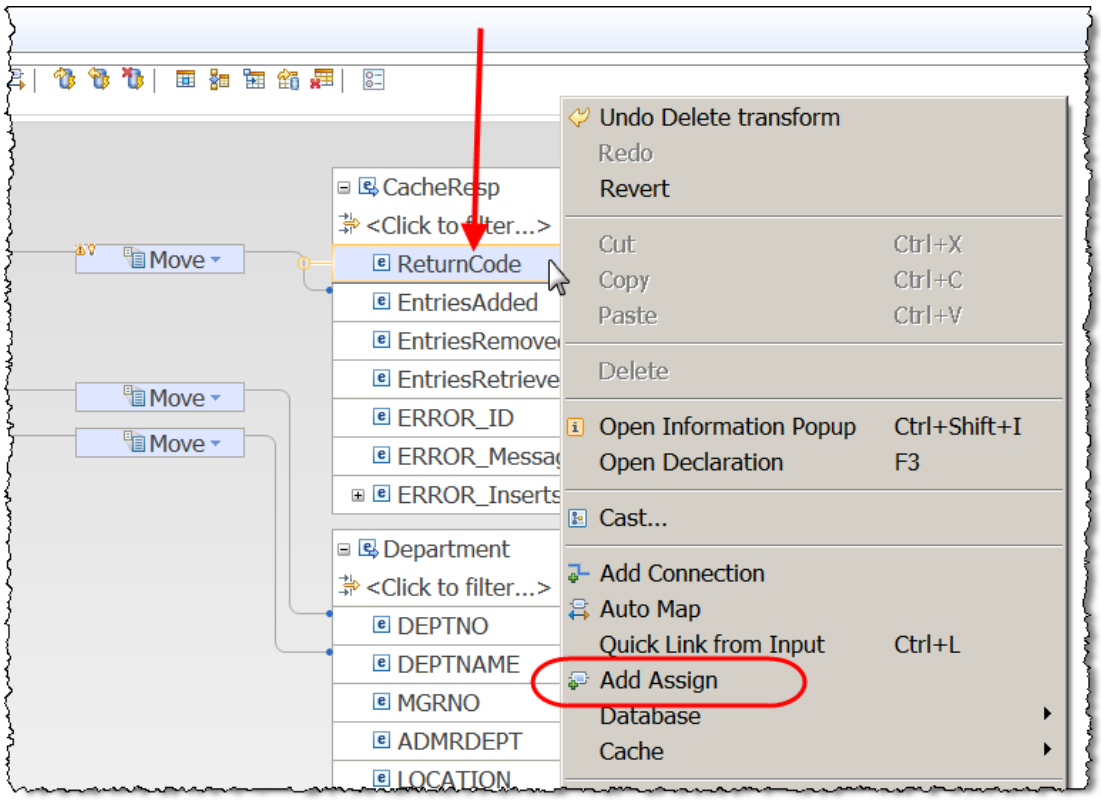
2. Click the Cache Return transform text.

   Connect the following:

   1) CachePutReturn.**Count** to CacheResp.**EntriesAdded**
   2) Item.**DEPTNO** to Department.**DEPTNO**
   3) Item.**DEPTNAME** to Department.**DEPTNAME**

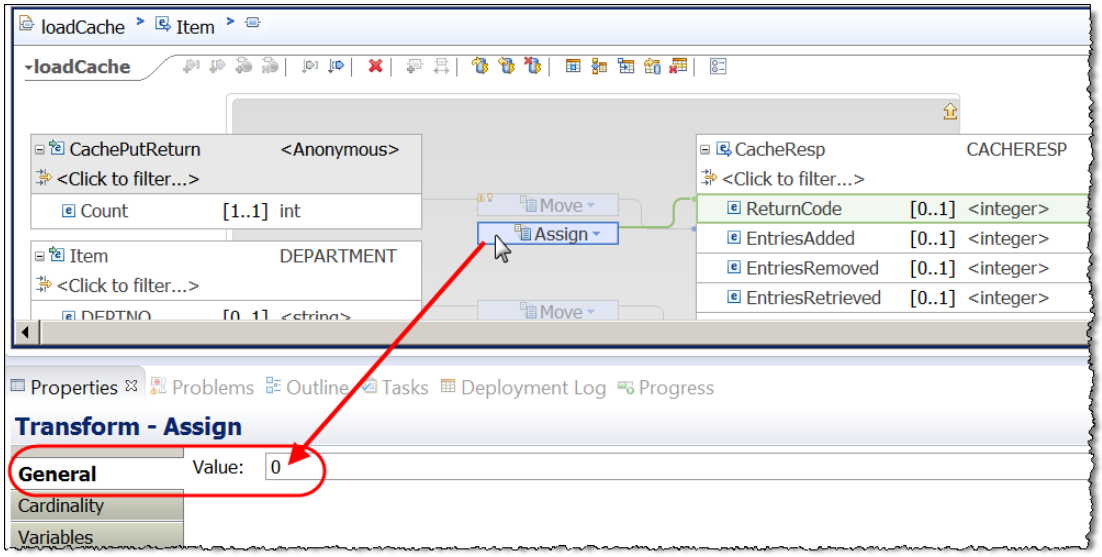


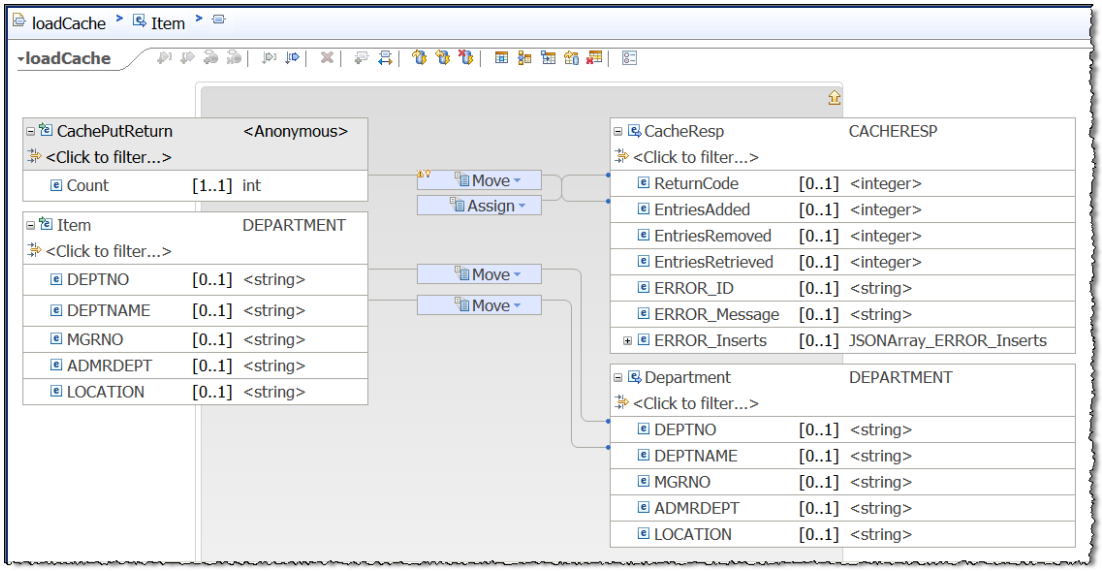
3. Right click on CacheResp.ReturnCode and click “Add Assign”



4. This will add an Assign transform to the CacheResp.ReturnCode element:

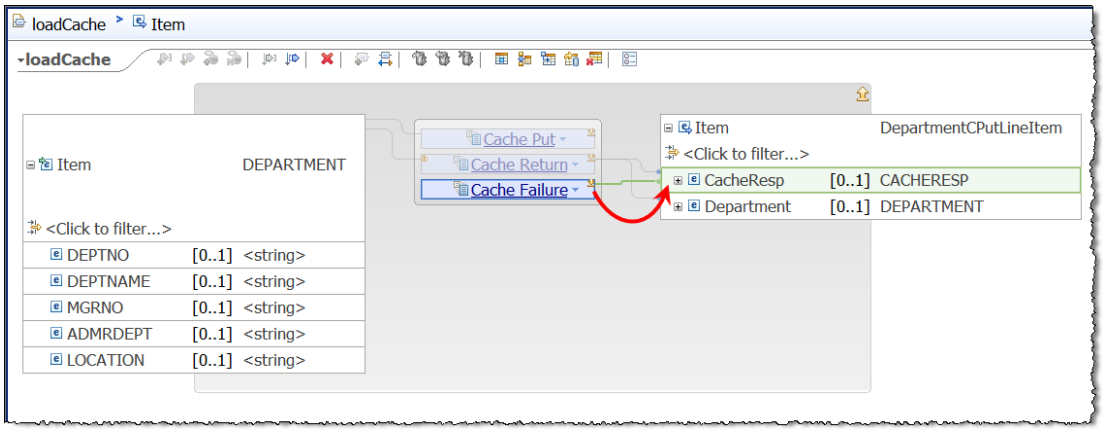
Check that the Value is set to 0 (zero in the General tab in Properties):

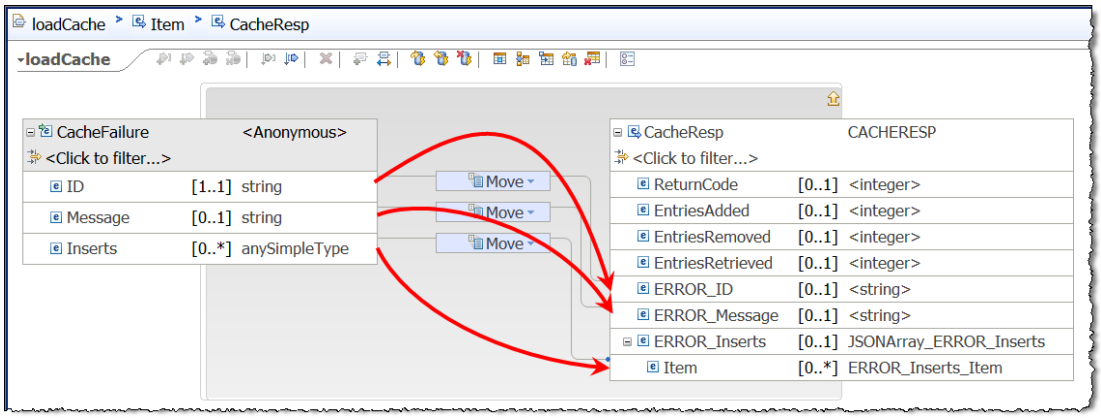
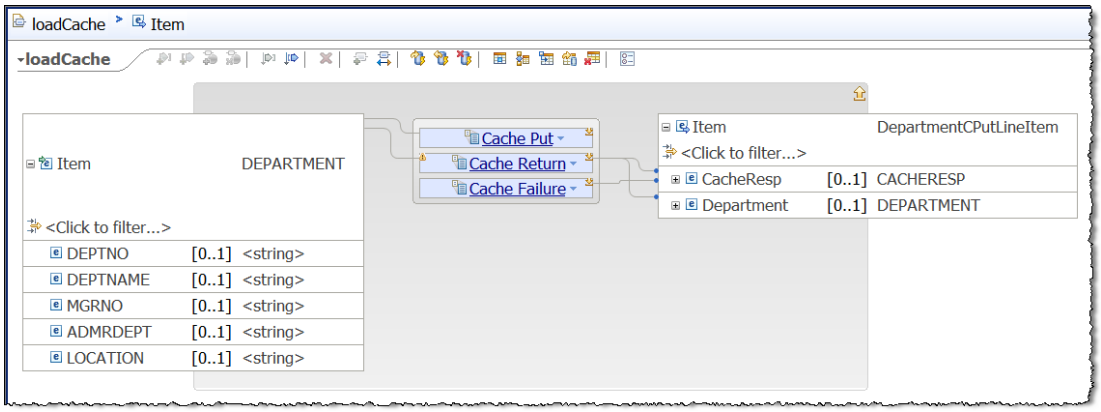


5.	The level associated with the Cache Return transform will now look like this: 
6.	Click the Yellow up arrow to return to the previous level.

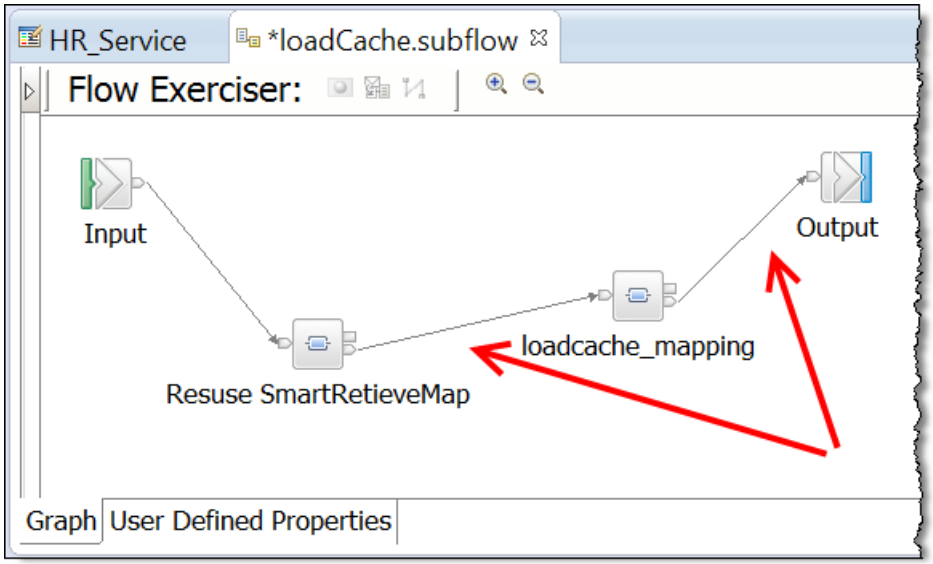
4.2.4 Configure the Cache Failure transform

In this next section you configure the Cache Failure transform. The logic in this transform will be executed when there is a problem accessing the Cache.

1.	Connect **Cache Failure** to CacheResp (of type **CACHERESP**) on the output tree: 

2.	Click the **Cache Failure** text, Connect: 1) CacheFailure.**ID** to CacheResp.**ERROR_ID** 2) CacheFailure.**Message** to CacheResp.**ERROR_Message** 3) CacheFailure.**Inserts** to CacheResp.ERROR_Inserts.**Item** 
3.	Click the Yellow Up arrow to return to the previous level.
4.	The For each level of the loadCache map will now look like this: 
5.	Save the loadCache map and exit the editor.

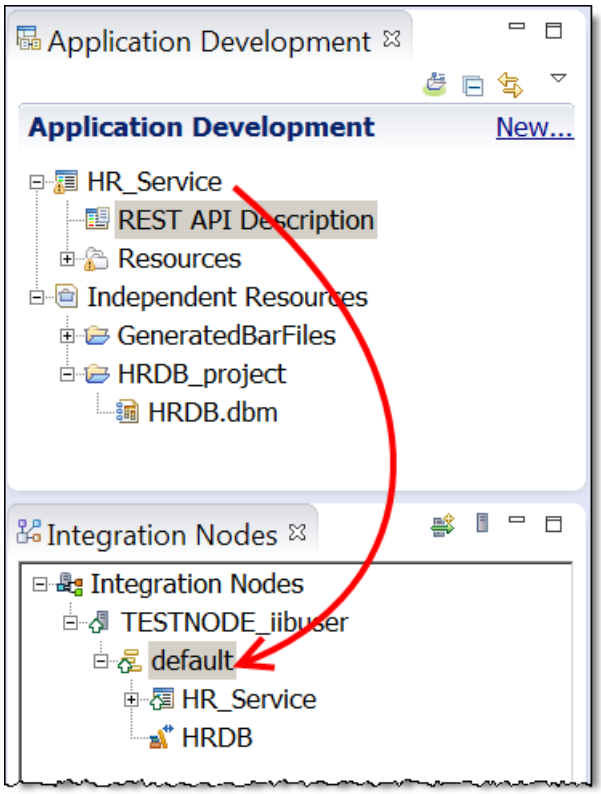
4.3 Complete the loadcache subflow

1.	In the loadCache subflow, connect the loadcache operation as follows: 
2.	Save the subflow and exit the subflow editor.

5. Test the loadCache operation

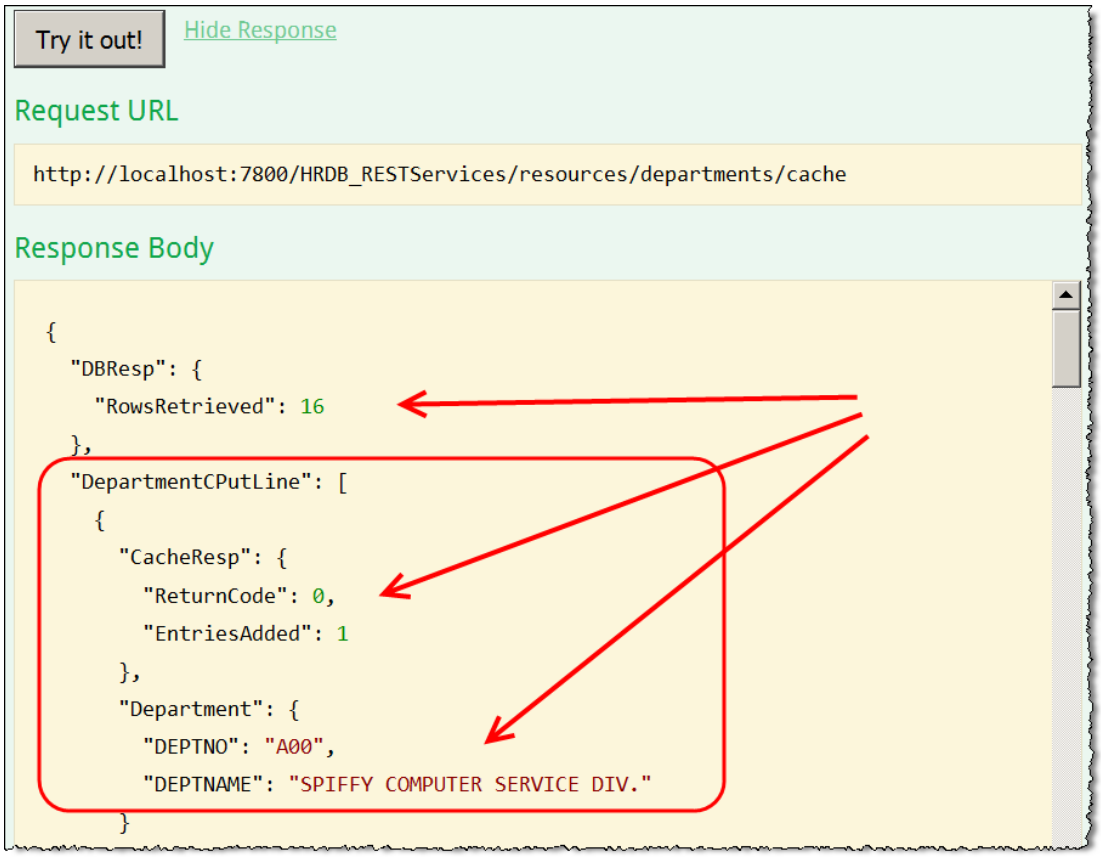
In this section you will test the Cache Load operation that you configured in the previous section.

1. Deploy the HR_Service to the default Integration Server on TESTNODE_iibuser



2. In SwaggerUI, rerun the REST API by clicking the “Try it out” button on the loadCache (POST) operation on Resource **/department/Cache (**you should already have this open from earlier in the lab guide):

3. You will see the following response (the number of rows retrieved will depend on how many rows are in the current version of your HRDB database):

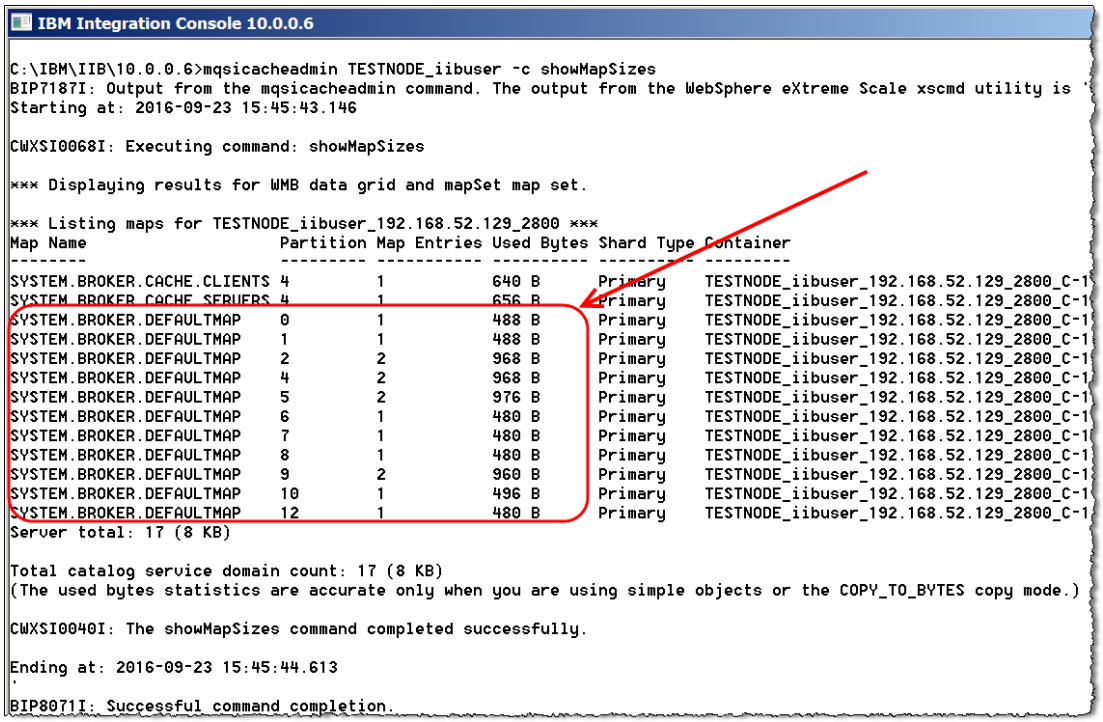


All entries returned from the DEPARTMENT table will have been successfully added to the cache. The response from adding each entry to the Cache is returned in a **json arra**y called **DepartmentCPutLine** which consists of a **CacheResp** field containing the status of the Cache Put command, followed by the data that was added to the cache in the "**Department**" field.

4. Switch to the IIB Console that you left open with the cacheadmin command in it and rerun the command:

```
mqsicacheadmin TESTNODE_iibuser -c showMapSizes
```

The output of this command will be similar to below, and will show that the global cache **SYSTEM.BROKER.DEFAULTMAP** now has user data entries:

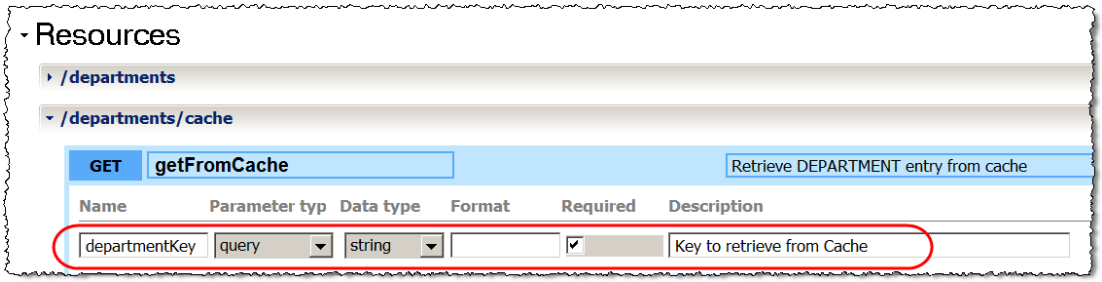
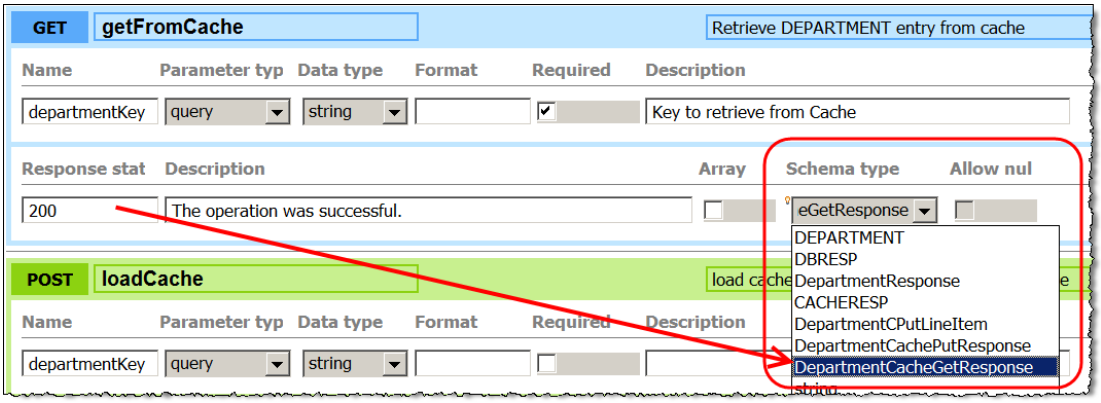


```
IBM Integration Console 10.0.0.6

C:\IBM\IIB\10.0.0.6>mqsicacheadmin TESTNODE_iibuser -c showMapSizes
BIP7187I: Output from the mqsicacheadmin command. The output from the WebSphere eXtreme Scale xscmd utility is '
Starting at: 2016-09-23 15:45:43.146

CWXSI0068I: Executing command: showMapSizes

*** Displaying results for WMB data grid and mapSet map set.

*** Listing maps for TESTNODE_iibuser_192.168.52.129_2800 ***
Map Name                    Partition Map Entries Used Bytes Shard Type Container
--------                    --------- ----------- ---------- ---------- ---------
SYSTEM.BROKER.CACHE.CLIENTS 4         1           640 B      Primary    TESTNODE_iibuser_192.168.52.129_2800_C-1
SYSTEM.BROKER.CACHE.SERVERS 4         1           656 B      Primary    TESTNODE_iibuser_192.168.52.129_2800_C-1
SYSTEM.BROKER.DEFAULTMAP    0         1           488 B      Primary    TESTNODE_iibuser_192.168.52.129_2800_C-1
SYSTEM.BROKER.DEFAULTMAP    1         1           488 B      Primary    TESTNODE_iibuser_192.168.52.129_2800_C-1
SYSTEM.BROKER.DEFAULTMAP    2         2           968 B      Primary    TESTNODE_iibuser_192.168.52.129_2800_C-1
SYSTEM.BROKER.DEFAULTMAP    4         2           968 B      Primary    TESTNODE_iibuser_192.168.52.129_2800_C-1
SYSTEM.BROKER.DEFAULTMAP    5         2           976 B      Primary    TESTNODE_iibuser_192.168.52.129_2800_C-1
SYSTEM.BROKER.DEFAULTMAP    6         1           480 B      Primary    TESTNODE_iibuser_192.168.52.129_2800_C-1
SYSTEM.BROKER.DEFAULTMAP    7         1           480 B      Primary    TESTNODE_iibuser_192.168.52.129_2800_C-1
SYSTEM.BROKER.DEFAULTMAP    8         1           480 B      Primary    TESTNODE_iibuser_192.168.52.129_2800_C-1
SYSTEM.BROKER.DEFAULTMAP    9         2           960 B      Primary    TESTNODE_iibuser_192.168.52.129_2800_C-1
SYSTEM.BROKER.DEFAULTMAP    10        1           496 B      Primary    TESTNODE_iibuser_192.168.52.129_2800_C-1
SYSTEM.BROKER.DEFAULTMAP    12        1           480 B      Primary    TESTNODE_iibuser_192.168.52.129_2800_C-1
Server total: 17 (8 KB)

Total catalog service domain count: 17 (8 KB)
(The used bytes statistics are accurate only when you are using simple objects or the COPY_TO_BYTES copy mode.)

CWXSI0040I: The showMapSizes command completed successfully.

Ending at: 2016-09-23 15:45:44.613

BIP8071I: Successful command completion.
```

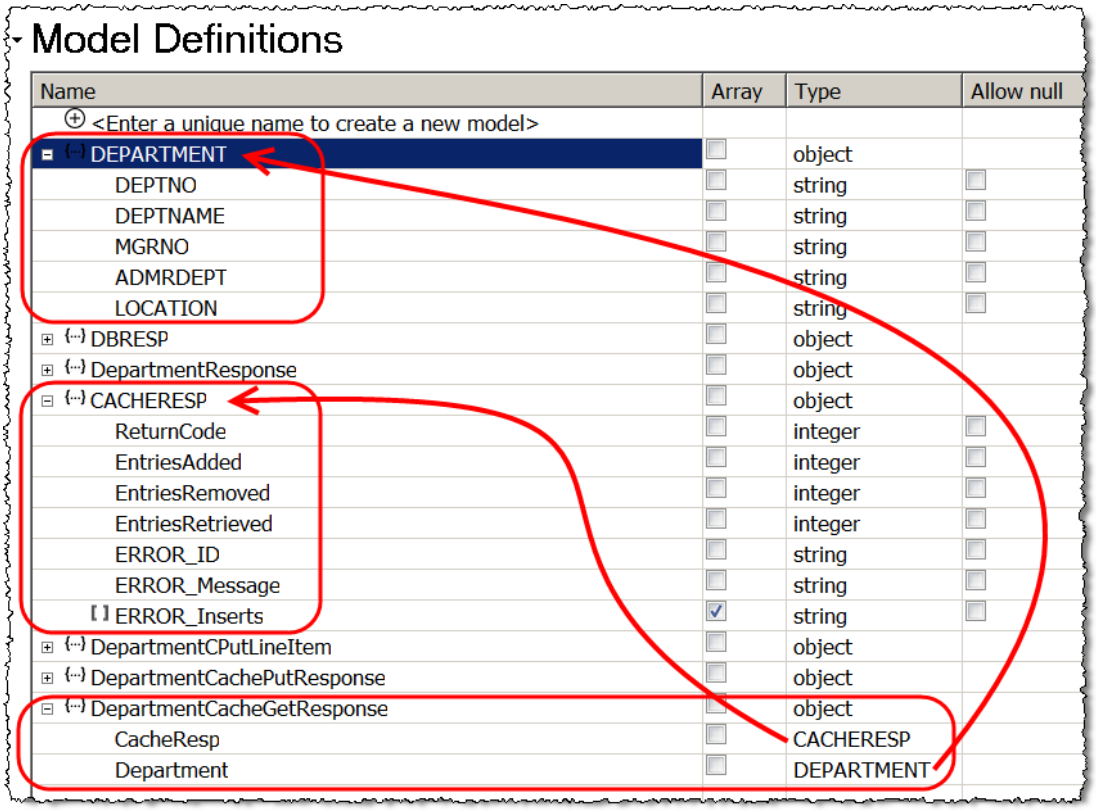
Each container has either 1 or 2 “Map Entries”, representing the data from each row in the Department table.

6. Implement the Get Department Cache operation

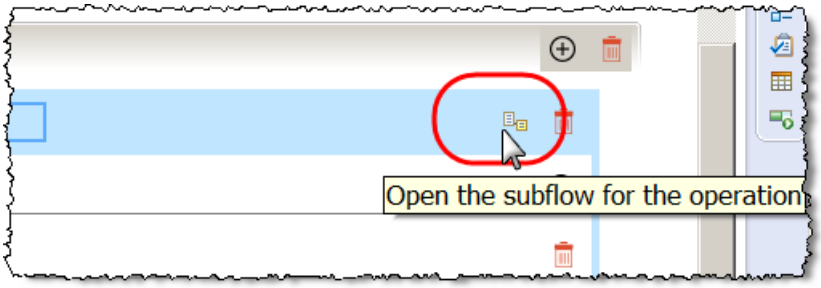
6.1 Review the */departments/cache* Resource Definition

1.	Open the REST API Description for the HR_Service. Expand the Resources section, then expand /departments/cache and scroll to the getFromCache (GET) operation:
2.	Note the (GET) operation has a mandatory ("required") parameter called **departmentKey:** 
3.	Note the (successful) response status of 200 from the getFromCache operation has been specified with a Schema Type of DepartmentCacheGetResponse: 

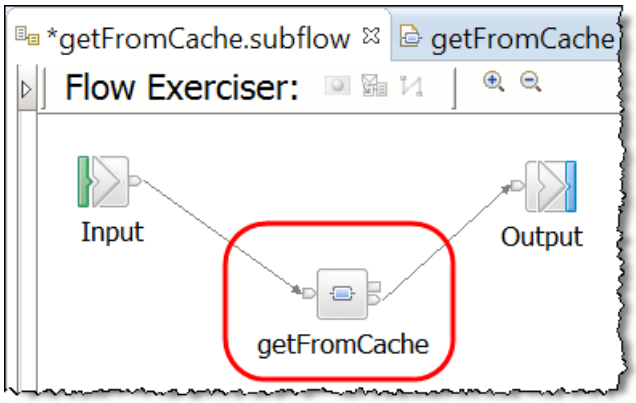
4. You can see the structure of the data that will be returned in **DepartmentCacheGetResponse** from the Models definition within the REST API description:



5. In the **getFromCache** operation, click the "**Open the subflow for the Operation**" icon to open the implementation for the **getFromCache** operation:

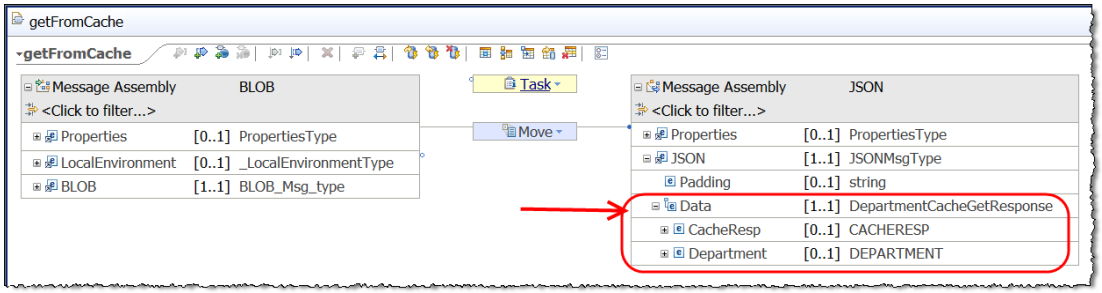


6. The implementation consists of a single map called **getFromCache**. Double click on the map to open the map:



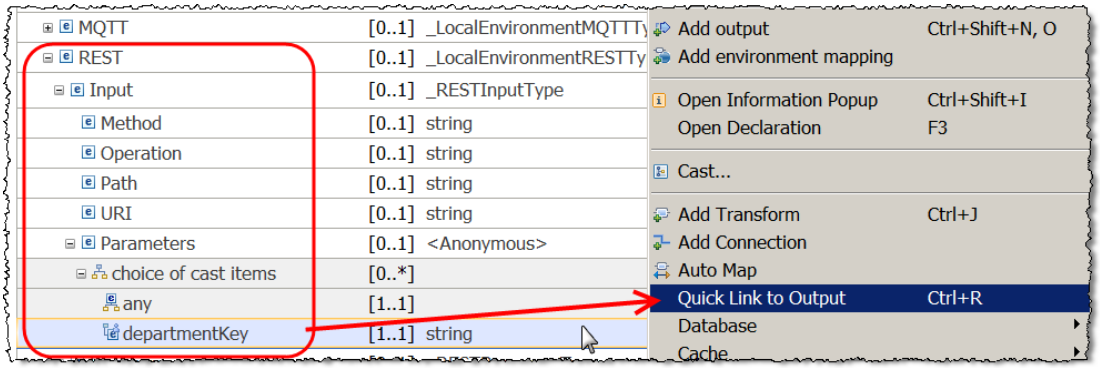
6.2 Review and complete the getFromCache map.

1. In the getFromCache map, on the output Message Assembly, expand JSON.Data and note that the output matches the DepartmentCacheGetResponse Schema type that was specified in the REST API Description (ie it is made up of two data structures of type CACHERESP and DEPARTMENT):

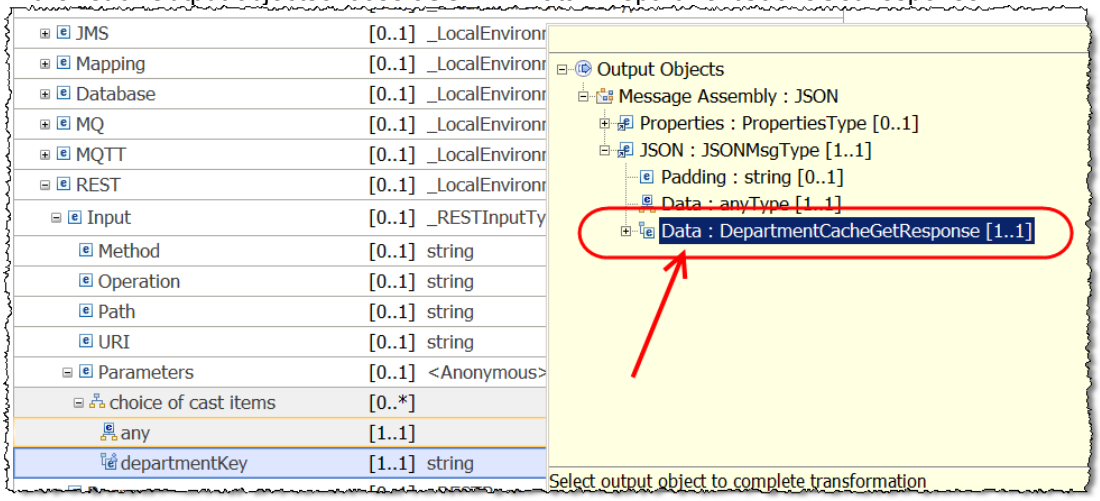


2. On the input Message Assembly, expand “LocalEnvironment > REST > Input > Parameters > “choice of cast items” and note that the (string) parameter **departmentKey** that was specified on the REST API Description is defined in the input.

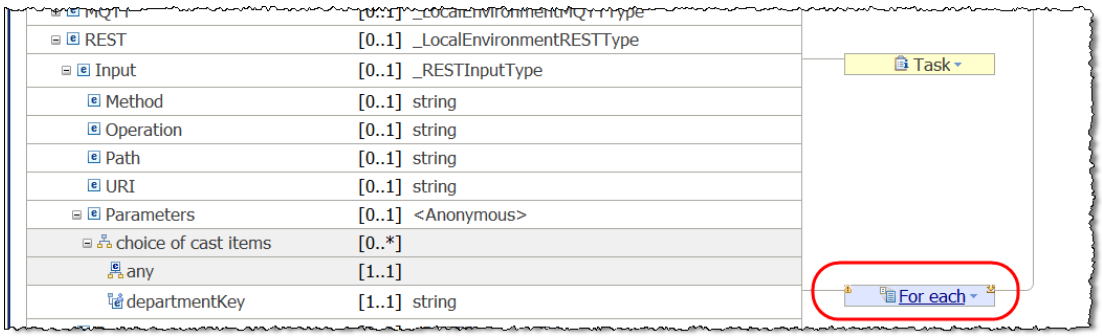
   Right click on departmentKey and choose “**Quick Link to Output**”:



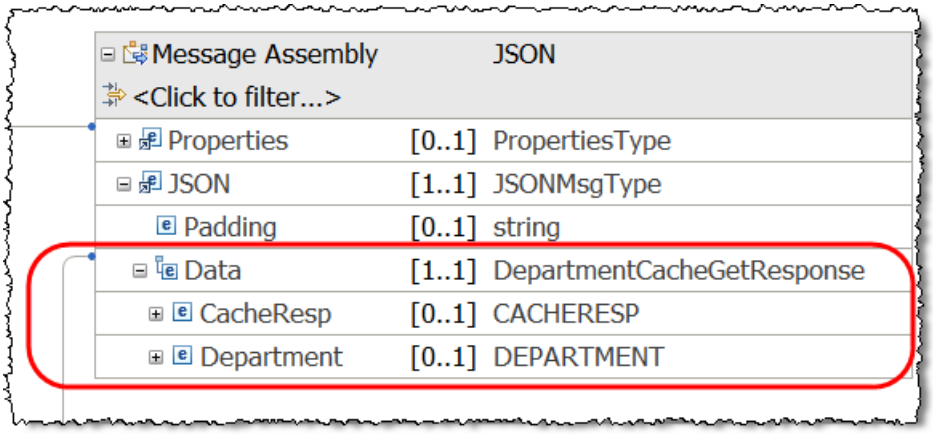
3. In the list of Output objectschoose JSON > “Data : DepartmentCacheGetResponse”:



4. A For each transform will be created,



It will be connected to the output Message Assembly at Data (of type DepartmentCacheGetResponse)

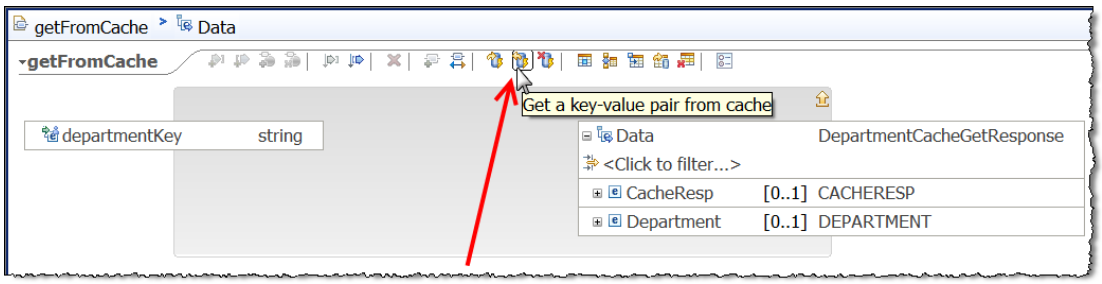


5. Click the For each transform to configure the cache operations.

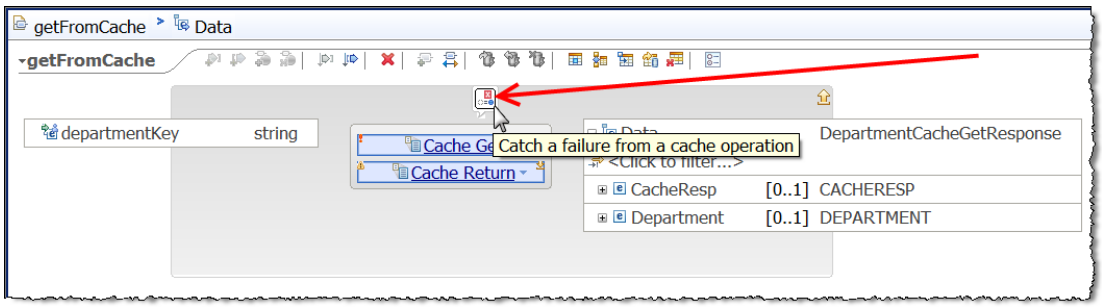
6.3 Configure the Cache (Get) transforms

The ability to obtain information from the Global Cache is done using a Cache Get transform. In this next section you will configure the Cache Get, Cache Return and Cache Failure transforms.

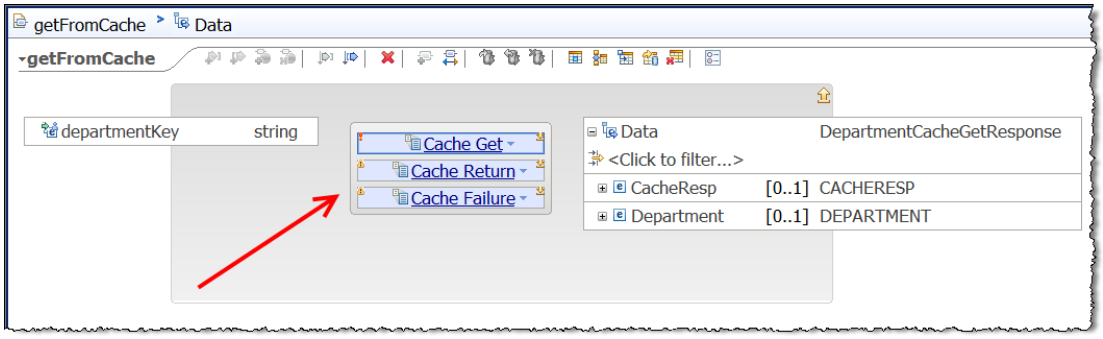
1. No transforms will exist when you first travel down to the For each nested map. Click the icon “Get a key-value pair from the cache” to add the Cache Get and Cache Return transforms:



2. Click the box surrounding both the Cache Get and Cache Return transforms, hoverover the box until the dialog icon to “Catch a failure from the cache operation” appears, click the icon to add the Cache Failure transform:



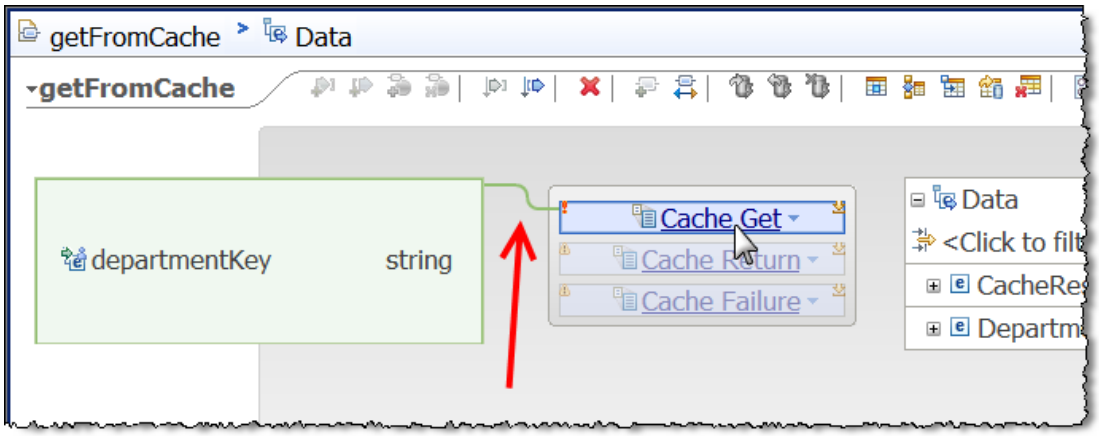
3. Three Cache transforms will now have been created in the nested map:



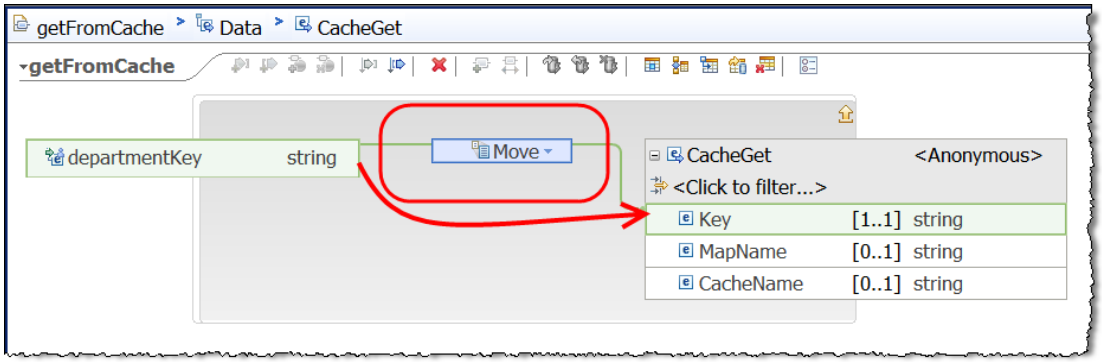
6.3.1 Configure the Cache Get transform

1. Connect departmentKey to the Cache Get transform.

   Click Cache Get to configure which key to use.



2. In the Cache Get nested map, connect **departmentKey** to **Key** (*this specifies the key to retrieve using the CacheGet command*):

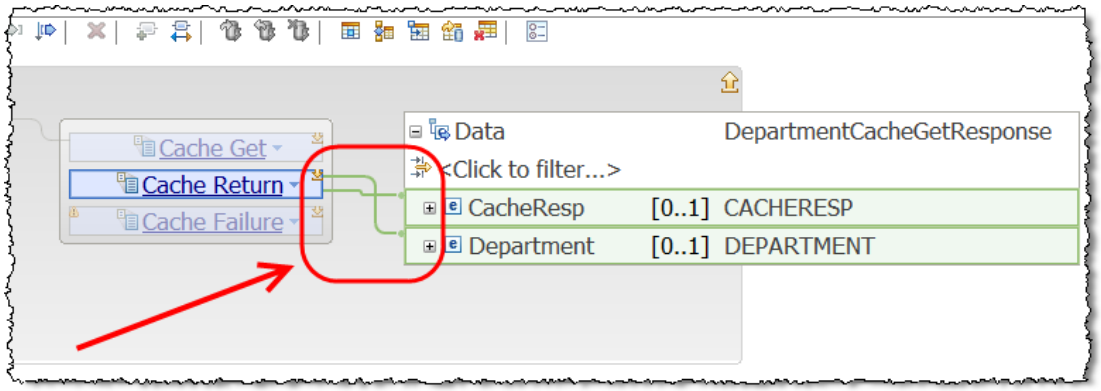


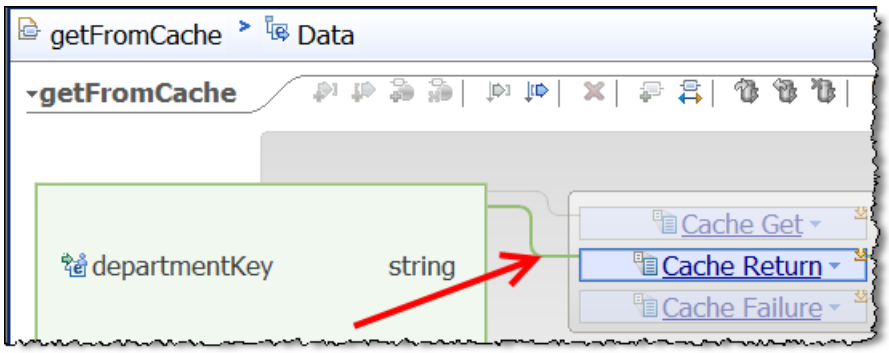
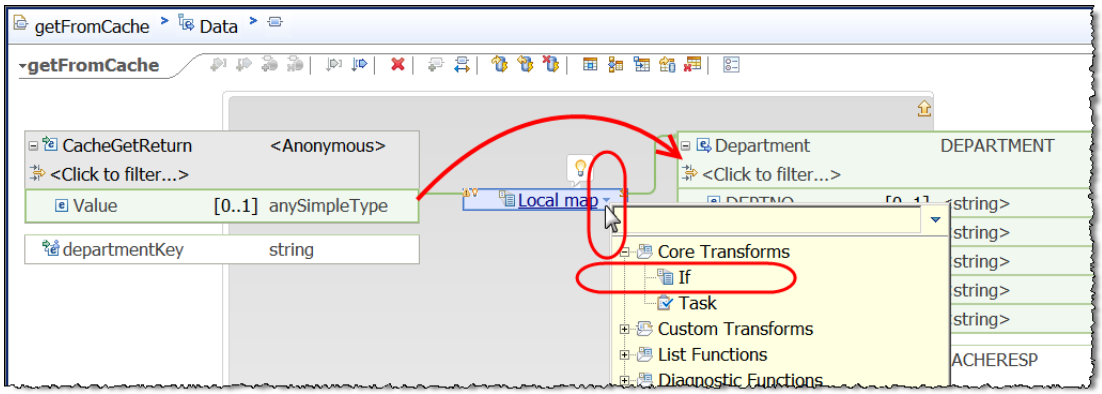
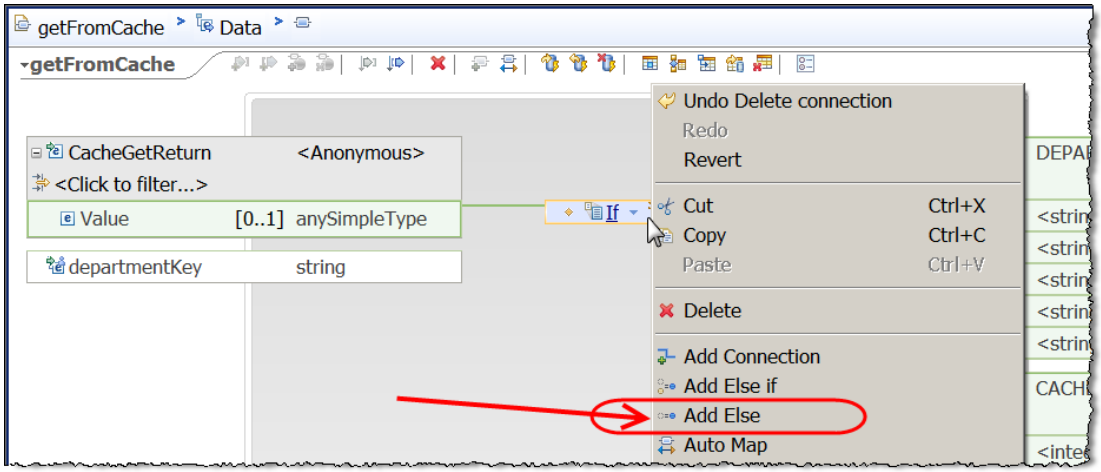
   When complete click the yellow up arrow to return to the previous nested map level.

3.

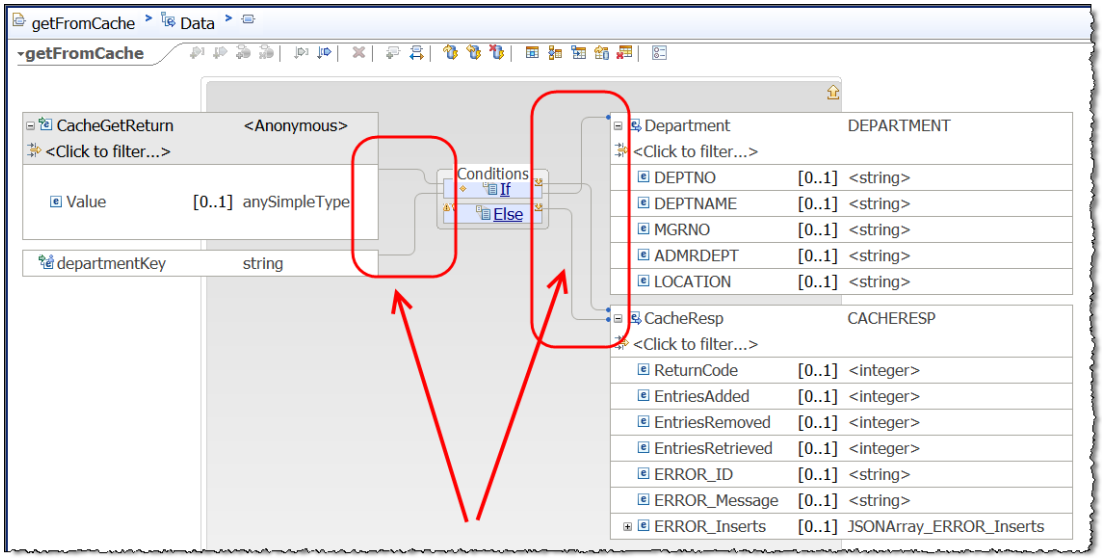
6.3.2 Configure the Cache Return transform

1. Connect the Cache Return transform to **CacheResp** and to **Department**:



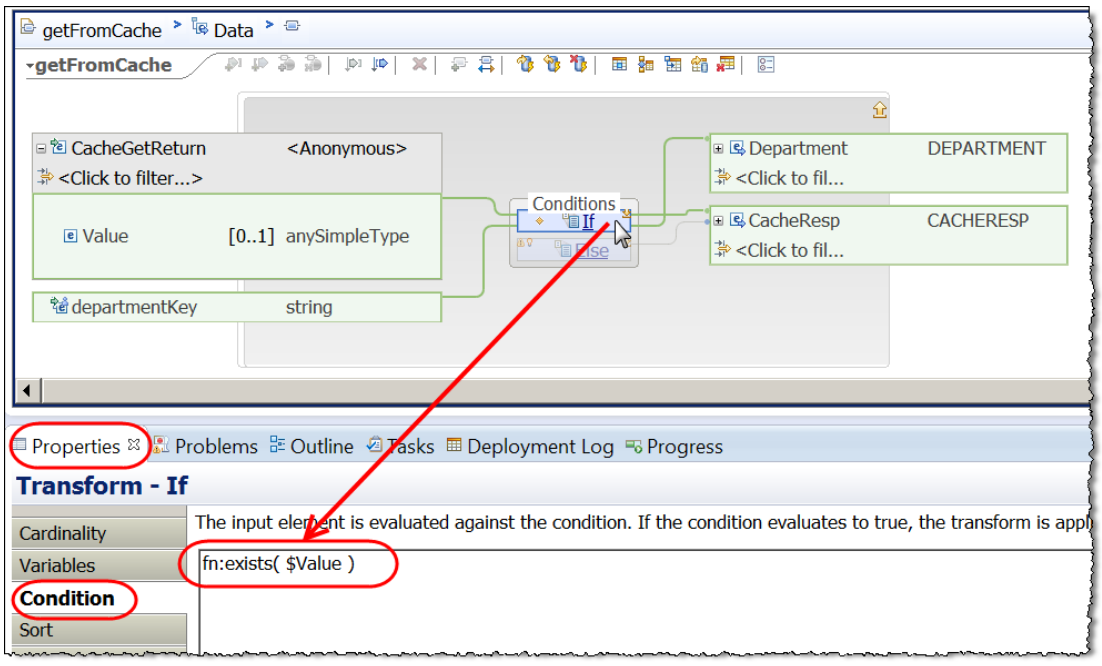
2.	Connect **departmentKey** to **Cache Return** 
3.	Click on Cache Return to go to the Cache Return nested map.
4.	Connect Value to Department. The transform that will be created is a “Local map”, change this to an IF transform: 
5.	Right click on the blue background of the **If** transform and select **Add Else** from the list of options: 

6. Connect the **Else** transform to CacheResp.
Connect the **If** transform to CacheResp.
Connect departmentKey to the **If** transform.



7. Click the blue box surrounding the If Transform. In the Condition tab (*Properties tab*) for the If transform specify (use Type assist <ctrl><Space bar> or just type it):

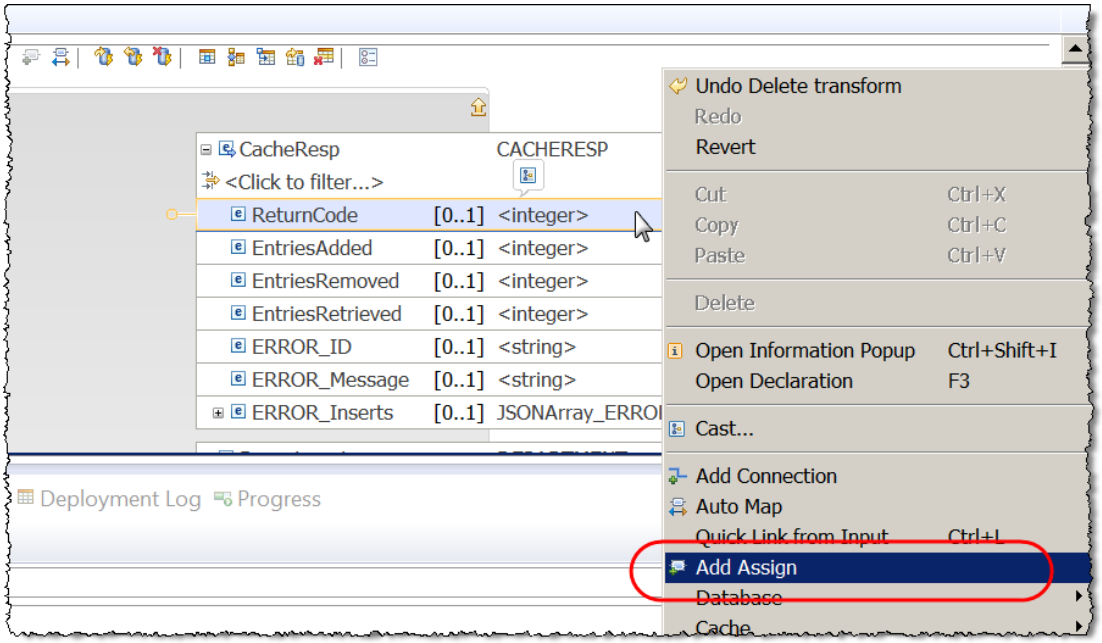
```
fn:exists($Value)
```



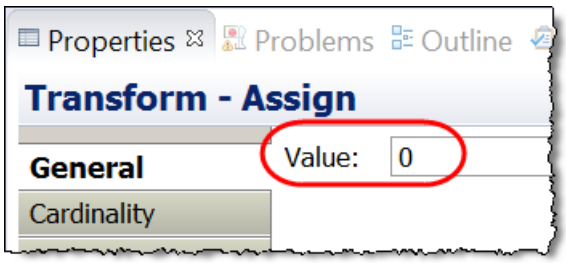
This configuration means – execute the If statement when $Value exists (*ie the Cache Get transform has returned a value*).

8. Click the **If** transform to configure what to do when the condition is true.

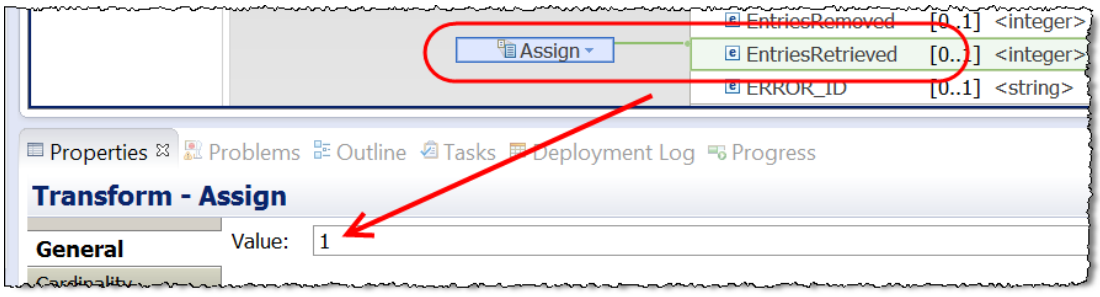
9. Right click on ReturnCode and choose “Add Assign” from the list of options:



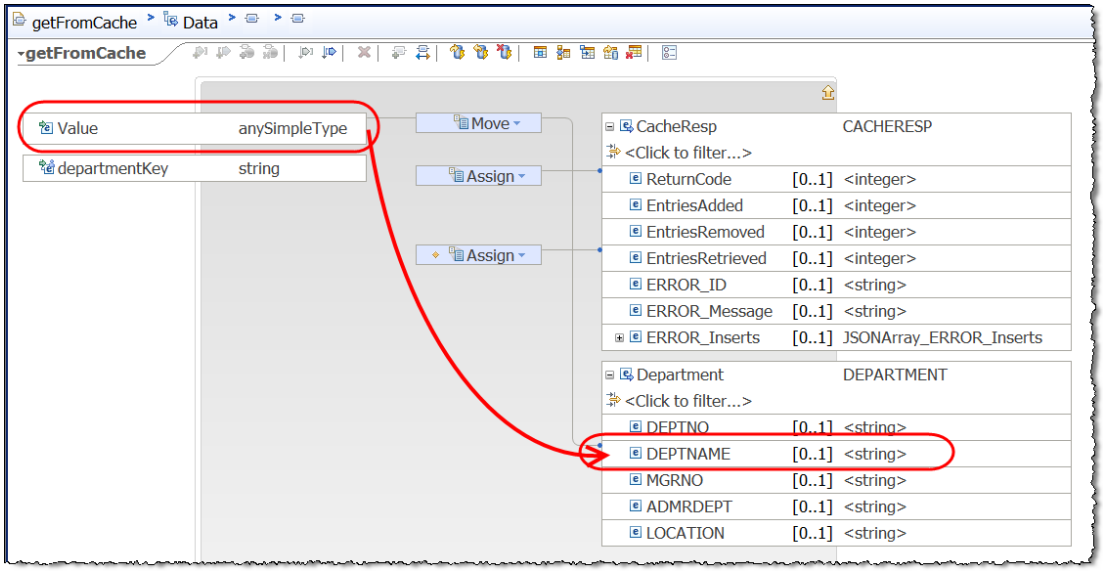
10. In the properties field for the Assign transform, set the Value to numeric zero (0):



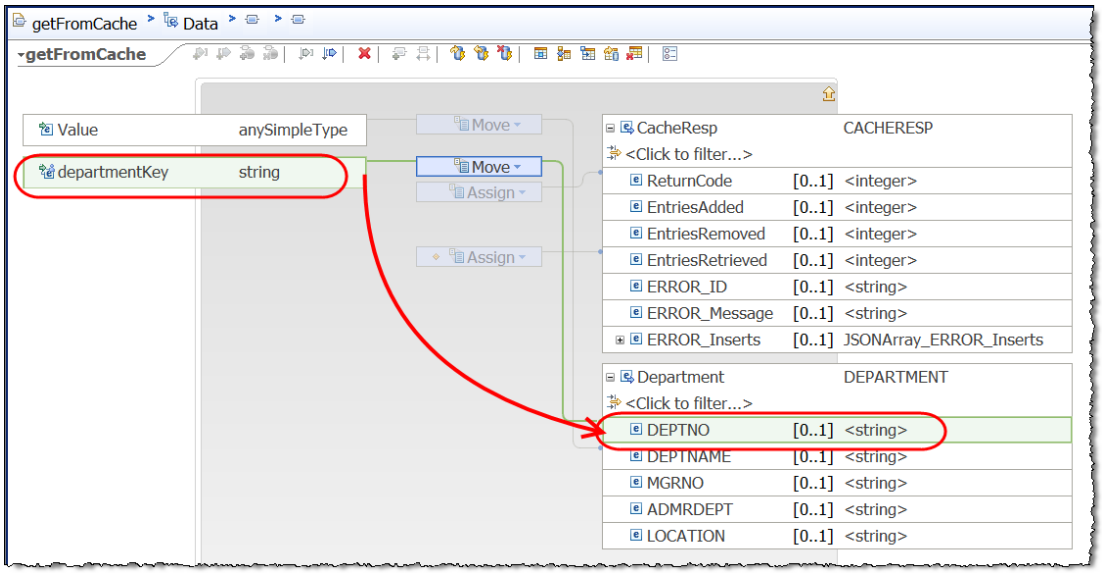
11. Repeat the above two steps and add an assign value of 1 to CacheResp.EntriesRetrieved:



12. Connect **Value** to **Department.DEPTNAME**:



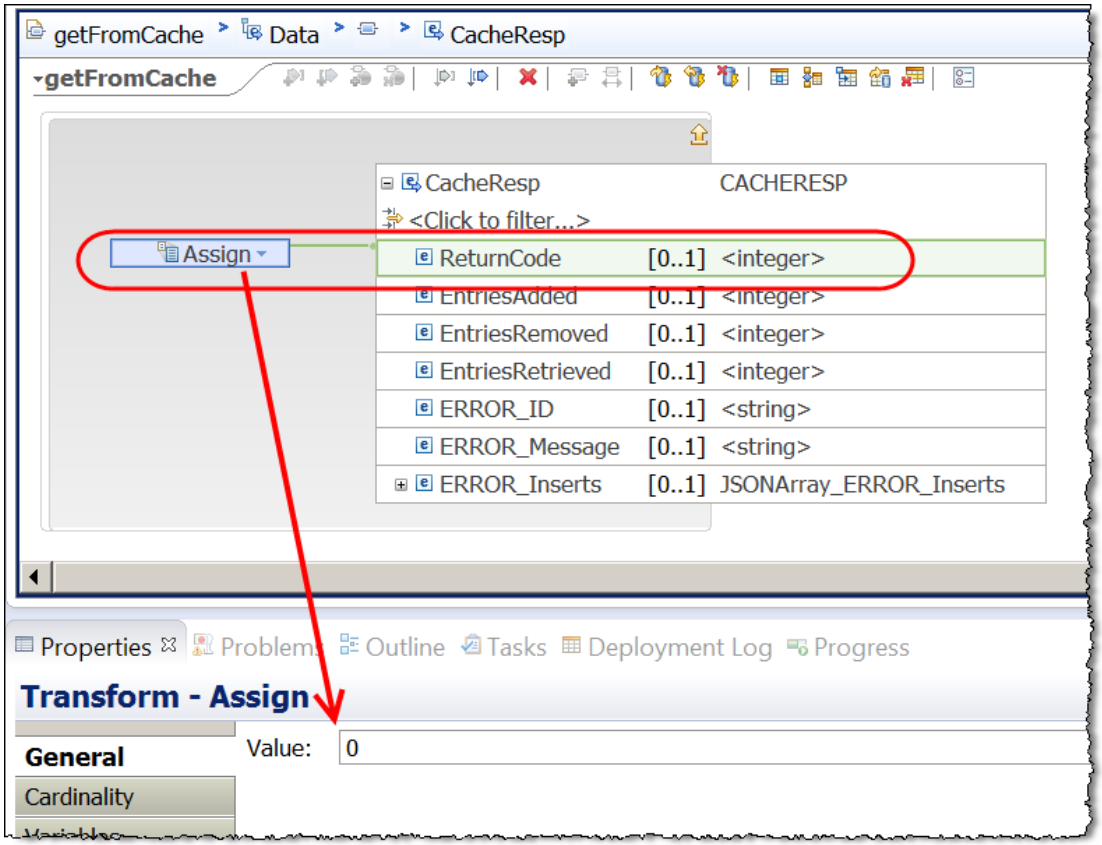
13. Connect departmentKey to DEPTNO:



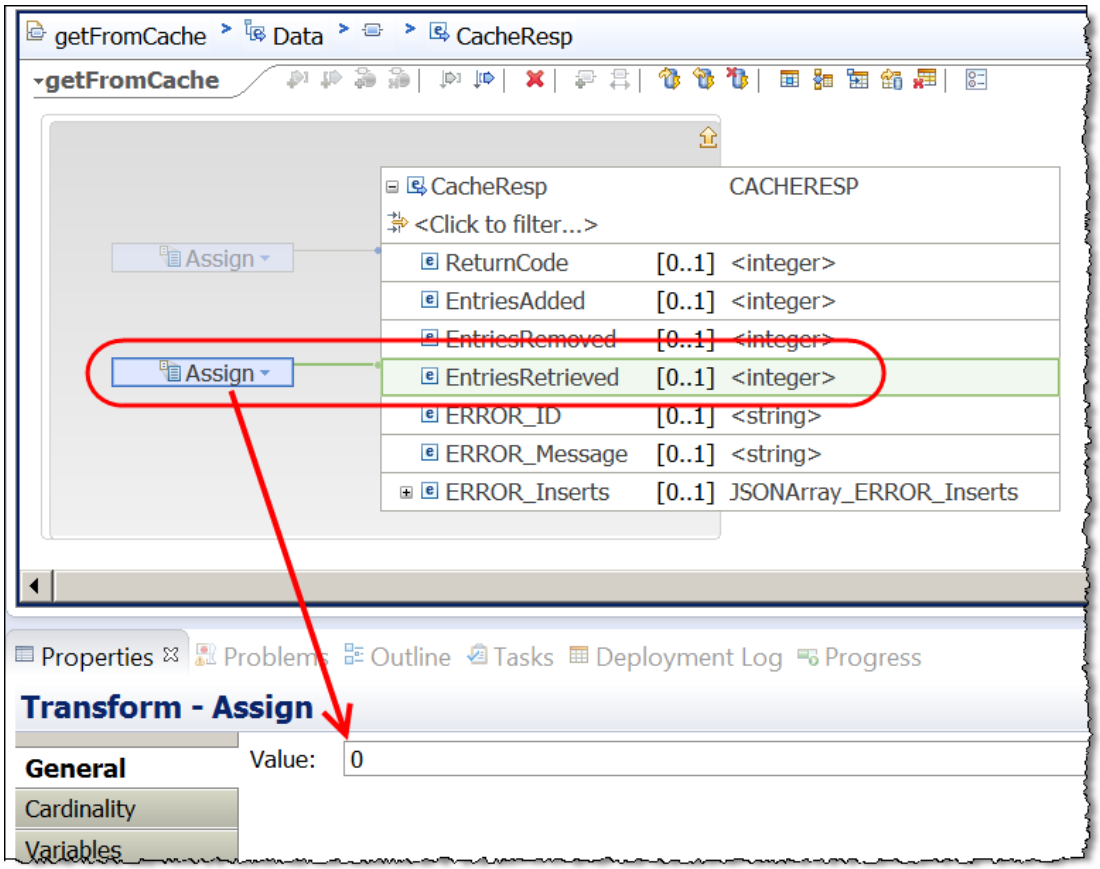
14. Click the yellow Up Arrow to return to the previous level:

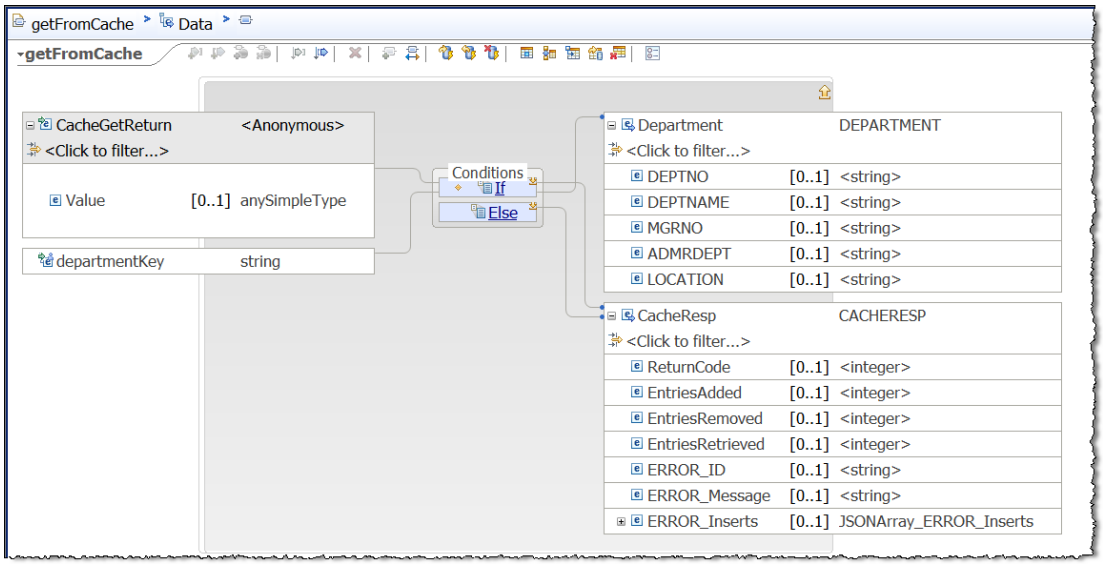
15. Click the Else Transform to configure what happens when $Value does not exist

16. Add an Assign transform to **CacheResp.ReturnCode** with a value of numeric zero (0):



17. Add an Assign transform to EntriesRetrieved and set the value to numeric zero (0):

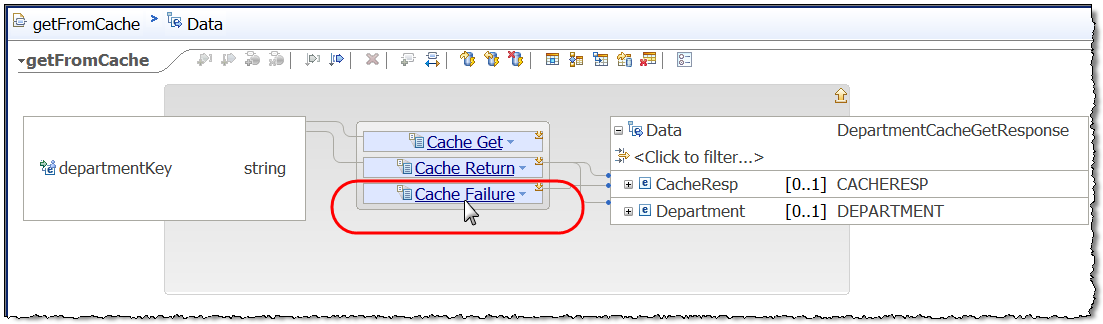


18.	Click the yellow up arrow to return to the previous level. The Cache Return transform will look like this when complete: 
19.	Click the yellow up arrow to return to the nested map with the Cache transforms.

6.3.3 Configure the Cache Failure transform

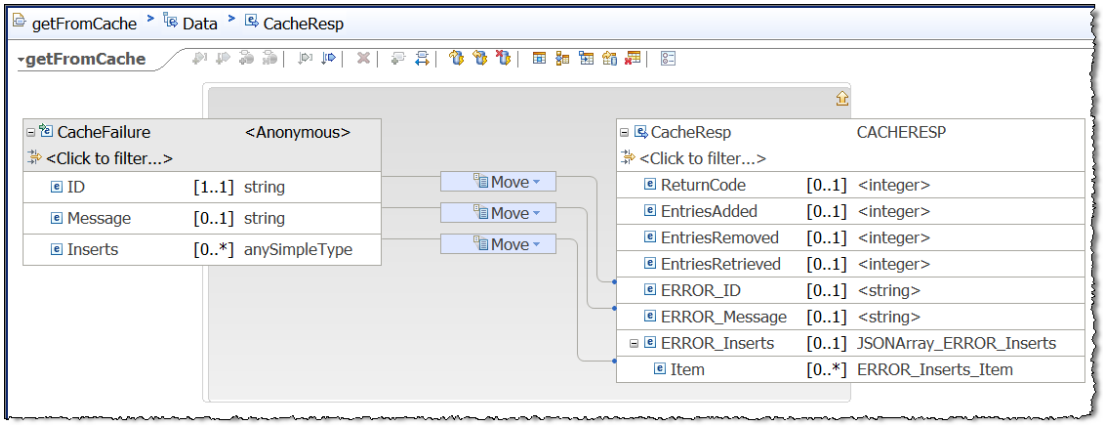
In this section you will configure the processing that will happen when the Cache Failure transform is executed.

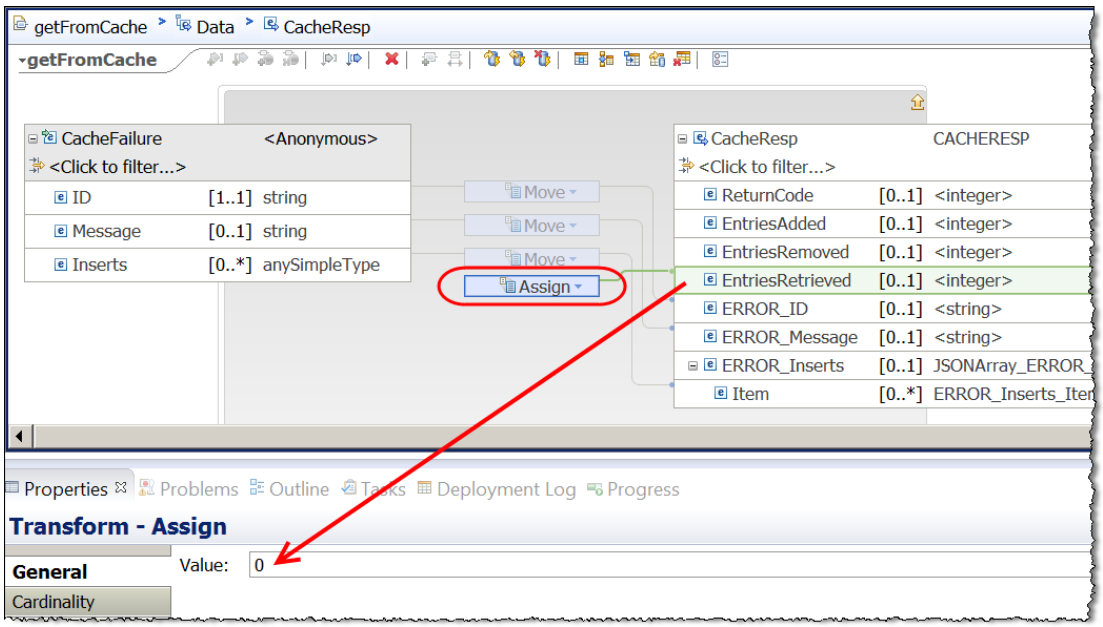
1. Click the Cache Failure text:



2. Connect the following:

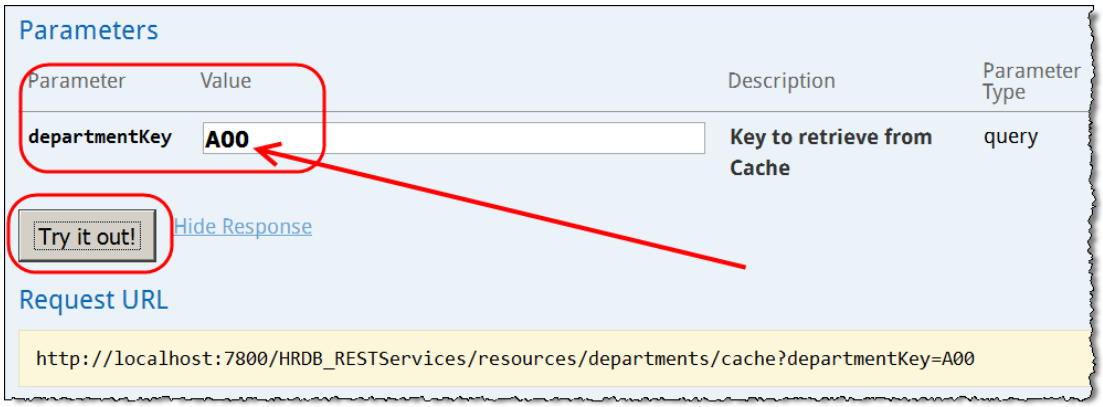
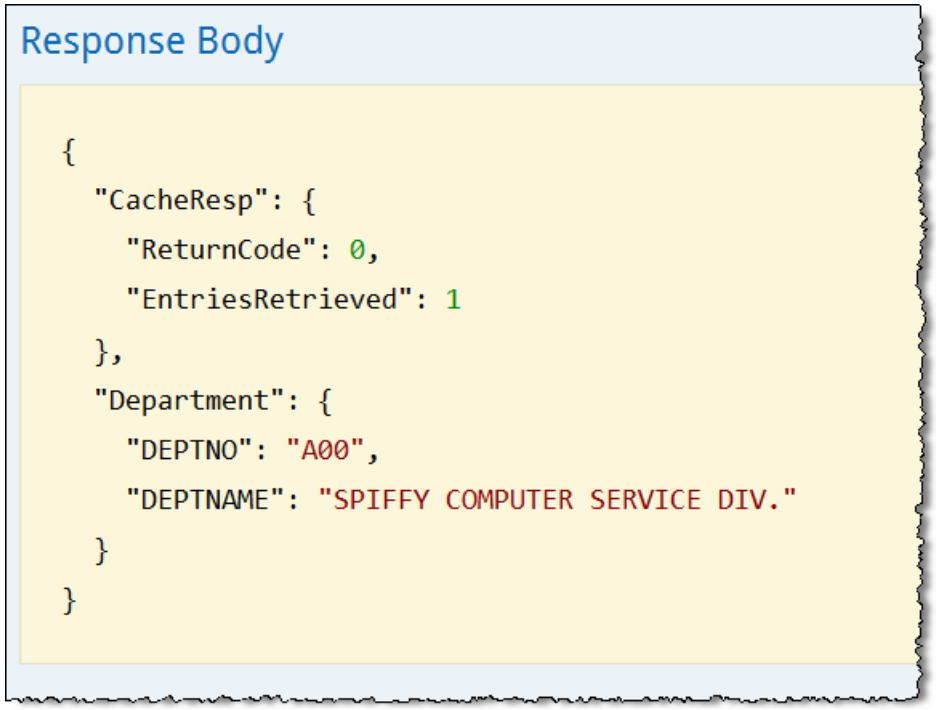
   a) CacheFailure.ID to CacheResp.ERROR_ID
   b) CacheFailure.Message to CacheResp.ERROR_Message
   c) CacheFailure.Inserts to CacheResp.ERROR_Inserts.Item



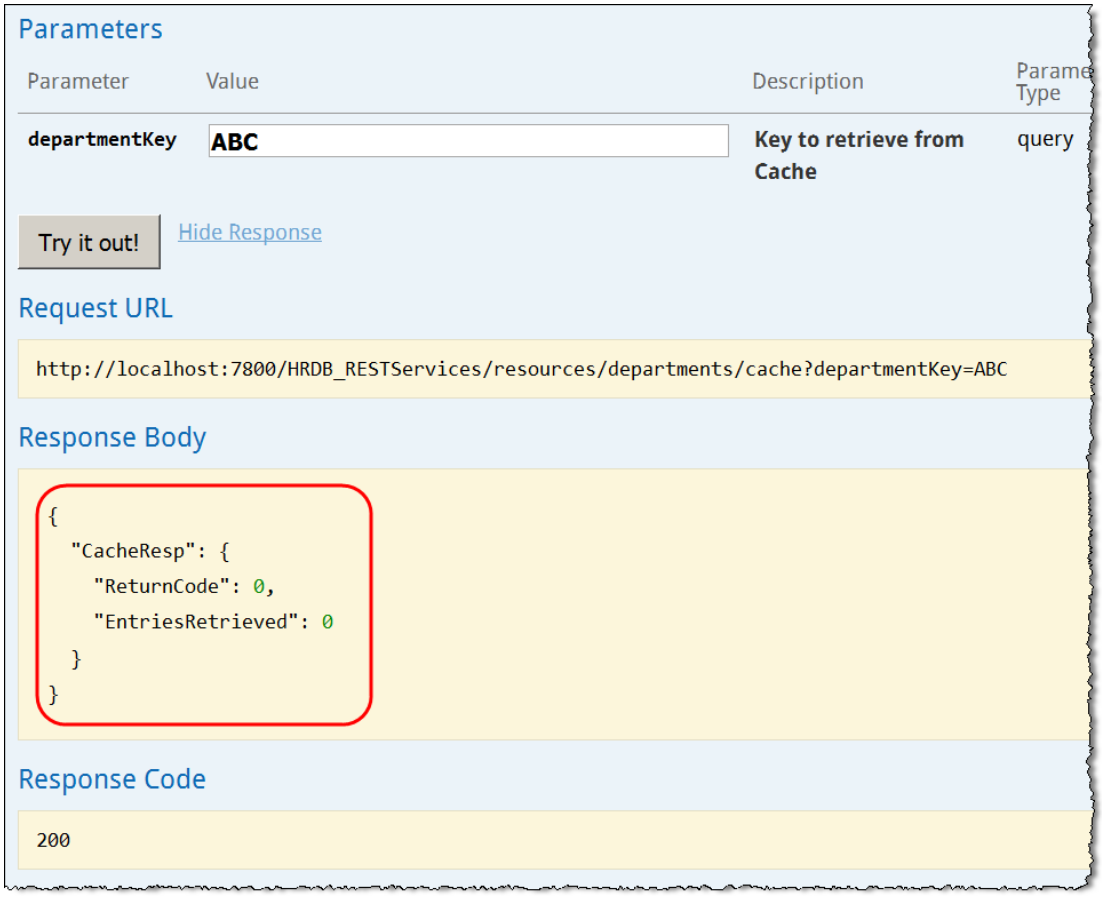
3.	Add an Assign transform to CacheResp.EntriesRetrieved with a numeric value of 0 (zero) 
4.	Add an assign transform to CacheResp.ReturnCode, assign a value of -1
5.	Save (ctrl S) and close the getFromCache map.

7. Test the getFromCache operation

7.1 Test the Cache Get and Cache_Return transform logic

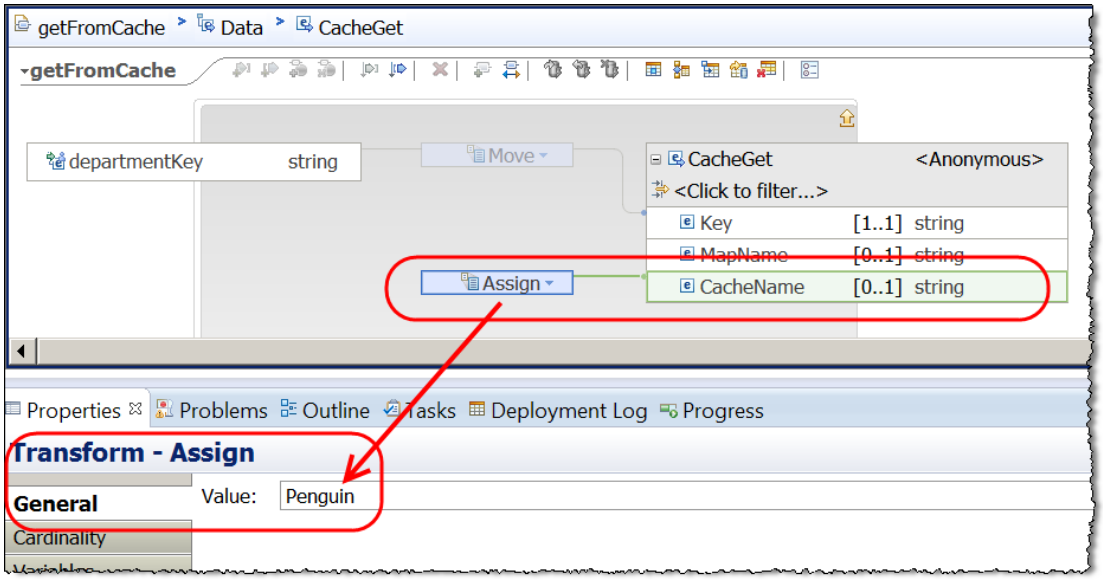
1.	Deploy the HR_Service to the default Integration Server on TESTNODE_iibuser
2.	In SwaggerUI, expand the GET operation for /departments/cache. Scroll to the Parameters section and specify a departmentKey value of A00 and click the Try it out button: 
3.	After a few seconds the Response Body section should show the following: 
4.	Rerun the test with a departmentKey value of **ABC** (*this entry will not exist in the cache*).

5. The Response Body will show zero entries retrieved:



7.2 Test the Cache Failure transform logic

In this section you will test the processing logic when the Cache Failure transform is called.

1.	Open the getFromCache map. Navigate to the nested map where the Cache Get transform is configured (in the For each transform > Cache Get).
2.	Add an Assign transform to CacheGet.CacheName. Assign a value of “Penguin” (*or a cache name that does not exist in your environment*). 
3.	Repeat the above process to add an Assign value of “**Bear**” to the CacheGet.**MapName**
4.	Save the Map and redeploy the HR_Service
5.	In Swagger UI click the Try it out button again to attempt to obtain the details for a key (any key will do)

6.	In the response Body you will see the following (this is returned via the Cache Failure transform):
	
	Response Body
	{
	"CacheResp": {
	"ReturnCode": -1,
	"EntriesRetrieved": 0,
	"ERROR_ID": "7182",
	"ERROR_Message": "[BIPmsgs:7182]BIP7182E: The WebSphere eXtreme Scale configurable service 'Penguin' cannot be found,
	"ERROR_Inserts": [
	"Penguin"
	]
	}
	}

END OF LAB GUIDE