IBM Integration Bus

Distributing IIB Workload using Docker Containers and Callable Flows

Featuring:

- Configuration of IIB runtime environment in a Docker Container
- Running a pre-built REST API in a Docker Container
- Configuration of an IIB Switch server and Connectivity agent to enable a REST API running in a Docker Container to access callable message flows running in a Windows environment

September 2016

Hands-on lab built at product Version 10.0.0.6

1. Introduction

IIB V10.0.0.4 introduced Callable Flows. This enables a message flow (or integration service, or REST API) to invoke a separate message flow in a call/return (blocked wait) programming model.
The calling and called message flows could operate on remote Integration Nodes, including one on IIB on Cloud (IIBoC) and one On Premise.

In this lab you will explore the Callable Flows feature using two Integration Bus Nodes. The first Integration Node is your Windows installation. The second Integration Node is started in a Docker Container, also hosted within your local Windows system.

1.1 Scenario Overview

For this lab, you will need a Docker installation in order to create the Docker container. Please note that you can have your Docker container running on the same machine where IIB is installed or on a remote host.

- If you are using the IIB10006 2016 Workshop VMWare Image, Docker has already been installed for you. The hostname of this VMWare image Windows system is BETAWORKS-ESB10.

- This lab guide assumes that the Docker container is hosted by the windows environment you are using to perform the tasks in this lab guide (ie. the Docker environment runs "inside" the Windows environment). The Workshop VM is configured in this way.

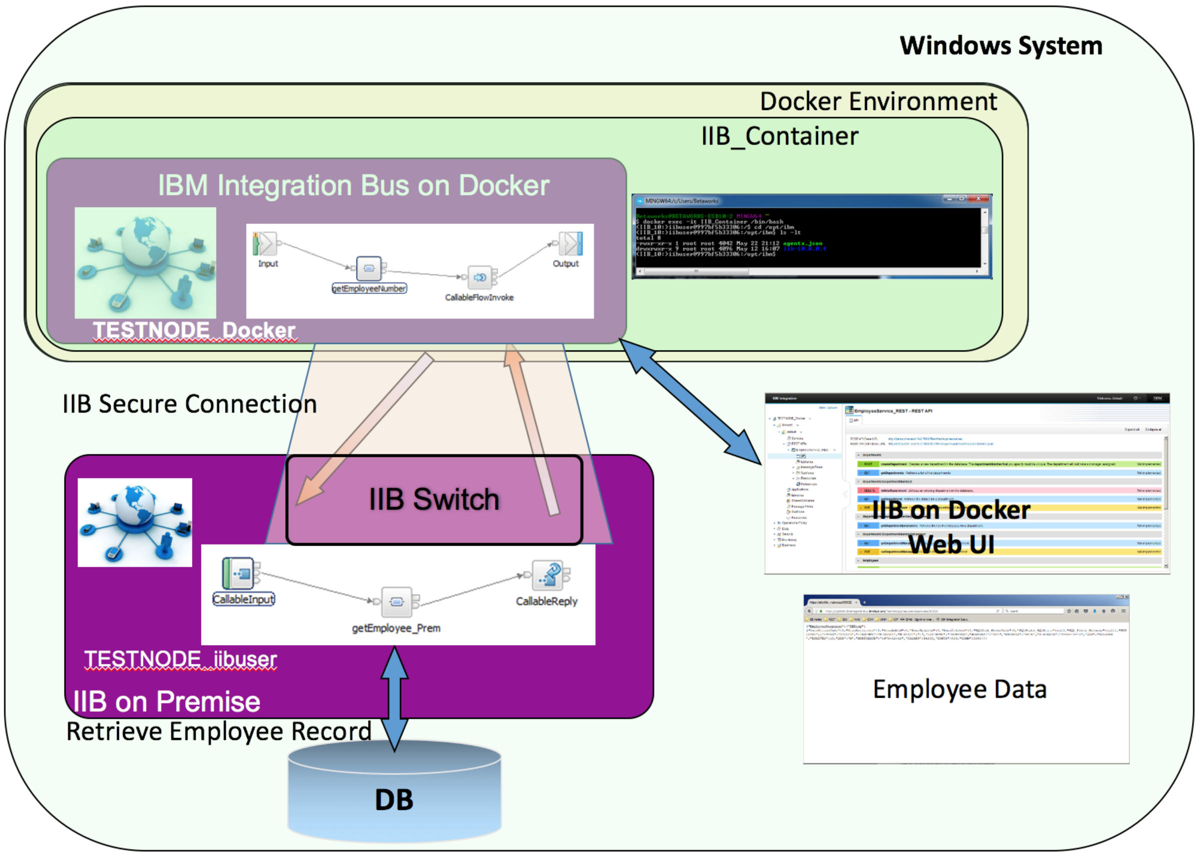
You will deploy and run the REST API HR_Service (that you created in the Callable Flows development lab 16L10) in the Docker Container. HR_Service will call the HR_Service_CallableApplication (created in lab 16L10) application, which runs on your local Integration Node on the hosted Windows environment. HR_Service_CallableApplication will access a database, located on the same host as its Integration Node, and retrieve employee details from the HRDB database.

Please Note: Throughout this lab you will see references to two IIB installations as follows:

- **IIB** – this is the Integration Node on the Windows environment used to host the Docker Container - **BETAWORKS-ESB10**
- **Docker IIB –** this is the Integration Node running in a Docker container – **BETAWORKS-ESB10-DOCKER**

Application Development
The development of the applications involved in this scenario, (both the Calling REST API and the Callable Application), was done in 'Callable Flows Development' lab 16L10.

A pre-built solution of these applications is provided for this lab.



1.2 Outline of tasks

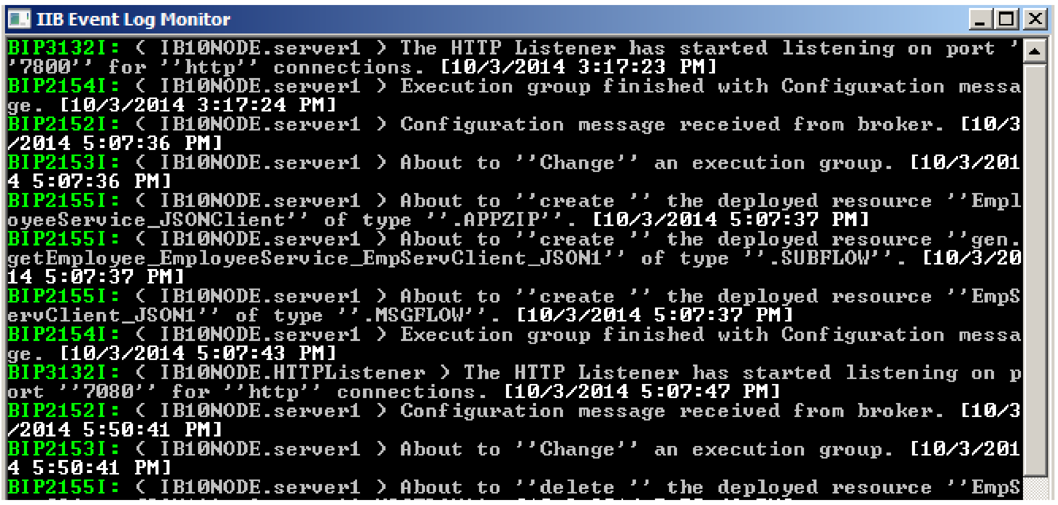
The tasks to complete in this lab are the following. You will start by downloading a set of files from the OT4I GitHub repository that will enable you to build a Docker image containing the runtime component of IIB Developer edition. You will then use this image to start a Docker Container so that you can deploy and run applications in the IIB runtime environment. Ubuntu is used as the OS by the Docker Container.
You will manage the IIB runtime environment running in the Docker container from the windows environment.

1. Create a Docker image with the IIB runtime environment installed. Use this image to start a Docker Container with a running Integration Node/Server to enable you to deploy and run applications.
2. View Web User interface for IIB node running in the Docker Container and deploy REST API
3. Import and deploy the Callable application to the IIB environment running in Windows
4. Create an IIB Switch on IIB on Windows
5. Set up the IIB runtime environment running in the Docker container to access the Switch on IIB on Windows

1.3 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.



IIB Event Log Monitor

```
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.

<u>Important note</u>

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

1.4 Configure Integration Bus node to work with DB2

Note 1

Login to the Windows VMWare Image as the user "iibuser", password = "passw0rd".

Start the IIB Toolkit from the Start menu.

Note 2

If you have already done Lab 1 in this series (create a REST API), you can skip straight to the next page.

The HRDB database, and the EMPLOYEE and DEPARTMENT tables have already been created on the supplied VMWare image. If you wish to recreate your own instance of this database, the commands

`1_Create_HRDB_database.cmd` and

`2_Create_HRDB_Tables.cmd`

are provided for this. If used in conjunction with the VM image, these commands must be run under the user "**`iibadmin`**". Appropriate database permissions are included in the scripts to GRANT access to the user **`iibuser`**.

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

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

   ```
   c:\student10\Create_HR_database
   ```

2. Run the command

   ```
   3_Create_JDBC_for_HRDB
   ```

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

3. Run the command

   ```
   4_Create_HRDB_SecurityID
   ```

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

   ```
   mqsistop TESTNODE_iibuser
   
   mqsistart TESTNODE_iibuser
   ```

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

2. IIB Docker Container

In this section you will download the relevant files from GitHub and create a Docker image with the IIB runtime installed. You will then use the Docker image to run IIB in a Docker container. You will access the IIB runtime environment running in the Docker container from your windows environment using the web UI.

From the Web UI, you will deploy a BAR file which contains the HR_Service REST API.

You may have the REST API and the Callable application in your Toolkit workspace if you have just completed the 'Callable Flows Development' lab. However, in this lab, please use the provided REST API and Callable application solutions.

2.1 Create the Docker image and start the IIB Container

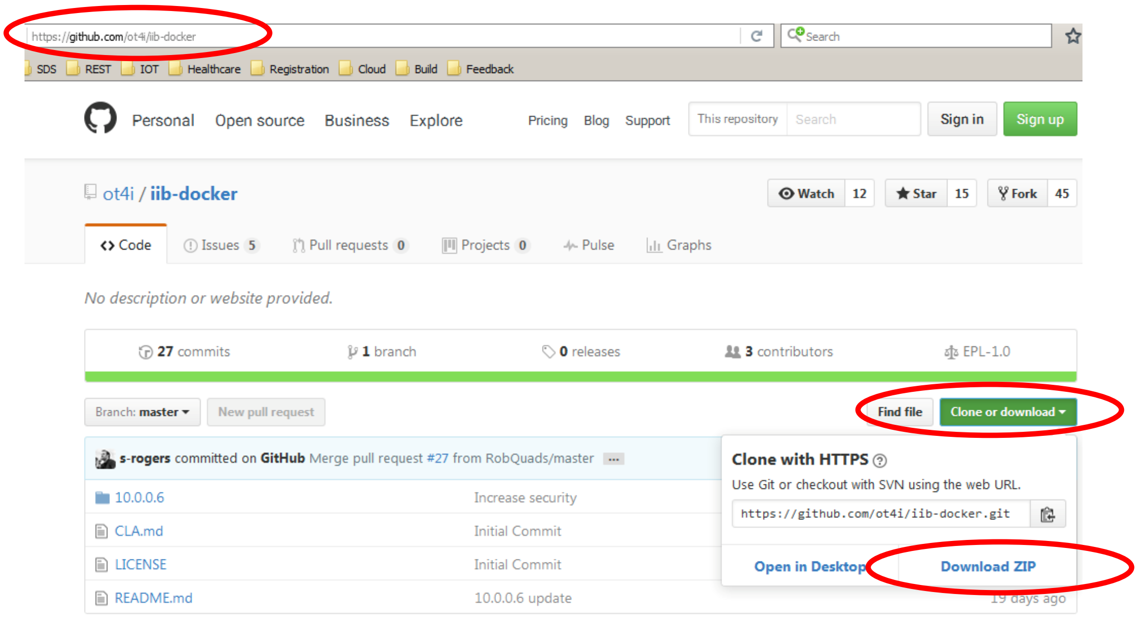
You will create the IIB Docker image based on IBM Integration Bus for Developers Edition V10.0.0.6.

The IIB Development team maintains a GitHub repository which contains a Dockerfile and scripts, which demonstrate how you might run IBM Integration Bus in a Docker container.

1. Open a new Browser window and navigate to the web page:

   https://github.com/ot4i/iib-docker

   Click on 'Clone or download' -> 'Download ZIP'.



No description or website provided.

s-rogers committed on GitHub Merge pull request #27 from RobQuads/master

10.0.0.6	Increase security
CLA.md	Initial Commit
LICENSE	Initial Commit
README.md	10.0.0.6 update

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

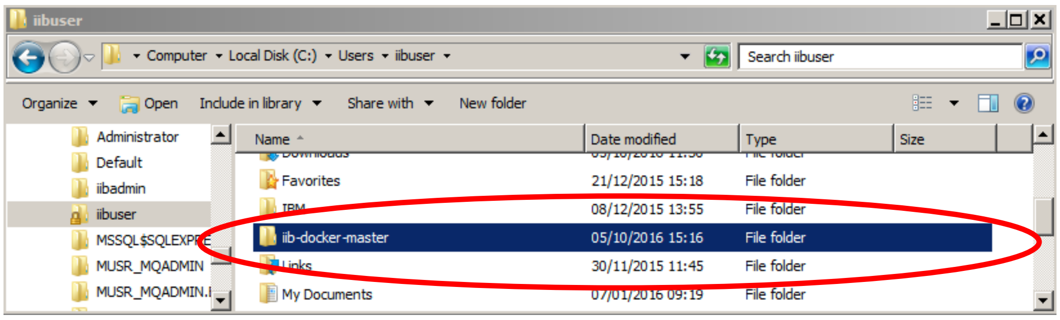
https://github.com/ot4i/iib-docker.git

On the next prompt, select 'Save File' to save the resources to your machine.

This file contains the Docker configuration and scripts to create a sample Docker image with an IIB runtime environment running in the default (Ubuntu) Docker container.

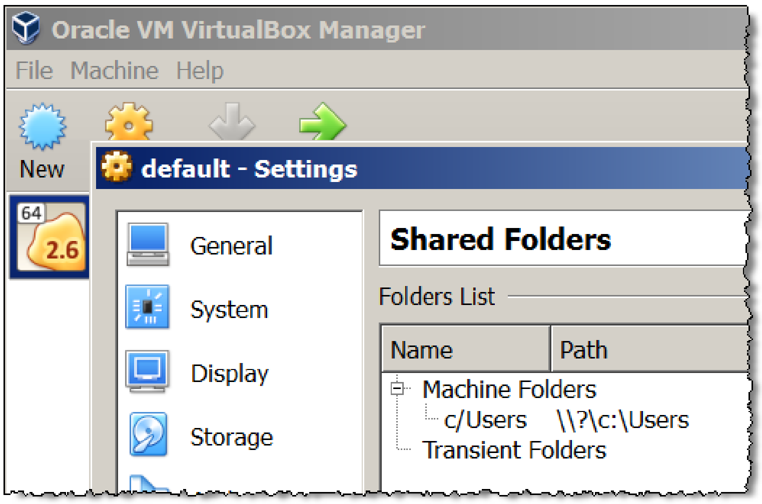
This does not contain the IIB Developer Edition itself. When you run the scripts, this is downloaded from a public IBM server.

2. If you are using the provided IIB workshop system, the download directory is **C:\Users\iibuser\Downloads**. Copy the downloaded zip file to **C:\Users\iibuser** and unzip. This folder is important, because it is referenced by the Docker VM later in the lab.



Name	Date modified	Type	Size
Favorites	21/12/2015 15:18	File folder	
IBM	08/12/2015 13:55	File folder	
iib-docker-master	05/10/2016 15:16	File folder	
Links	30/11/2015 11:45	File folder	
My Documents	07/01/2016 09:19	File folder	

3. This folder is important, because it is referenced by the Ubuntu Docker VM. In the VirtualBox settings, "Shared Folders" tab shows that by default c:\users is defined as a shared folder in the Ubuntu VM.



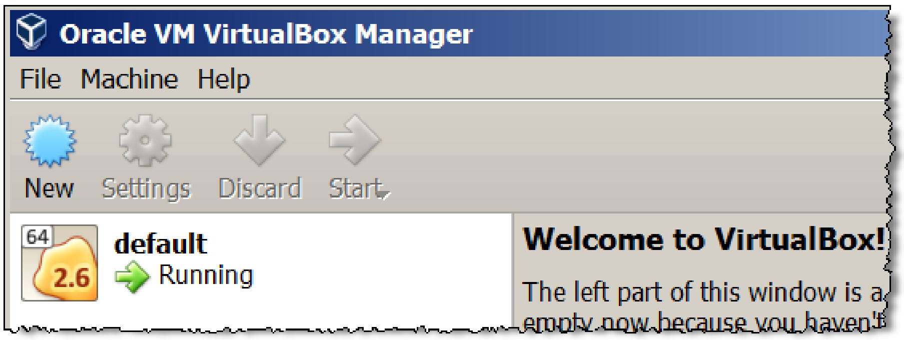
Name	Path
Machine Folders	
c/Users	\\?\c:\Users
Transient Folders	

4. Open a Docker command prompt by double-clicking on the icon

   Docker Quickstart Terminal

   This Docker Quickstart Terminal will complete a few steps for you:
   - Opens a terminal window
   - Creates a default VM if it doesn't exist, and starts the VM. This VM is used by Docker as its Operating System. In this environment, Docker Containers use the OS in this VM to run. Docker Toolbox installs VirtualBox to manage and run this VM. The IIB Docker Container will run inside this Virtual Box maintained Ubuntu environment.
   - Points the terminal environment to the Ubuntu Linux VM

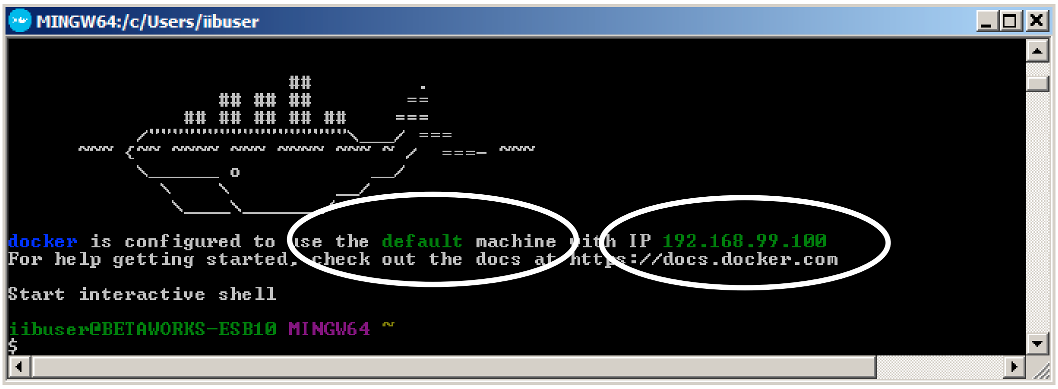
   Open VirtualBox Manager (you may have it open already from the previous step). You will see that the default Ubuntu VM is now running.


Oracle VM VirtualBox Manager
File Machine Help
New Settings Discard Start
64
2.6
default
Running
Welcome to VirtualBox!
The left part of this window is a
empty now because you haven't

   The Docker technology is not the focus in this lab and you can find more at ***https://www.docker.com.***

5. When the terminal is opened, you will see the details of the Docker VM. In this example, the IP address of the Docker VM is 192.168.99.100.

   - Name: **`default`**
   - IP: **`192.168.99.100`**

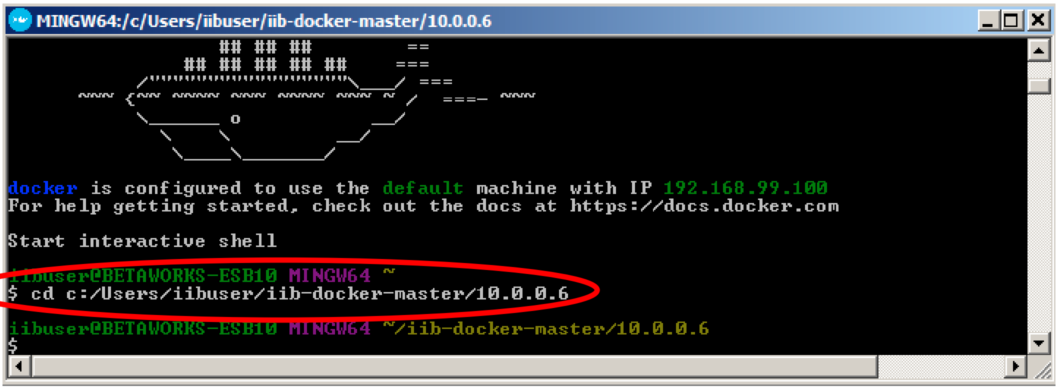


```
docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com

Start interactive shell

iibuser@BETAWORKS-ESB10 MINGW64 ~
$
```

6. Navigate to the downloaded folder 10.0.0.6:

   ```
   cd c:/Users/iibuser/iib-docker-master/10.0.0.6
   ```

   (remember this folder is shared between Ubuntu and Windows)

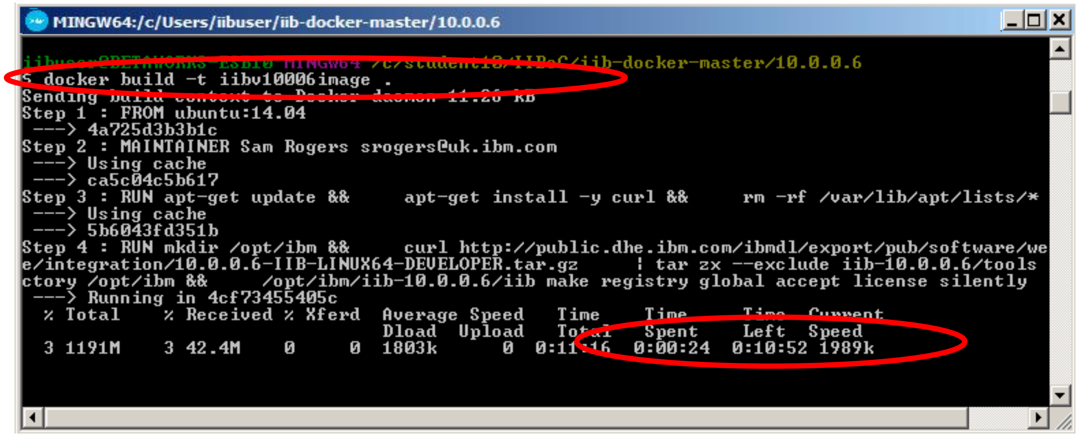


```
MINGW64:/c/Users/iibuser/iib-docker-master/10.0.0.6
                        ## ## ##        ==
                     ## ## ## ## ##    ===
                 /"""""""""""""""""""\___/ ===
            ~~~ {~~ ~~~~ ~~~ ~~~~ ~~~ ~ /  ===- ~~~
                 \______ o          __/
                   \    \        __/
                    \____\_______/

docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com

Start interactive shell

iibuser@BETAWORKS-ESB10 MINGW64 ~
$ cd c:/Users/iibuser/iib-docker-master/10.0.0.6

iibuser@BETAWORKS-ESB10 MINGW64 ~/iib-docker-master/10.0.0.6
$
```

7. Run the following command (**note** the 'full stop' at the end, which is required).

   ```
   docker build -t iibv10006image .
   ```

   The command performs the tasks outlined in the Dockerfile (in the zip file you just downloaded from GitHub). One of the tasks is to download the installation files for IIB Developer Edition from an IBM Server. Depending on your connection speed this make some time.

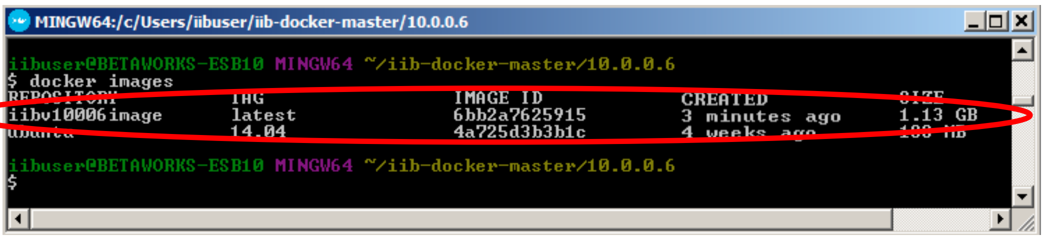


```
MINGW64:/c/Users/iibuser/iib-docker-master/10.0.0.6
iibuser@BETAWORKS-ESB10 MINGW64 /c/student10/IIBoC/iib-docker-master/10.0.0.6
$ docker build -t iibv10006image .
Sending build context to Docker daemon 11.26 kB
Step 1 : FROM ubuntu:14.04
 ---> 4a725d3b3b1c
Step 2 : MAINTAINER Sam Rogers srogers@uk.ibm.com
 ---> Using cache
 ---> ca5c04c5b617
Step 3 : RUN apt-get update &&     apt-get install -y curl &&     rm -rf /var/lib/apt/lists/*
 ---> Using cache
 ---> 5b6043fd351b
Step 4 : RUN mkdir /opt/ibm &&     curl http://public.dhe.ibm.com/ibmdl/export/pub/software/we
e/integration/10.0.0.6-IIB-LINUX64-DEVELOPER.tar.gz     | tar zx --exclude iib-10.0.0.6/tools
ctory /opt/ibm &&     /opt/ibm/iib-10.0.0.6/iib make registry global accept license silently
 ---> Running in 4cf73455405c
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  3 1191M    3 42.4M    0     0  1803k      0  0:11:16  0:00:24  0:10:52 1989k
```

8. Once the command has completed, run the command

   ```
   docker images
   ```

   You will see the IIB Docker image created:



```
MINGW64:/c/Users/iibuser/iib-docker-master/10.0.0.6

iibuser@BETAWORKS-ESB10 MINGW64 ~/iib-docker-master/10.0.0.6
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
iibv10006image      latest              6bb2a7625915        3 minutes ago       1.13 GB
ubuntu              14.04               4a725d3b3b1c        4 weeks ago         188 MB

iibuser@BETAWORKS-ESB10 MINGW64 ~/iib-docker-master/10.0.0.6
$
```

9. When you run the command for creating the IIB container, there are a few arguments that need to be specified:

    - Provide container's name
    - Define Integration Node's name
    - Accept the terms of the IBM Integration Bus for IIB Developers license
    - Expose required ports
    - Set a hostname for the IIB container

    Run the command below, all on the **same** row (formatted here for readability):

```
docker run
            --name IIB_Container
             -e LICENSE=accept
             -e NODENAME=TESTNODE_Docker
             -p 7800:7800
             -p 4414:4414
             -h BETAWORKS-ESB10-DOCKER
              iibv10006image
```

    After a few seconds you will see that the container has been started and is running in the Ubuntu Linux environment (output in the console as the one below).

```
MQSI 10.0.0.6
/opt/ibm/iib-10.0.0.6/server

----------------------------------------
Version:       '10.0.0.6'
Product:       'IBM Integration Bus'
Build Number: '197'
IE02 level:    'ie02-L20140415-1143'
IB Level:      'ib1000-L160824.197_P'
Server level: 'S1000-L160823.10129'
Toolkit level:'20160819-1112' [not installed]
----------------------------------------
----------------------------------------
Node TESTNODE_Docker does not exist...
Creating node TESTNODE_Docker
BIP8071I: Successful command completion.
----------------------------------------
----------------------------------------
Starting syslog
Starting node TESTNODE_Docker
BIP8096I: Successful command initiation, check the system log to
ensure that the component started w
ithout problem and that it continues to run without problem.
----------------------------------------
----------------------------------------
Running - stop container to exit
```

10. An important point of configuration when running a container from this image is port mapping.

    The Dockerfile exposes ports **`4414`** and **`7800`** by default, for Integration Node administration and Integration Server HTTP traffic respectively. This means you can run with the **`-P`** flag to auto map these ports to ports on your host. Alternatively, you can use **`-p`** to expose and map any ports of your choice.

    In the command above the port mapping has been specified with the flags **`-p`**.

    ```
    -p 7800:7800
    -p 4414:4414
    ```

    This allows you to specify your own ports when connecting to the IIB docker container accessing the Integration Node. More sophisticated port mapping can be configured with VirtualBox, but is beyond the scope of this lab.

11. Confirm that the container has started successfully. Run the command

    ```
    docker ps
    ```

    Use Ctrl-C to be able to type commands in the same prompt.

    When the container is started, the command returns details about the container, including which ports of the IIB Container have been mapped to the host (in this case 4414 to 4414, and 7800 to 7800).

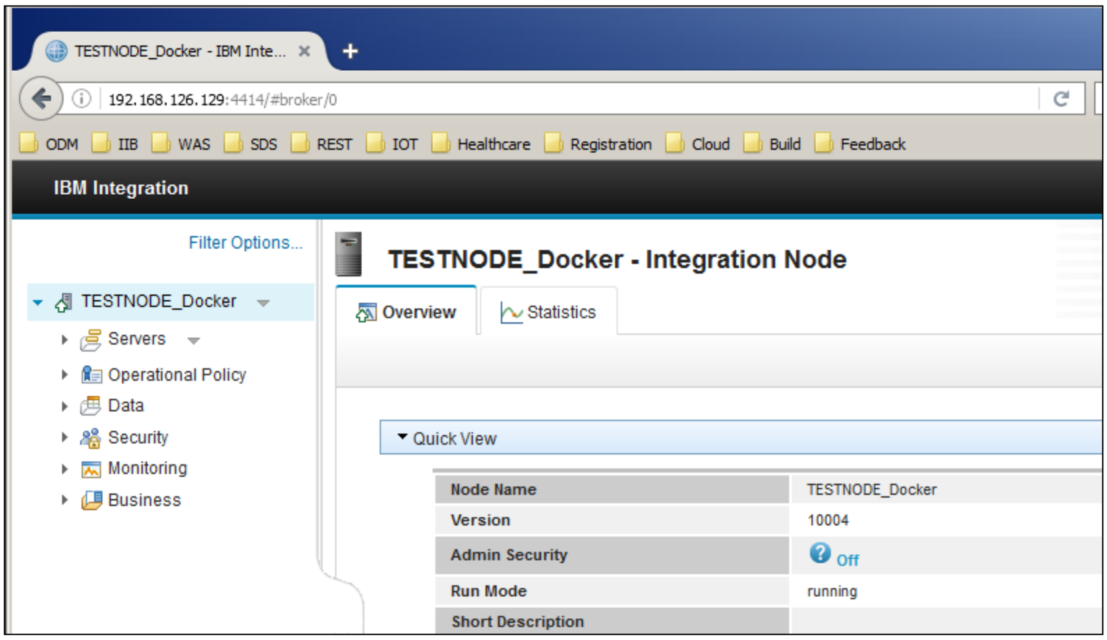
    ```
    CONTAINER ID        IMAGE               COMMAND
    CREATED             STATUS
    PORTS                                            NAMES
    555e7310a378        iibv10006image      "iib_manage.sh"     3
    minutes ago       Up 3 minutes
    0.0.0.0:4414->4414/tcp, 0.0.0.0:7800->7800/tcp   IIB_Container
    ```

2.2 Deploy a REST API to IIB Docker Container

1. In the Windows environment open a browser window and enter the Web UI URL for IIB in Docker:

   `192.168.99.100:4414`

   Note, that if you have previously created Virtual Box images, the assigned IP address may be different.



TESTNODE_Docker - IBM Inte...

192.168.126.129:4414/#broker/0

ODM IIB WAS SDS REST IOT Healthcare Registration Cloud Build Feedback

IBM Integration

Filter Options...

TESTNODE_Docker
- Servers
- Operational Policy
- Data
- Security
- Monitoring
- Business

TESTNODE_Docker - Integration Node

Overview | Statistics

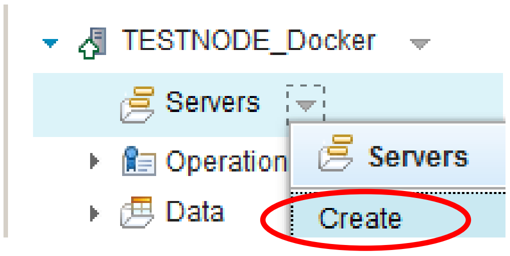
Quick View

Node Name	TESTNODE_Docker
Version	10004
Admin Security	Off
Run Mode	running
Short Description	

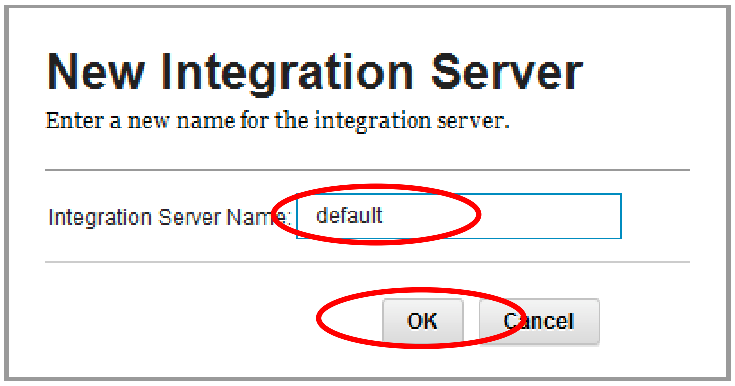
   In the Web UI for the Integration Node you will see that its name is **TESTNODE_Docker**.

2. There will be no default integration server, you will need to create a new integration server in order to run the REST API.

   Click the blue arrow adjacent to Servers, then Create:



3. Name the new integration server ‘default’ and click OK:



New Integration Server

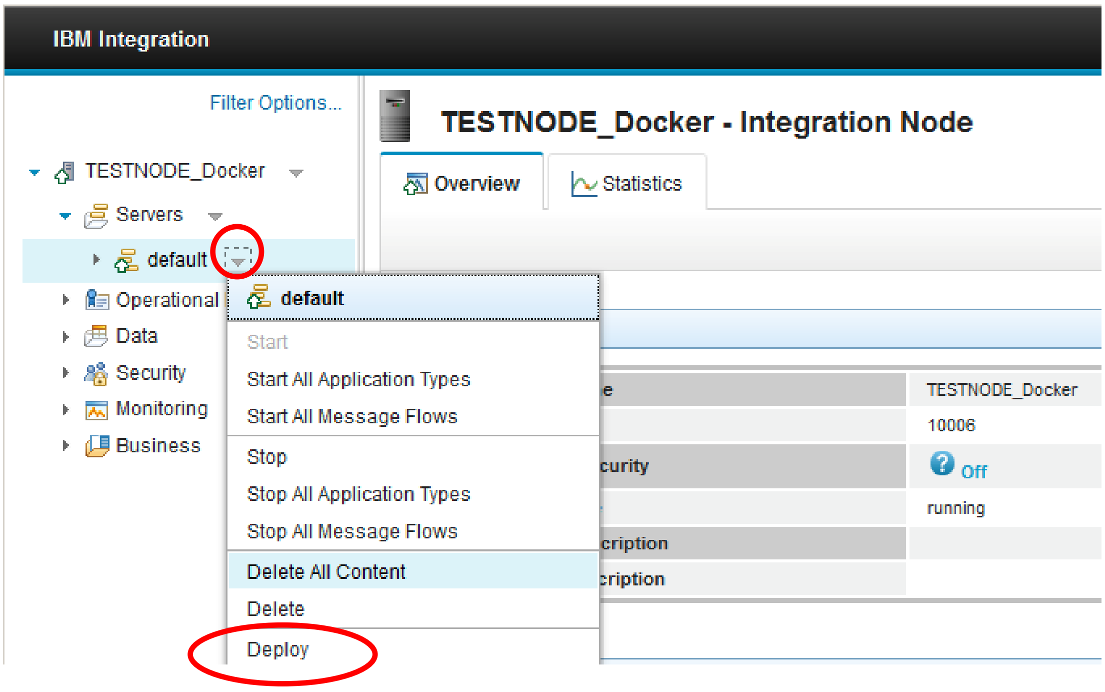
Enter a new name for the integration server.

Integration Server Name: default

4. You will now deploy a REST API using this view.

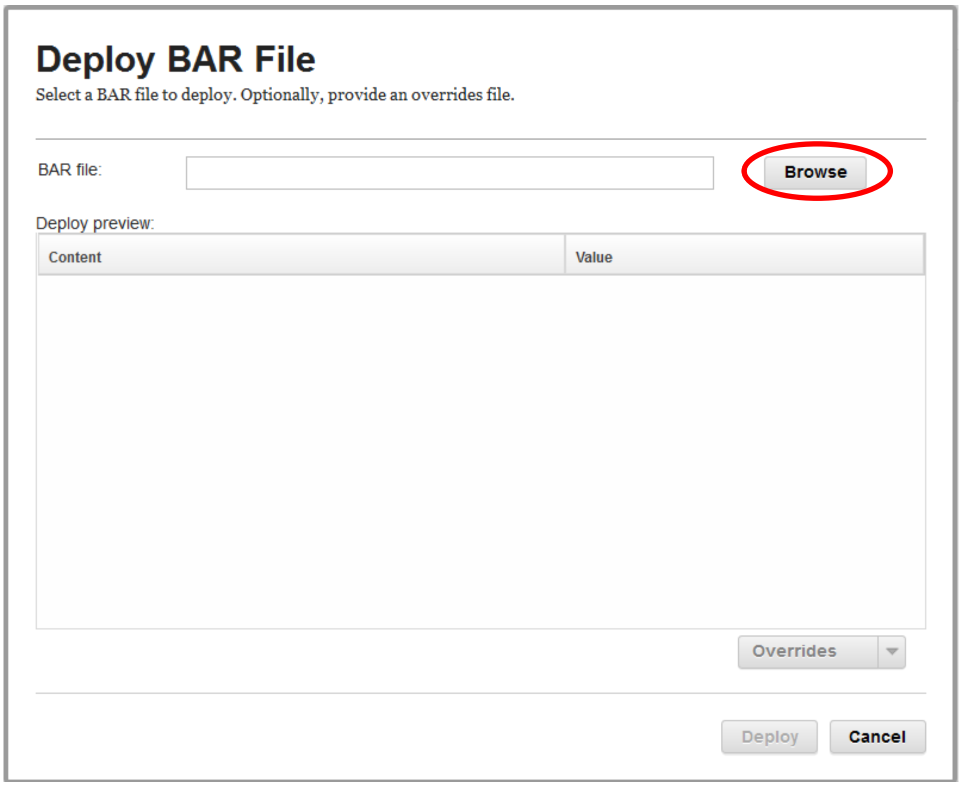
   Click on the arrow next to ‘**default**’ integration server to open its menu (refresh your Browser if you do not see the integration server).

   Click ‘**Deploy**’.



5. A File Upload window is opened.

   Click ‘**Browse**’.



Deploy BAR File

Select a BAR file to deploy. Optionally, provide an overrides file.

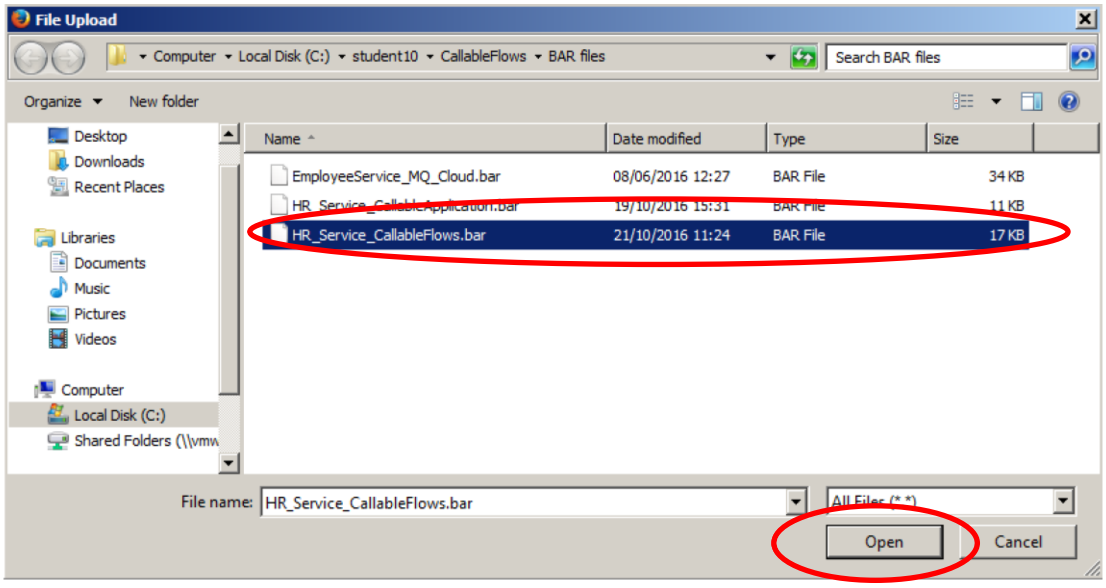
BAR file:

Deploy preview:

Content	Value

6. Navigate to **C:\student10\CallableFlows\BAR files** and select **HR_Service_CallableFlows.bar**.

   Click **Open**.



File Upload

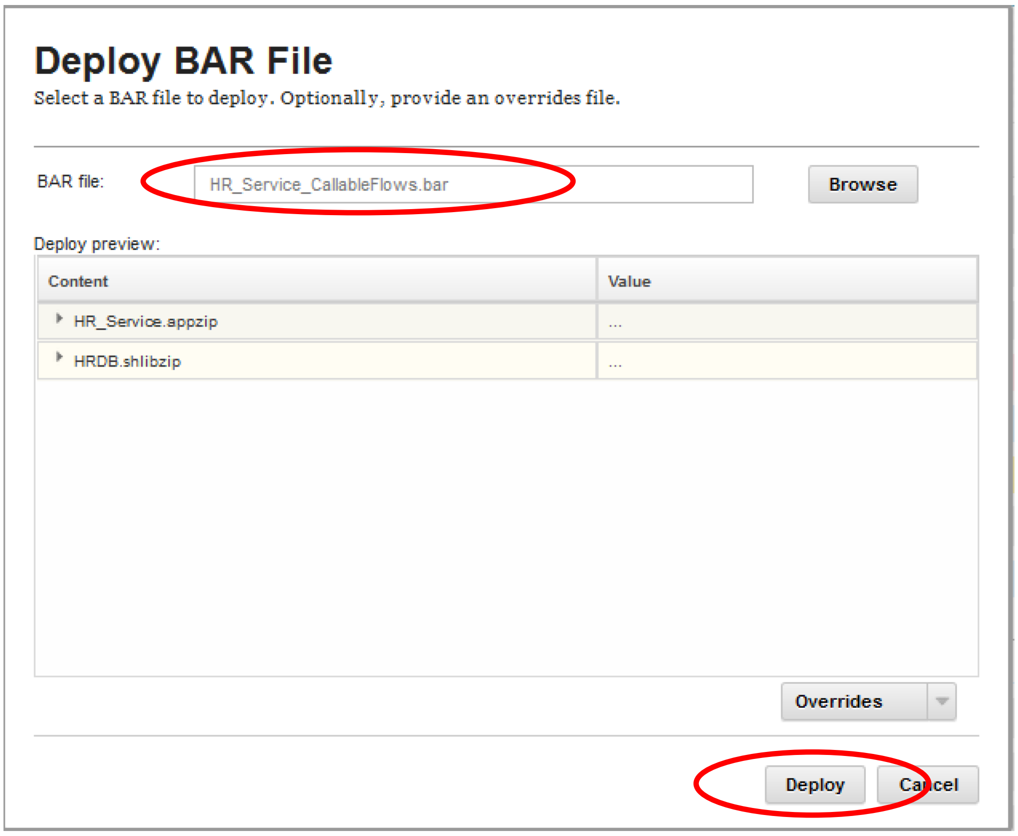
Computer ▸ Local Disk (C:) ▸ student10 ▸ CallableFlows ▸ BAR files

Name	Date modified	Type	Size
EmployeeService_MQ_Cloud.bar	08/06/2016 12:27	BAR File	34 KB
HR_Service_CallableApplication.bar	19/10/2016 15:31	BAR File	11 KB
HR_Service_CallableFlows.bar	21/10/2016 11:24	BAR File	17 KB

File name: HR_Service_CallableFlows.bar

All Files (*.*)

7. Click **Deploy** to upload and deploy the barfile.



Deploy BAR File

Select a BAR file to deploy. Optionally, provide an overrides file.

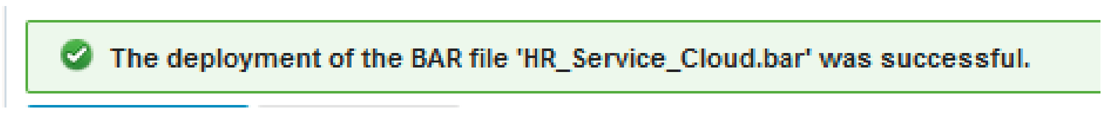
BAR file: HR_Service_CallableFlows.bar

Deploy preview:

Content	Value
HR_Service.appzip	...
HRDB.shlibzip	...

8. When the file has been deployed, you will see a confirmation message.

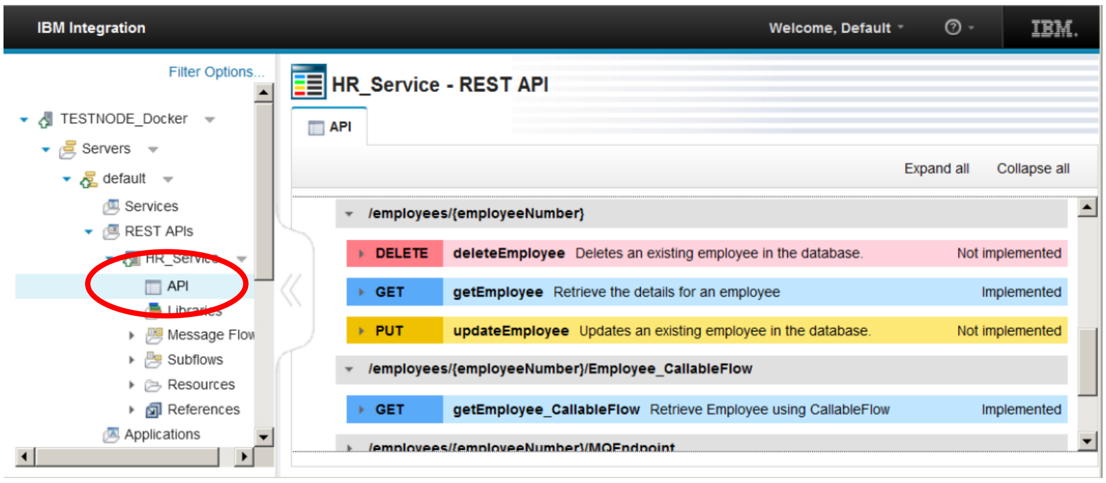
   This is only present for a few seconds, so to make sure you see the message, don't switch away from the browser.



The deployment of the BAR file 'HR_Service_Cloud.bar' was successful.

9. When the REST API has been deployed, you can view the **HR_Service** API and its resources (you may need to refresh the page).

   Expand **HR_Service**, and then click on **API.**



/employees/{employeeNumber}

DELETE	deleteEmployee	Deletes an existing employee in the database.	Not implemented
GET	getEmployee	Retrieve the details for an employee	Implemented
PUT	updateEmployee	Updates an existing employee in the database.	Not implemented

/employees/{employeeNumber}/Employee_CallableFlow

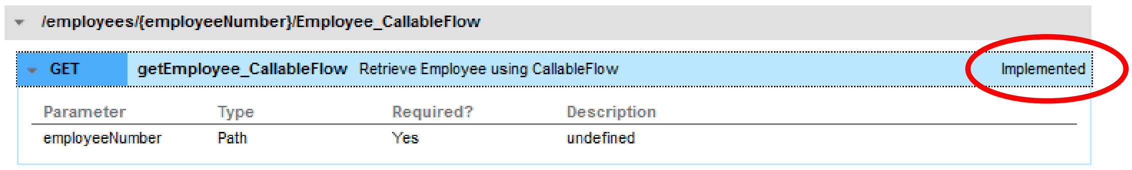
GET	getEmployee_CallableFlow	Retrieve Employee using CallableFlow	Implemented

/employees/{employeeNumber}/MQEndpoint

You will see the REST API resources and corresponding list of operations. However, for this scenario we will be using one of the implemented operations.

10. Keep this browser window open as you will use the service links in later sections of this lab.

    Scroll down to confirm the implemented **Get** operation under **/employees/{employeeNumber}/Employee_CallableFlow**



/employees/{employeeNumber}/Employee_CallableFlow

GET getEmployee_CallableFlow Retrieve Employee using CallableFlow Implemented

Parameter	Type	Required?	Description
employeeNumber	Path	Yes	undefined

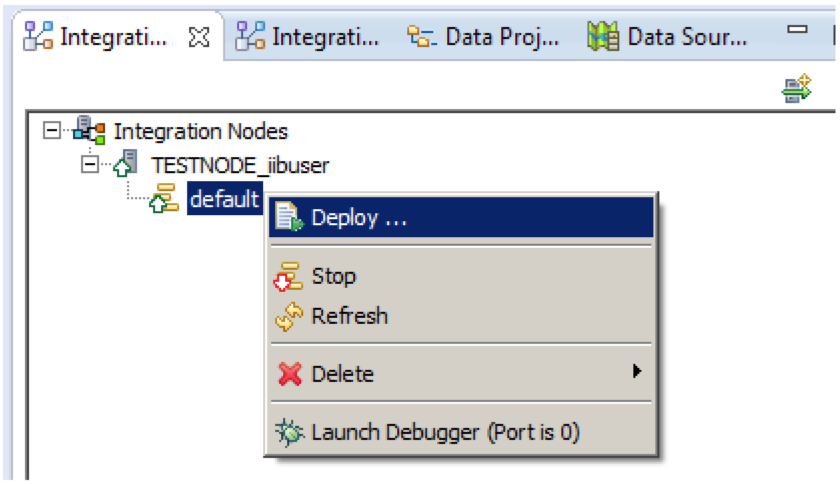
3. Deploy the Callable Application to the Integration Node on the Windows Environment

In this part of the lab you will deploy the Callable Application to your IIB node on Windows. The next few steps show this being done in the Integration Toolkit. However, you can also use the Web UI to perform the same task.

1. In the Integration Toolkit Application Development view, expand TESTNODE_iibuser. Delete resources if there are any deployed to the default integration server.

   Right-click the default server.

   Select deploy from the context menu.

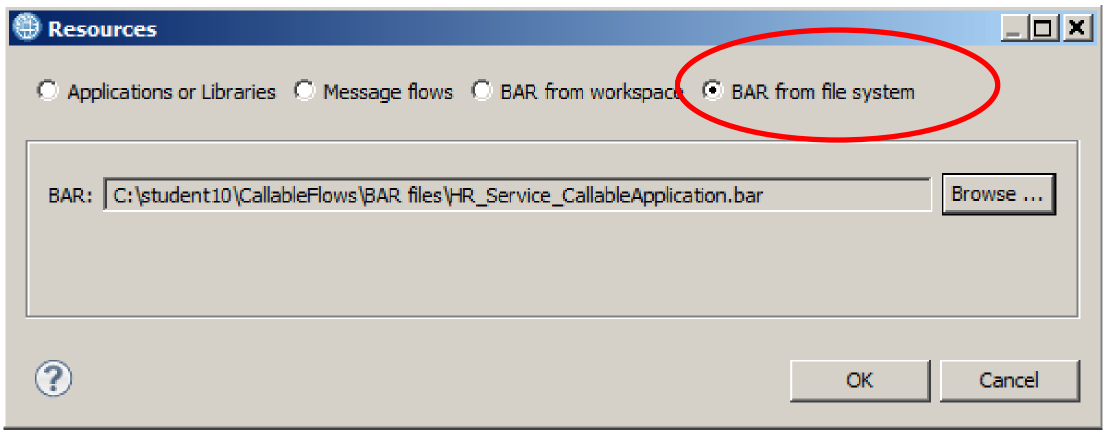


2. In the Resources window, select "BAR from file system".

   Use the Browse button to select the file

   `c:\student10\CallableFlows\BAR files\HR_Service_CallableApplication.bar`

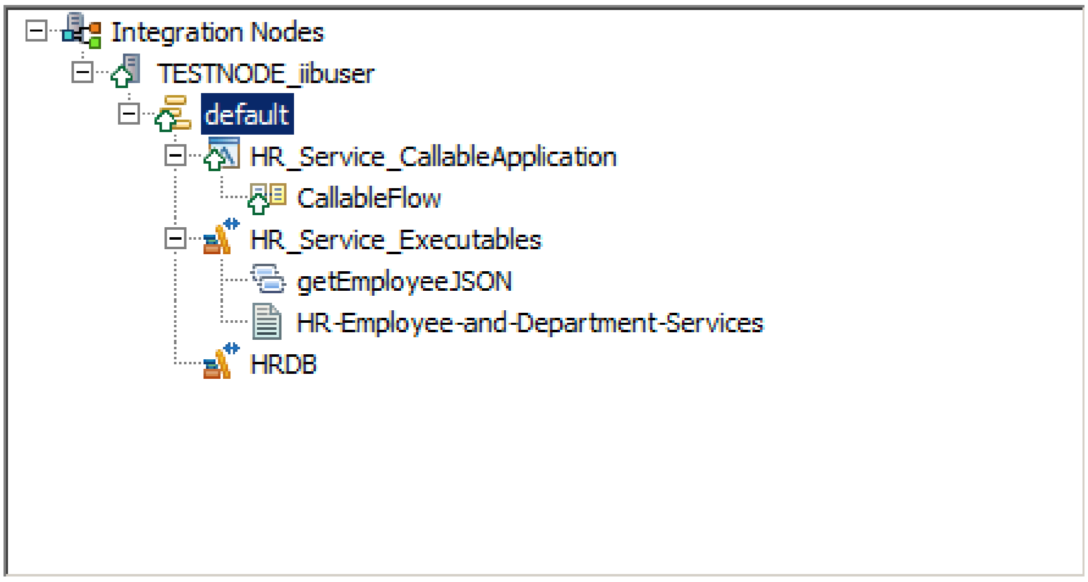
   Click OK.



Applications or Libraries [ ] Message flows [ ] BAR from workspace [ ] [✓ BAR from file system]

BAR: C:\student10\CallableFlows\BAR files\HR_Service_CallableApplication.bar

3. When deployed, the Application and its related artefacts can be seen in the Integration Nodes view.



```
Integration Nodes
  TESTNODE_iibuser
    default
      HR_Service_CallableApplication
        CallableFlow
      HR_Service_Executables
        getEmployeeJSON
        HR-Employee-and-Department-Services
      HRDB
```

At this point you have a REST API running in the IIB Docker container and a callable application running “on premises”.

The next task is to ensure that the Callable Flow Invoke function can call the Callable application. This is done by creating an IIB Switch on each of the IIB environments (both Windows and Docker Container). This enables the two environments to communicate through a secure connection. You will configure this in the next section.

4. Create and configure IIB Switch

In this part of the lab you will create an IIB Switch on your Windows IIB, which will enable a Secure Connection between the Integration Node running in Docker and the Integration Node running in Windows.

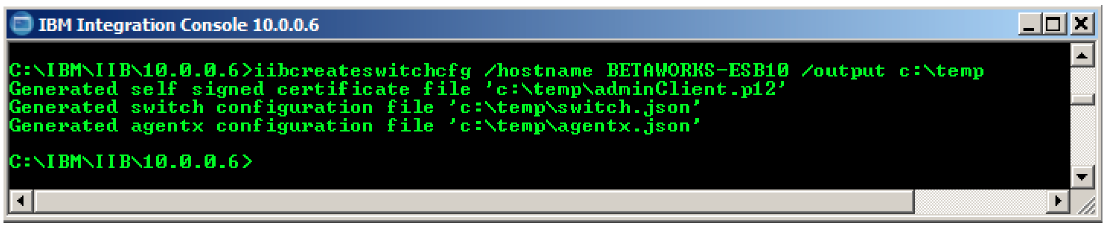
The Switch can be created on either of the IIB installations. In this lab you will create it on the IIB in the Windows environment.

Please note that the files generated will be stored in the folder **c:\temp**.

4.1 Create the IIB Switch on the Windows environment

1. Open an Integration Console and run the command:

```
iibcreateswitchcfg /hostname BETAWORKS-ESB10 /output c:\temp
```



```
C:\IBM\IIB\10.0.0.6>iibcreateswitchcfg /hostname BETAWORKS-ESB10 /output c:\temp
Generated self signed certificate file 'c:\temp\adminClient.p12'
Generated switch configuration file 'c:\temp\switch.json'
Generated agentx configuration file 'c:\temp\agentx.json'

C:\IBM\IIB\10.0.0.6>
```

If the command is successful, you will see in the response that three files have been generated.

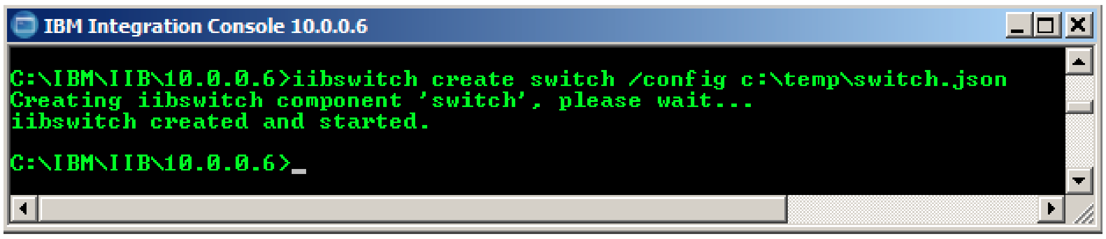
The command creates two JSON configuration files, and a certificate.

- (adminClient.p12) is a certificate used to store a private keys and certificate chain for the connection between the IIB Docker container and the 'On-premises' Integration Node.
- (switch.json) is used to create the Switch server.
- (agentx.json) is used by the **mqsichangeproperties** command to configure secure connectivity for the integration servers where your flows are deployed. The file is used to configure both IIB environments – IIB running on Windows and IIB running in the Docker Container.

Please note the flag **`/hostname`** in the command above ensures that the configuration files created contain the hostname for the host where the Switch is being created.

2. Run the **iibswitch** command to create (or update) the Switch server by using the configuration file (switch.json) that you created in the previous step.

```
iibswitch create switch /config c:\temp\switch.json
```



```
C:\IBM\IIB\10.0.0.6>iibswitch create switch /config c:\temp\switch.json
Creating iibswitch component 'switch', please wait...
iibswitch created and started.

C:\IBM\IIB\10.0.0.6>_
```

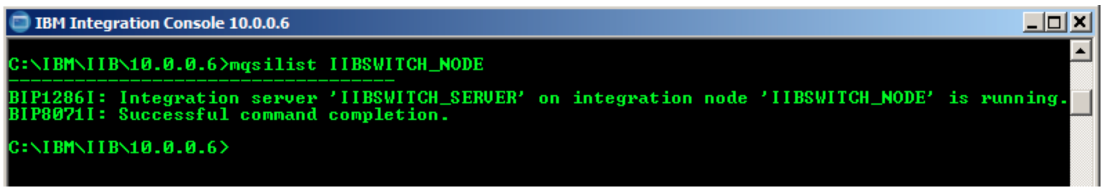
If you receive the response "**iibswitch already created, cannot create**", rerun the command using the update option:

```
iibswitch update switch /config c:\temp\switch.json
```

If the command is successful, you will see a message that the switch has been created and started, or has been updated.

3. To test that the Switch server is created and running, run the command

```
mqsilist IIBSWITCH_NODE
```



```
C:\IBM\IIB\10.0.0.6>mqsilist IIBSWITCH_NODE
-----------------------------------------------
BIP1286I: Integration server 'IIBSWITCH_SERVER' on integration node 'IIBSWITCH_NODE' is running.
BIP8071I: Successful command completion.

C:\IBM\IIB\10.0.0.6>
```

4. You will need to ensure that the integration server where you have deployed your Callable Application has the correct certificate to communicate securely with the Switch server.

   This requires you to run the **mqsichangeproperties** command for each Integration server where you have deployed callable message flows. The command will use the integration server configuration file (**agentx.json**) that you created in step 1.

   Run the command below **all on one** line (formatted here for readability):

```
mqsichangeproperties TESTNODE_iibuser
                              -e default
                              -o ComIbmIIBSwitchManager
                              -n agentXConfigFile
                              -p c:\temp\agentx.json
```

5. Restart the Integration Node on the Windows environment, to make the changes.

```
mqsistop TESTNODE_iibuser
```

   and

```
mqsistart TESTNODE_iibuser
```

5. Configure the IIB Connectivity agent in the Docker environment

In this part of the lab, you will configure your IIB node running in the Docker container to access the IIB in the Windows environment through a secure connection. You will first verify that the **agentx.json** file has been successfully copied to the IIB Container. Also you will see how you can run any **mqsi** command in a shell inside your container.

5.1 Copy configuration file agentx.json to IIB on Docker Container

Now that you have the IIB Switch up and running on the Windows environment, you will need to use the **agentx.json** file to configure the IIB node on Docker. When complete, IIB on Docker will be able to connect to IIB running on the windows environment through a secure connection.

1. To set up the IIB on Docker to connect to the IIB Switch on the IIB in the Windows environment, you need to run the **mqsichangeproperties** command using the generated **agentx.json** configuration file. This file must first be copied to the Docker image. Remember, this is a Linux-based image.

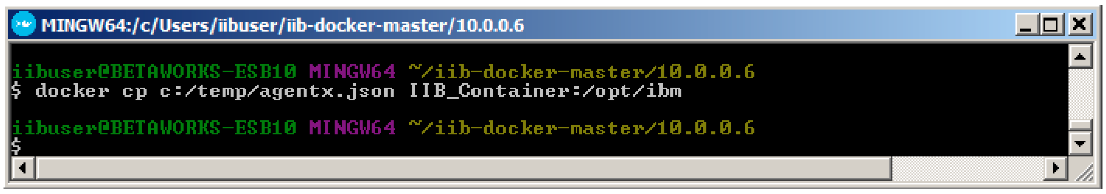
   Docker provides a command (**docker cp**) which will copy a file from a host to the Docker image (or vice versa).

   The configuration file **agentx.json** is in the **c:\temp** folder, which is a local folder on the VM. You will copy that file to a folder (let's say **/opt/ibm**) in the Docker container **IIB_Container.**

   In the Docker command prompt that you used earlier (step 2.1), run the command below.

   Note the forward slashes in all parts of this command.

   ```
   docker cp c:/temp/agentx.json IIB_Container:/opt/ibm
   ```



MINGW64:/c/Users/iibuser/iib-docker-master/10.0.0.6

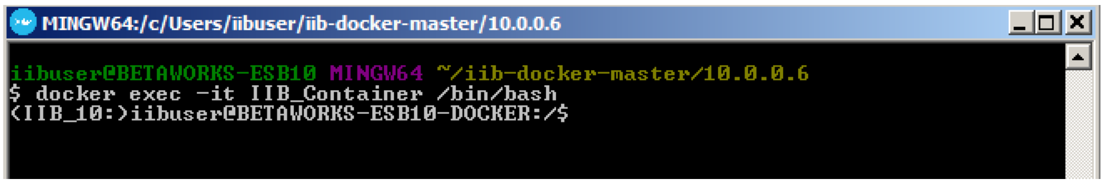
```
iibuser@BETAWORKS-ESB10 MINGW64 ~/iib-docker-master/10.0.0.6
$ docker cp c:/temp/agentx.json IIB_Container:/opt/ibm

iibuser@BETAWORKS-ESB10 MINGW64 ~/iib-docker-master/10.0.0.6
$
```

   You will verify the copy in the following steps.

5.2 Work in your IIB Container

1. You will now set up your Docker command prompt to run commands directly in your shell. In your Docker command prompt, attach a bash session to your container by running the command below.

   ```
   docker exec -it IIB_Container /bin/bash
   ```

   
   MINGW64:/c/Users/iibuser/iib-docker-master/10.0.0.6

   ```
   iibuser@BETAWORKS-ESB10 MINGW64 ~/iib-docker-master/10.0.0.6
   $ docker exec -it IIB_Container /bin/bash
   (IIB_10:)iibuser@BETAWORKS-ESB10-DOCKER:/$
   ```

   You are now in a shell in your IIB Container.
   You will also notice the hostname that you specified, when you were creating the container in step 2.1.7 – BETAWORKS-ESB10-DOCKER

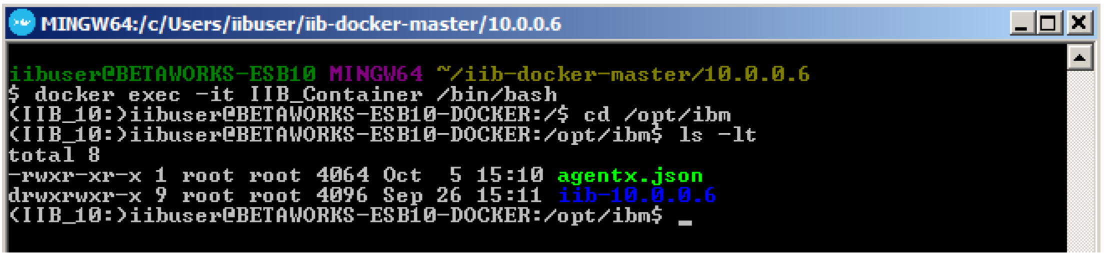
2. Verify that the earlier copy of the **agentx.json** file (step 4.2.1) was successful and is available in the container.

   If you followed the instructions the file should be in **/opt/ibm** folder. Go to that directory to view its content. In the command prompt, run the command below.

   ```
   cd /opt/ibm
   ```

3. Now that you are in the `/opt/ibm` directory, see its content by running the command

   ```
   ls -lt
   ```

   
   MINGW64:/c/Users/iibuser/iib-docker-master/10.0.0.6

   ```
   iibuser@BETAWORKS-ESB10 MINGW64 ~/iib-docker-master/10.0.0.6
   $ docker exec -it IIB_Container /bin/bash
   (IIB_10:)iibuser@BETAWORKS-ESB10-DOCKER:/$ cd /opt/ibm
   (IIB_10:)iibuser@BETAWORKS-ESB10-DOCKER:/opt/ibm$ ls -lt
   total 8
   -rwxr-xr-x 1 root root 4064 Oct  5 15:10 agentx.json
   drwxrwxr-x 9 root root 4096 Sep 26 15:11 iib-10.0.0.6
   (IIB_10:)iibuser@BETAWORKS-ESB10-DOCKER:/opt/ibm$ _
   ```

   You will see that the agentx.json file has been copied successfully and you should see the current date.

4. When you start an IIB runtime component on Linux and UNIX systems, it inherits the environment from where you issue the **mqsistart** command.

   You must therefore initialize the environment before you start a component; the command **mqsiprofile** performs this initialization

   The **`mqsiprofile`** command is located in the IIB installation folder **`/opt/ibm/iib-10.0.0.6/server/bin`**.

   Please note that if you are using a later version of IIB for your Docker container, **`10.0.0.6`** should be replaced with the corresponding version number. On this occasion you can see in step 5.2.3 showing the level of the current installation.

   In the Docker command prompt, run the command below (**note the dot** at the beginning).

   ```
   . /opt/ibm/iib-10.0.0.6/server/bin/mqsiprofile
   ```

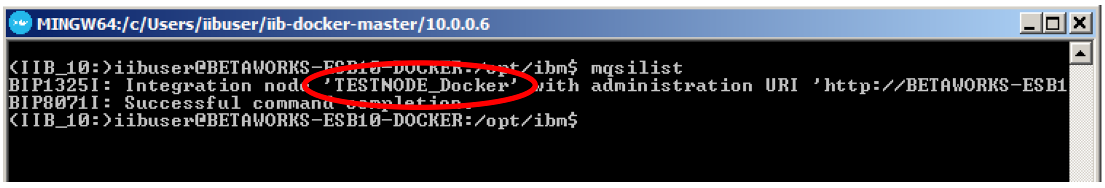
   On successful completion of the command you will see

   ```
   MQSI 10.0.0.6
   /opt/ibm/iib-10.0.0.6/server
   ```

5. You are now able to run **mqsi** commands directly into the Docker command prompt. Run the command :

   ```
   mqsilist
   ```

   You will see that the command return confirmation that your **TESTNODE_Docker** is up and running.


MINGW64:/c/Users/iibuser/iib-docker-master/10.0.0.6

```
(IIB_10:)iibuser@BETAWORKS-ESB10-DOCKER:/opt/ibm$ mqsilist
BIP1325I: Integration node 'TESTNODE_Docker' with administration URI 'http://BETAWORKS-ESB1
BIP8071I: Successful command completion.
(IIB_10:)iibuser@BETAWORKS-ESB10-DOCKER:/opt/ibm$
```

5.3 Configure IIB on Docker to connect to IIB Switch

1. In order to set up the IIB on Docker to connect to the IIB Switch on the IIB in your Windows environment, you need to run the **mqsichangeproperties** command using the generated **agentx.json** configuration file. Similar to step 4.1.4 you will run the command for the **TESTNODE_Docker** Integration Node. Run the command below **all on one line**.

```
mqsichangeproperties TESTNODE_Docker
                           -e default
                           -o ComIbmIIBSwitchManager
                           -n agentXConfigFile
                           -p /opt/ibm/agentx.json
```

   If the command completed successfully, you will see

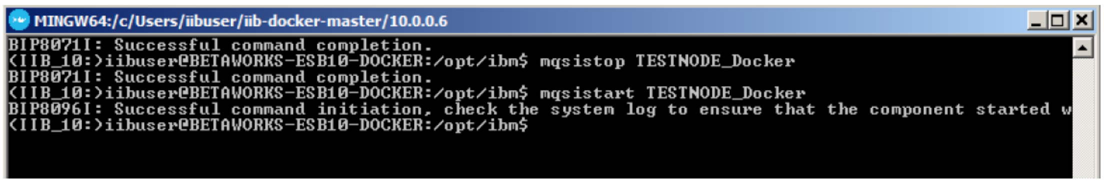
```
BIP8071I: Successful command completion.
```

2. Restart the Integration Node in the Docker container, which contains the integration server with the REST API.

```
mqsistop TESTNODE_Docker
```

   and

```
mqsistart TESTNODE_Docker
```



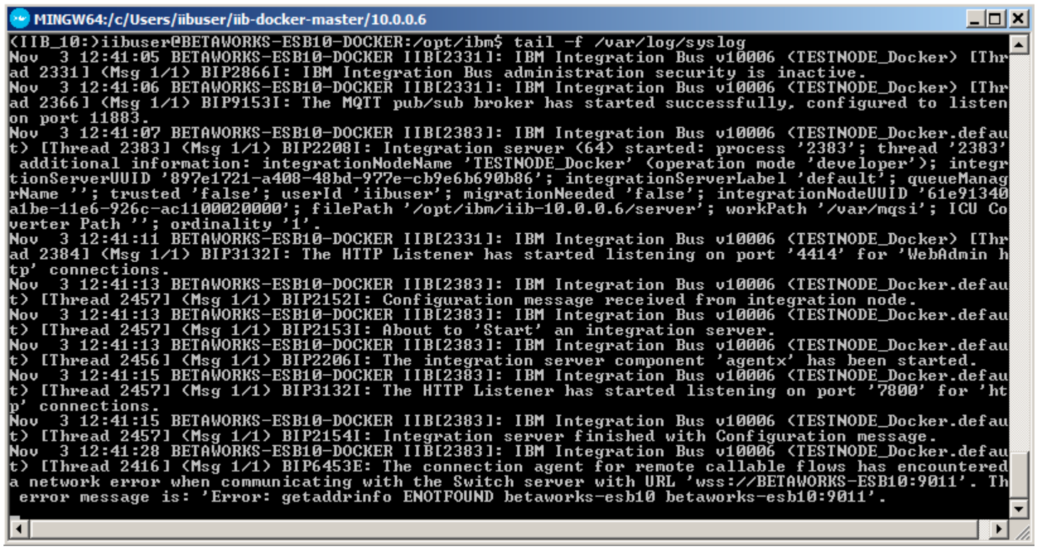
```
MINGW64:/c/Users/iibuser/iib-docker-master/10.0.0.6
BIP8071I: Successful command completion.
(IIB_10:)iibuser@BETAWORKS-ESB10-DOCKER:/opt/ibm$ mqsistop TESTNODE_Docker
BIP8071I: Successful command completion.
(IIB_10:)iibuser@BETAWORKS-ESB10-DOCKER:/opt/ibm$ mqsistart TESTNODE_Docker
BIP8096I: Successful command initiation, check the system log to ensure that the component started w
(IIB_10:)iibuser@BETAWORKS-ESB10-DOCKER:/opt/ibm$
```

3. As with any other IIB installation, you can access the IIB syslog messages in an IIB Container.

   Run the command

   tail –f /var/log/syslog

   This will show the IIB Syslog:

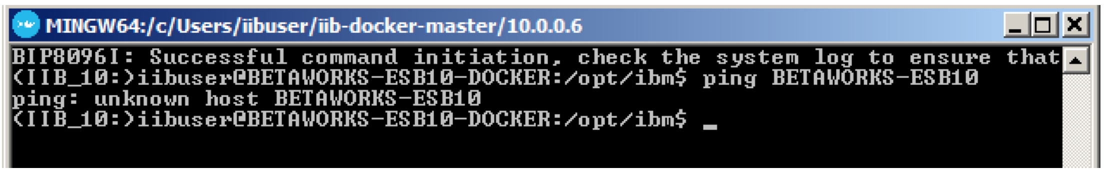


```
MINGW64:/c/Users/iibuser/iib-docker-master/10.0.0.6
(IIB_10:)iibuser@BETAWORKS-ESB10-DOCKER:/opt/ibm$ tail -f /var/log/syslog
Nov  3 12:41:05 BETAWORKS-ESB10-DOCKER IIB[2331]: IBM Integration Bus v10006 (TESTNODE_Docker) [Thr
ad 2331] (Msg 1/1) BIP2866I: IBM Integration Bus administration security is inactive.
Nov  3 12:41:06 BETAWORKS-ESB10-DOCKER IIB[2331]: IBM Integration Bus v10006 (TESTNODE_Docker) [Thr
ad 2366] (Msg 1/1) BIP9153I: The MQTT pub/sub broker has started successfully, configured to listen
on port 11883.
Nov  3 12:41:07 BETAWORKS-ESB10-DOCKER IIB[2383]: IBM Integration Bus v10006 (TESTNODE_Docker.defau
t) [Thread 2383] (Msg 1/1) BIP2208I: Integration server (64) started: process '2383'; thread '2383'
 additional information: integrationNodeName 'TESTNODE_Docker' (operation mode 'developer'); integr
tionServerUUID '897e1721-a408-48bd-977e-cb9e6b690b86'; integrationServerLabel 'default'; queueManag
rName ''; trusted 'false'; userId 'iibuser'; migrationNeeded 'false'; integrationNodeUUID '61e91340
a1be-11e6-926c-ac1100020000'; filePath '/opt/ibm/iib-10.0.0.6/server'; workPath '/var/mqsi'; ICU Co
verter Path ''; ordinality '1'.
Nov  3 12:41:11 BETAWORKS-ESB10-DOCKER IIB[2331]: IBM Integration Bus v10006 (TESTNODE_Docker) [Thr
ad 2384] (Msg 1/1) BIP3132I: The HTTP Listener has started listening on port '4414' for 'WebAdmin h
tp' connections.
Nov  3 12:41:13 BETAWORKS-ESB10-DOCKER IIB[2383]: IBM Integration Bus v10006 (TESTNODE_Docker.defau
t) [Thread 2457] (Msg 1/1) BIP2152I: Configuration message received from integration node.
Nov  3 12:41:13 BETAWORKS-ESB10-DOCKER IIB[2383]: IBM Integration Bus v10006 (TESTNODE_Docker.defau
t) [Thread 2457] (Msg 1/1) BIP2153I: About to 'Start' an integration server.
Nov  3 12:41:13 BETAWORKS-ESB10-DOCKER IIB[2383]: IBM Integration Bus v10006 (TESTNODE_Docker.defau
t) [Thread 2456] (Msg 1/1) BIP2206I: The integration server component 'agentx' has been started.
Nov  3 12:41:15 BETAWORKS-ESB10-DOCKER IIB[2383]: IBM Integration Bus v10006 (TESTNODE_Docker.defau
t) [Thread 2457] (Msg 1/1) BIP3132I: The HTTP Listener has started listening on port '7800' for 'ht
p' connections.
Nov  3 12:41:15 BETAWORKS-ESB10-DOCKER IIB[2383]: IBM Integration Bus v10006 (TESTNODE_Docker.defau
t) [Thread 2457] (Msg 1/1) BIP2154I: Integration server finished with Configuration message.
Nov  3 12:41:28 BETAWORKS-ESB10-DOCKER IIB[2383]: IBM Integration Bus v10006 (TESTNODE_Docker.defau
t) [Thread 2416] (Msg 1/1) BIP6453E: The connection agent for remote callable flows has encountered
a network error when communicating with the Switch server with URL 'wss://BETAWORKS-ESB10:9011'. Th
 error message is: 'Error: getaddrinfo ENOTFOUND betaworks-esb10 betaworks-esb10:9011'.
```

5.4 Confirm connectivity from Docker container to Windows environment

1. Exit the 'tail' command and type

`ping BETAWORKS-ESB10`



MINGW64:/c/Users/iibuser/iib-docker-master/10.0.0.6

```
BIP8096I: Successful command initiation, check the system log to ensure that
<IIB_10:>iibuser@BETAWORKS-ESB10-DOCKER:/opt/ibm$ ping BETAWORKS-ESB10
ping: unknown host BETAWORKS-ESB10
<IIB_10:>iibuser@BETAWORKS-ESB10-DOCKER:/opt/ibm$ _
```

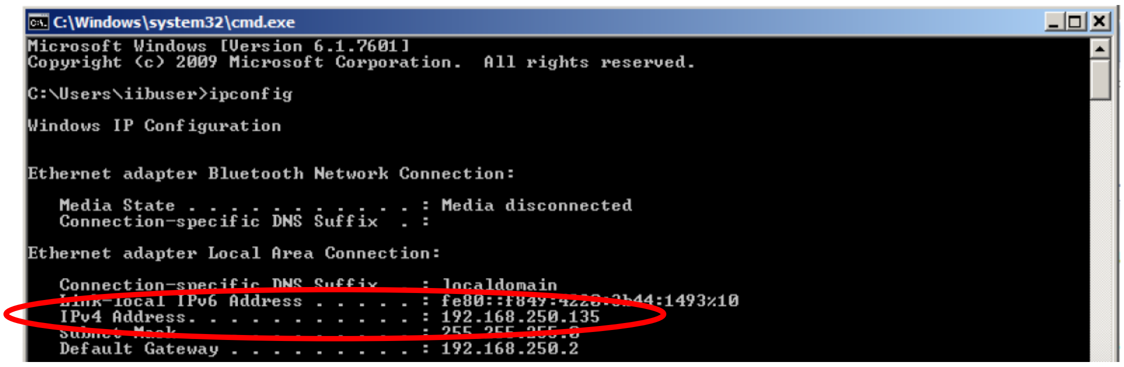
It is possible that the VMWare installation is able to resolve the hostname ping. If you see a successful ping response, skip to Section 6 - Test Docker Integration on page 28.

However, if the ping command fails, you will need to add a new entry to the local hosts file. Follow the instructions in the next three steps.

For those, who may have looked at the syslog closely will notice that in the bottom of the syslog, there is a message suggesting a network error. This is due to the fact that the host name is not recognized. This will be resolved in the next few steps.

2. You will need the IP address of the Windows environment (BETAWORKS-ESB10).

Obtain the IP address for the Windows environment. You can see the IP address by running **ipconfig** in Windows Command Prompt:



C:\Windows\system32\cmd.exe

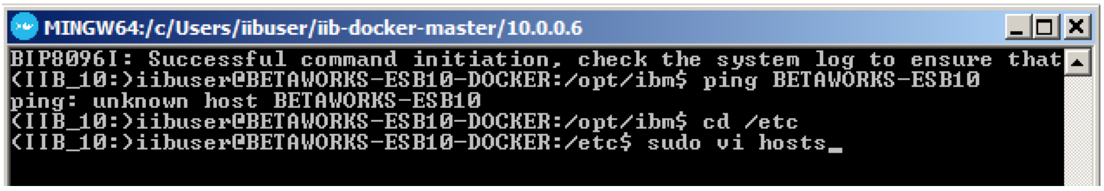
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\iibuser>ipconfig

Windows IP Configuration


Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : localdomain
   Link-local IPv6 Address . . . . . : fe80::f849:4220:3b44:1493%10
   IPv4 Address. . . . . . . . . . . : 192.168.250.135
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.250.2
```

3. Back in the Docker VM bash shell:

`cd /etc`

then

`sudo vi hosts`



MINGW64:/c/Users/iibuser/iib-docker-master/10.0.0.6

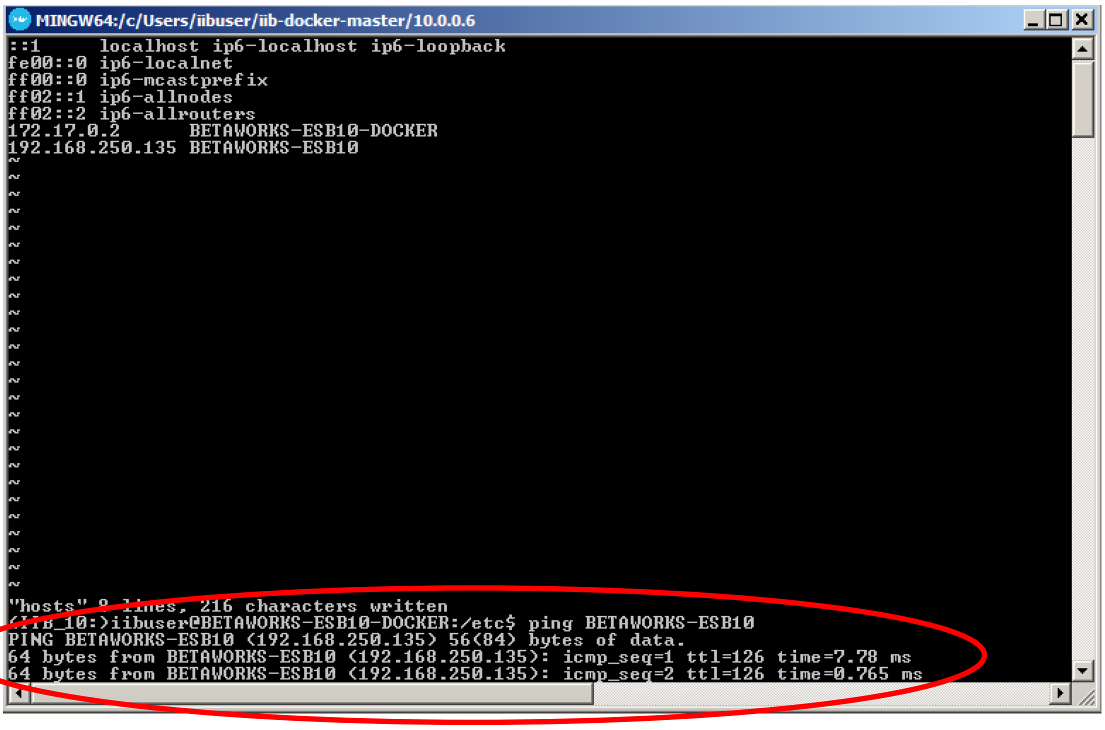
```
BIP8096I: Successful command initiation, check the system log to ensure that
<IIB_10:>iibuser@BETAWORKS-ESB10-DOCKER:/opt/ibm$ ping BETAWORKS-ESB10
ping: unknown host BETAWORKS-ESB10
<IIB_10:>iibuser@BETAWORKS-ESB10-DOCKER:/opt/ibm$ cd /etc
<IIB_10:>iibuser@BETAWORKS-ESB10-DOCKER:/etc$ sudo vi hosts_
```

4. You now have opened the hosts file as administrator. Follow the next steps **very carefully**:

   - Hit the 'j' on your keyboard multiple times until the cursor reaches the last line (`172.xxx.xxx.xx`);
   - Hit 'o' – this will insert an empty line below;
   - Type `192.168.xxx.xxx BETAWORKS-ESB10`
     where **xxx** is the last part of the exact IP address of the main Windows VM
   - Hit '**Esc**' on your keyboard
   - Type `:wq!` and then hit **Enter**

   When complete your file should look similar to the screen capture below.

   If changes have been made correctly, you should now have a successful ping.



```
::1      localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.2       BETAWORKS-ESB10-DOCKER
192.168.250.135 BETAWORKS-ESB10
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"hosts" 8 lines, 216 characters written
(IIB_10:)iibuser@BETAWORKS-ESB10-DOCKER:/etc$ ping BETAWORKS-ESB10
PING BETAWORKS-ESB10 (192.168.250.135) 56(84) bytes of data.
64 bytes from BETAWORKS-ESB10 (192.168.250.135): icmp_seq=1 ttl=126 time=7.78 ms
64 bytes from BETAWORKS-ESB10 (192.168.250.135): icmp_seq=2 ttl=126 time=0.765 ms
```

6. Test your Docker Integration Scenario

In this part of the lab you will test your integration scenario. All the required configuration is now complete.

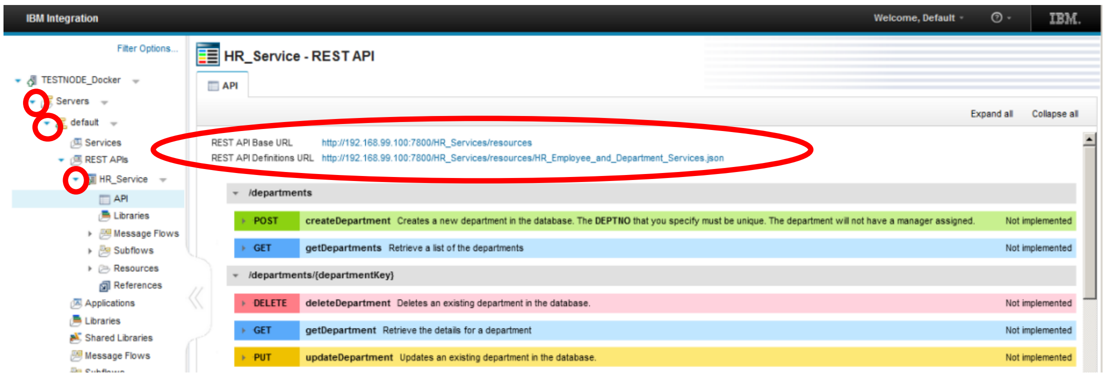
You will invoke the REST API HR_Service, running in a Docker container. The REST API contains a 'Callable Flow Invoke' which will connect to IIB on Windows Callable application HR_Service_CallableApplication. The Callable application (in the Windows environment) will retrieve Employee data from the database and returns it to the Calling application in the IIB Docker container.

6.1 View REST API Service URLs

1. Bring up your browser window (you should have this open) with the TESTNODE_Docker Web UI at the URL

```
192.168.99.100:4414
```

As in step 2.2, expand **default -> REST API -> HR_Service** and click on **API**.



REST API Base URL http://192.168.99.100:7800/HR_Services/resources
REST API Definitions URL http://192.168.99.100:7800/HR_Services/resources/HR_Employee_and_Department_Services.json

/departments

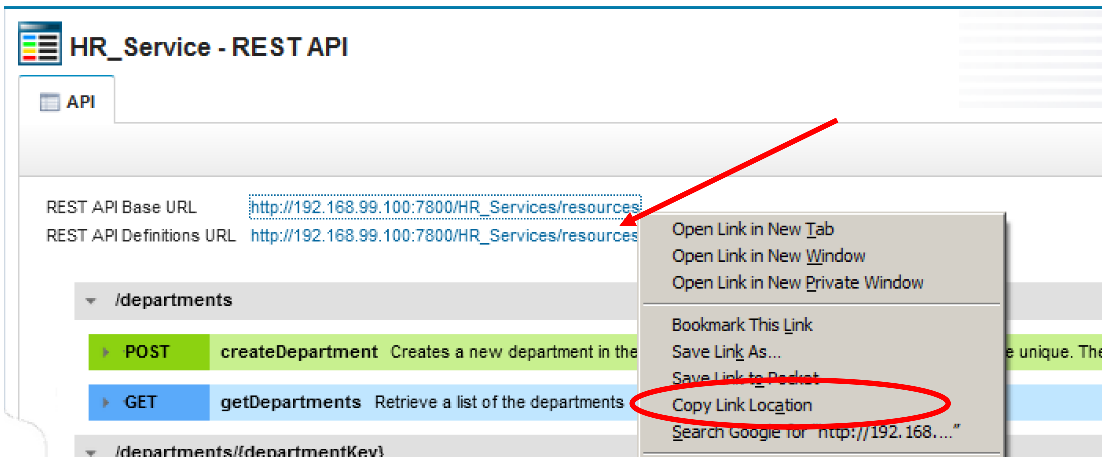
POST	**createDepartment** Creates a new department in the database. The **DEPTNO** that you specify must be unique. The department will not have a manager assigned.	Not implemented
GET	**getDepartments** Retrieve a list of the departments	Not implemented

/departments/{departmentKey}

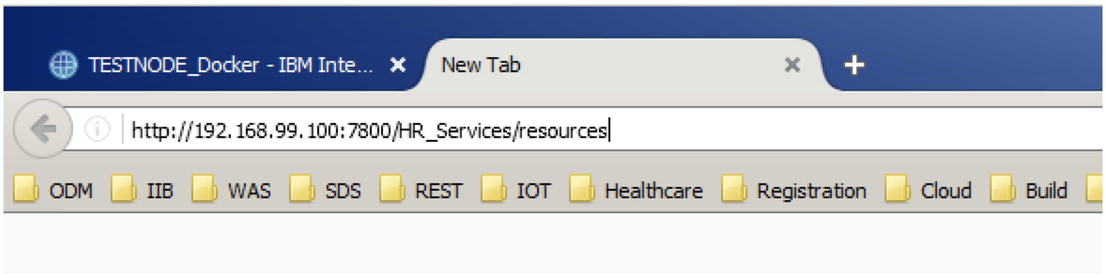
DELETE	**deleteDepartment** Deletes an existing department in the database.	Not implemented
GET	**getDepartment** Retrieve the details for a department	Not implemented
PUT	**updateDepartment** Updates an existing department in the database.	Not implemented

6.2 Test the REST API in a Browser

1. Right-click and **Copy Link Location** of the REST API Base URL

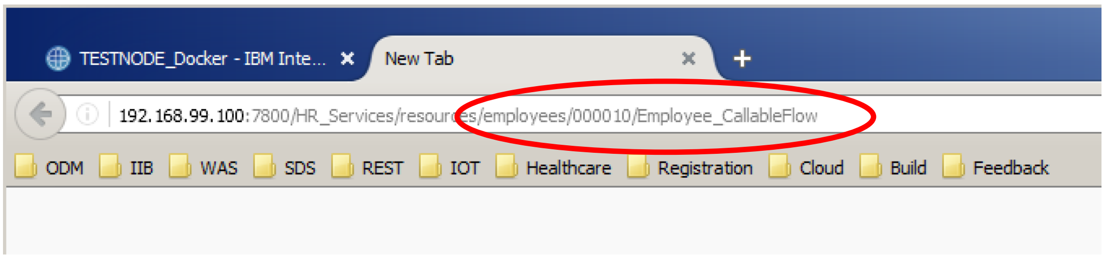

HR_Service - REST API
API
REST API Base URL http://192.168.99.100:7800/HR_Services/resources
REST API Definitions URL http://192.168.99.100:7800/HR_Services/resources
/departments
POST createDepartment Creates a new department in the
GET getDepartments Retrieve a list of the departments
Open Link in New Tab
Open Link in New Window
Open Link in New Private Window
Bookmark This Link
Save Link As...
Save Link to Pocket
Copy Link Location
Search Google for "http://192.168...."

2. Open a new browser window and paste the URL.


TESTNODE_Docker - IBM Inte...
New Tab
http://192.168.99.100:7800/HR_Services/resources
ODM
IIB
WAS
SDS
REST
IOT
Healthcare
Registration
Cloud
Build

3. To the end of the URL add **`/employees/000010/Employee_CallableFlow`**

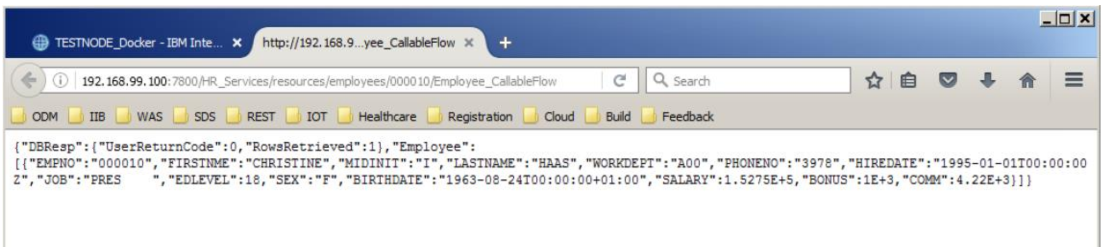
`000010` is the Employee number that you are requesting the record of.


TESTNODE_Docker - IBM Inte...
New Tab
192.168.99.100:7800/HR_Services/resources/employees/000010/Employee_CallableFlow
ODM
IIB
WAS
SDS
REST
IOT
Healthcare
Registration
Cloud
Build
Feedback

Hit **Enter**.

4. You should see the retrieved data…

   In summary, you have invoked the HR_Service REST API, which runs in the IIB Docker container. The implemented operation in the HR_Service API has called the callable flow, which runs in your Windows environment to retrieve the data from DB2 on Windows and returns the response.


TESTNODE_Docker - IBM Inte...
http://192.168.9...yee_CallableFlow
192.168.99.100:7800/HR_Services/resources/employees/000010/Employee_CallableFlow
Search
ODM
IIB
WAS
SDS
REST
IOT
Healthcare
Registration
Cloud
Build
Feedback

```
{"DBResp":{"UserReturnCode":0,"RowsRetrieved":1},"Employee":
[{"EMPNO":"000010","FIRSTNME":"CHRISTINE","MIDINIT":"I","LASTNAME":"HAAS","WORKDEPT":"A00","PHONENO":"3978","HIREDATE":"1995-01-01T00:00:00
Z","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}]}
```

You can try with other Employee records such as **000020** or **000050.**

END OF LAB GUIDE