IBM Integration Bus

Java Script Client API

Featuring:

- Generate Java Script Client API
- View and prepare Java Script API resources
- Test using Node.js
- Test in a Browser using Web application

June 2015

Hands-on lab built at product
Version 10.0.0.0

1. Introduction

This lab guide shows you how to generate the JavaScript API for an existing Integration Service and how to implement it in an application.

For this lab you will need an installation of Node.js and the Node Package Module (npm) for Dojo.

If you are using the pre-built image for the IBM Integration Bus V10 Workshop, this has already been completed for you and you may proceed with the lab.

In the Appendix at the end of this lab you will find instructions on how to install Node.js.

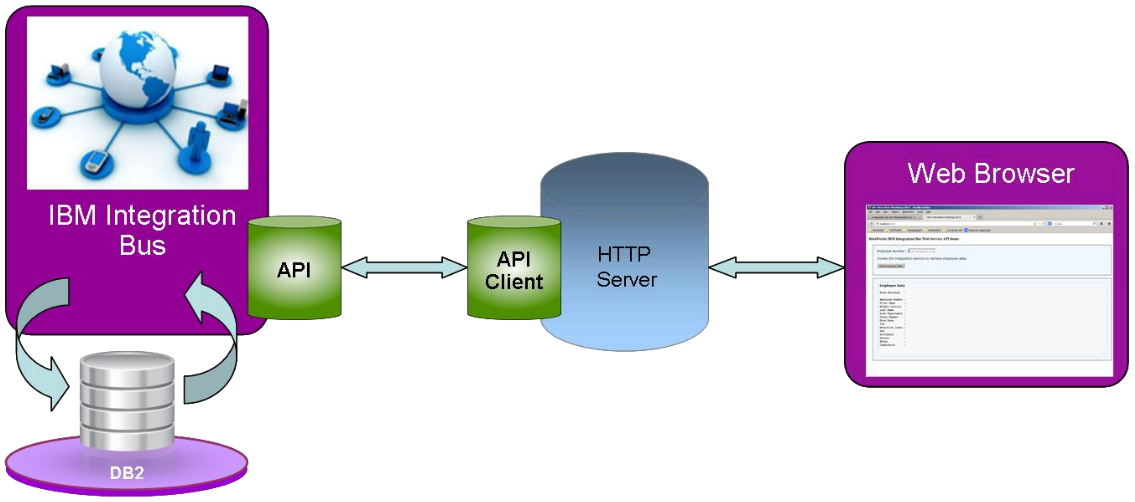
1.1 Scenario overview

For this lab you will be using an integration service which has been described in a previous lab, the Employee Service.

In this lab you will generate the JavaScript API and will view all the generated resources in a Web Browser.

You will then test the generated JavaScript API in two scenarios:

- Using only the Node.js environment and command console;
- Using a Web Browser JavaScript application. Please note that the JavaScript application has already been written for you and is to demonstrate an example of the JavaScript API implementation. The JavaScript application is not a part of the IIB product feature.



1.2 Outline of tasks

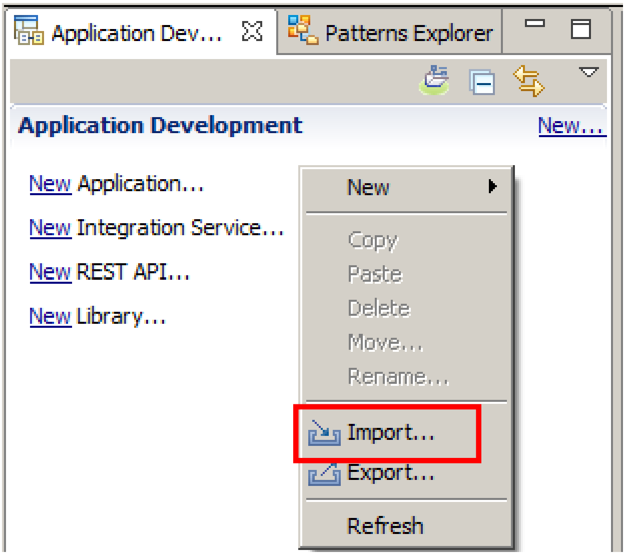
- Import Project Interchange (PI) file with a prebuilt solution of EmployeeService;
- Create the JavaScript Client bindings;
- Review and deploy the Integration Service to an Integration Node;
- In a Web Browser, view the generated JavaScript API code and create a file to contain this code;
- In a Node.js command window, test the JavaScript Client API;
- View and test a Web Browser JavaScript Application, which uses the generated JavaScript API;

2. Import the EmployeeService application

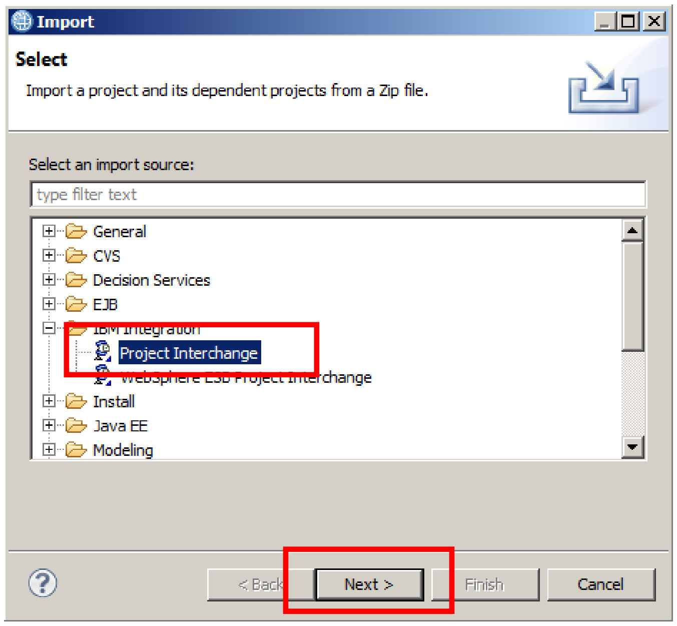
1. To avoid naming clashes with earlier labs, this lab will be developed using a new workspace.

   If you already have a workspace open, click File, Switch Workspace. Give the new workspace the name "JavaScriptAPI", or similar.

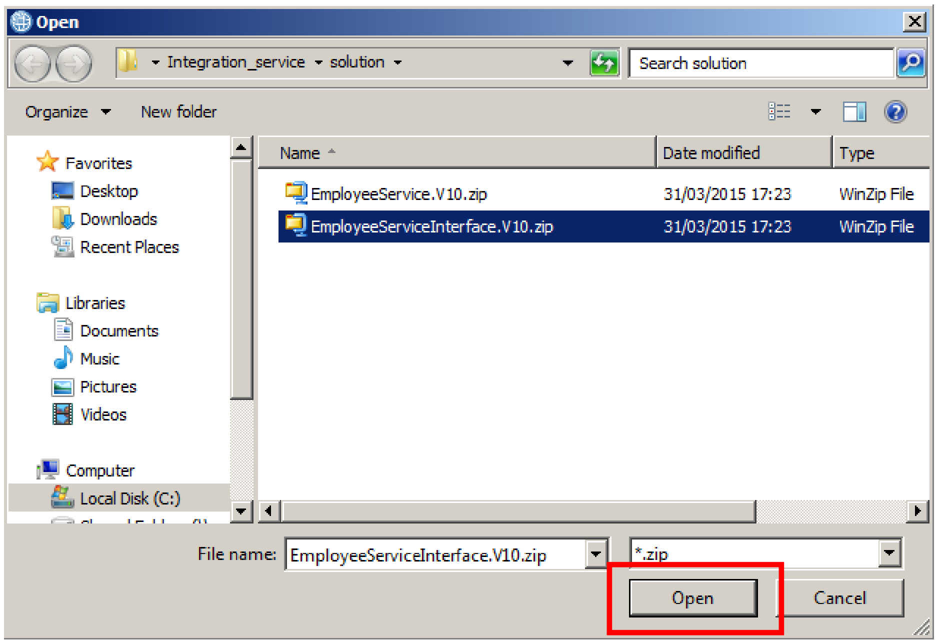
2. In the Integration Toolkit 'Application Development' view, right-click and select 'Import':



3. Select 'Project Interchange' on the bottom of the 'Import' Dialog, then Next, and then Browse.



4. Navigate to **`C:\student10\Integration_service\solution\`** and select the file EmployeeServiceInterfaceV10.zip

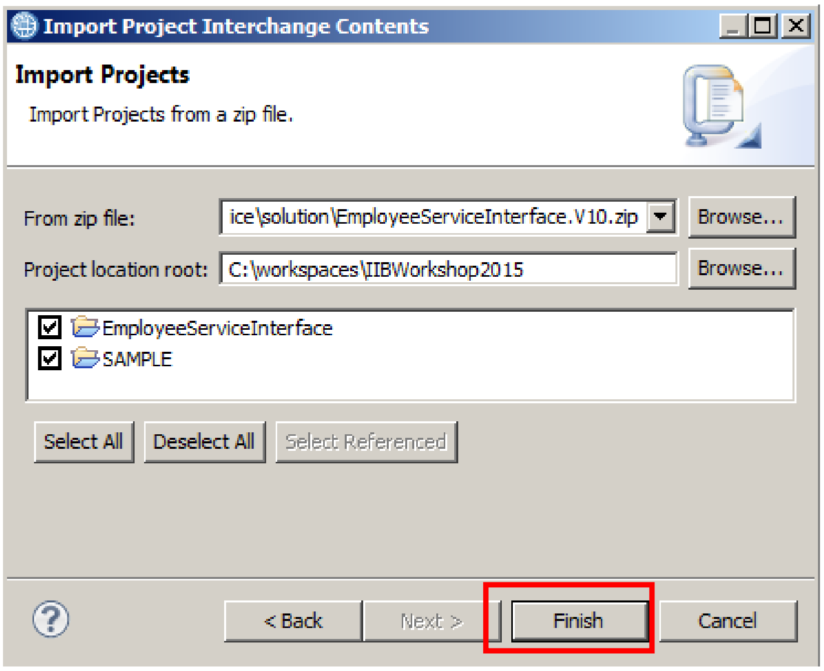


Name	Date modified	Type
EmployeeService.V10.zip	31/03/2015 17:23	WinZip File
EmployeeServiceInterface.V10.zip	31/03/2015 17:23	WinZip File

File name: EmployeeServiceInterface.V10.zip

*.zip

5. Click Finish to import the projects:



Import Projects

Import Projects from a zip file.

From zip file: ice\solution\EmployeeServiceInterface.V10.zip

Project location root: C:\workspaces\IIBWorkshop2015

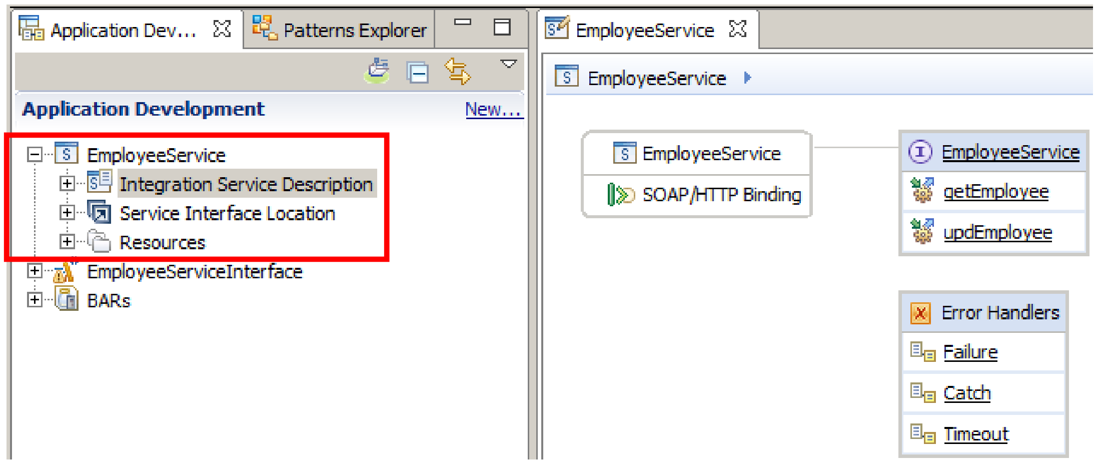
- [x] EmployeeServiceInterface
- [x] SAMPLE

6. Repeat steps 2.1 to 2.3 only this time import the PI file EmployeeService.V10.zip.

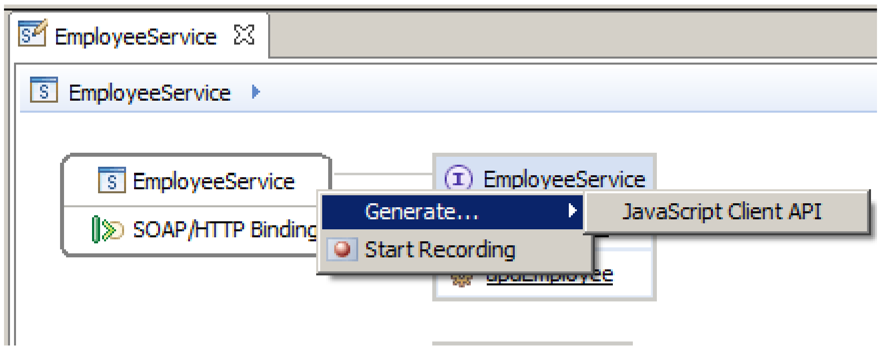
3. Generate the JavaScript Client API

3.1 Generate the JavaScript API

1. In the Application Development view, expand 'EmployeeService' in the navigator and double click 'Integration Service Description'.



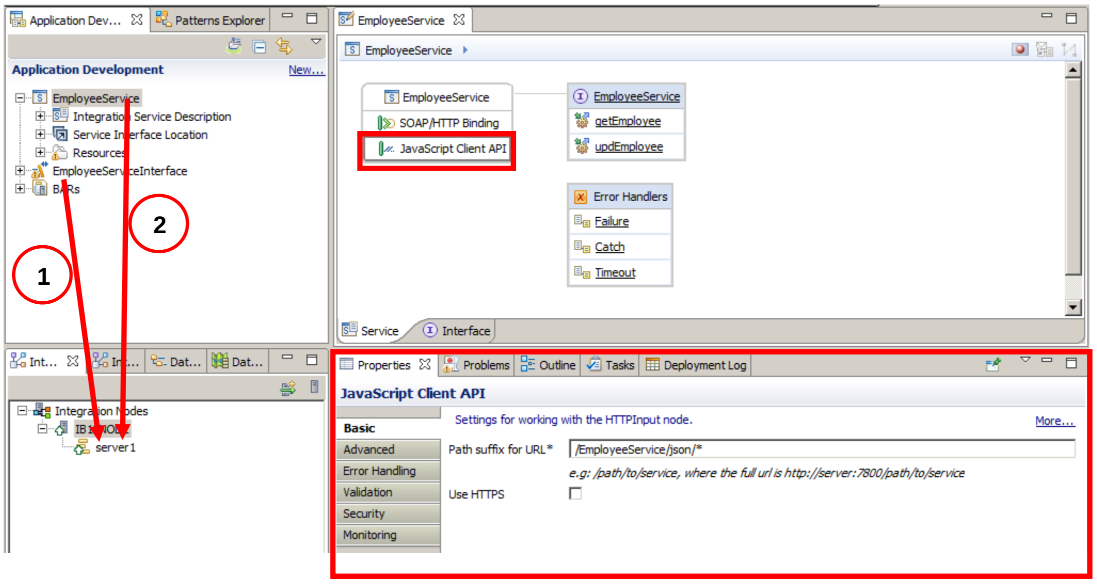
2. Right-click EmployeeService and then Generate -> JavaScript Client API.



3. After a few seconds the JavaScript binding is generated. Click the binding to view its properties in the Properties pane.

   Once finished reviewing, deploy first the shared library EmployeeServiceInterface and then the EmployeeService to **server1** on IB10NODE integration node.

   Start IB10NODE if not already started.



JavaScript Client API

Settings for working with the HTTPInput node.

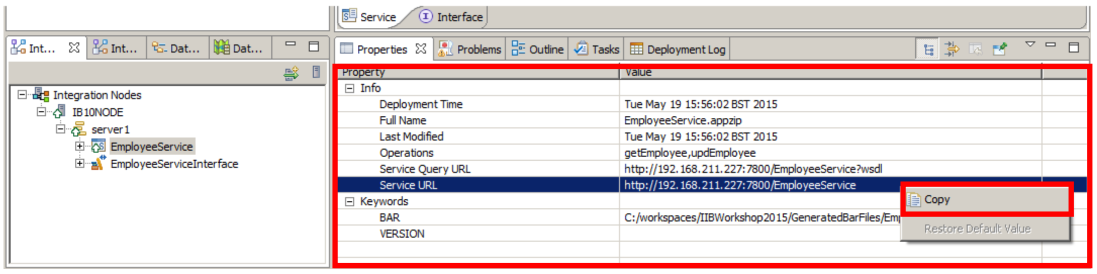
Path suffix for URL* /EmployeeService/json/*

e.g: /path/to/service, where the full url is http://server:7800/path/to/service

Use HTTPS ☐

4. In **server1**, click EmployeeService and you will see some of its properties.

   Right-Click the Service URL and copy the link.



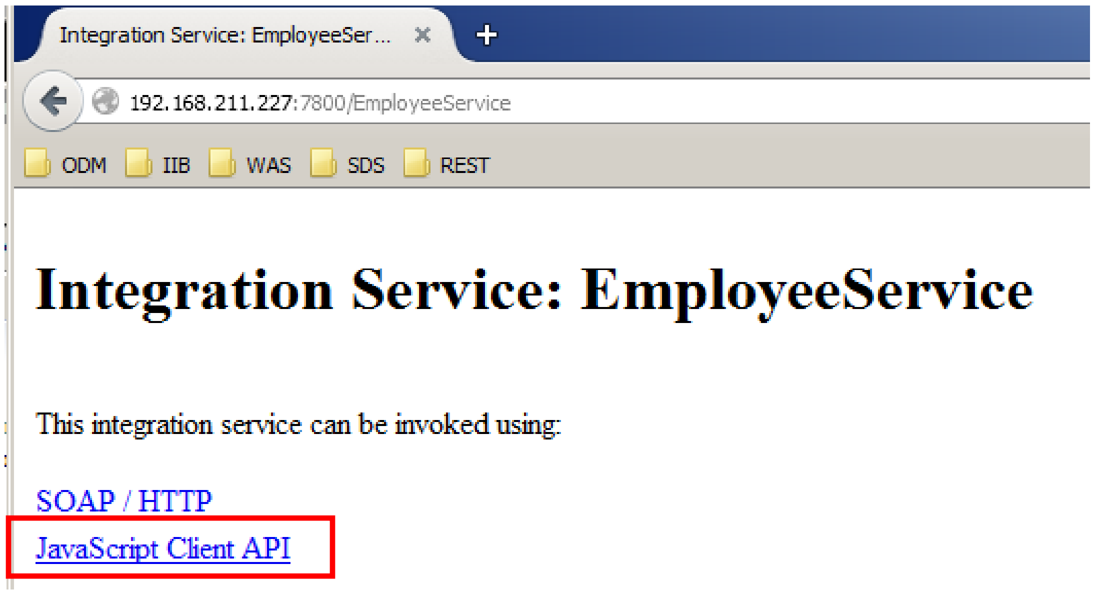
Property	Value
Info	
Deployment Time	Tue May 19 15:56:02 BST 2015
Full Name	EmployeeService.appzip
Last Modified	Tue May 19 15:56:02 BST 2015
Operations	getEmployee,updEmployee
Service Query URL	http://192.168.211.227:7800/EmployeeService?wsdl
Service URL	http://192.168.211.227:7800/EmployeeService
Keywords	
BAR	C:/workspaces/IIBWorkshop2015/GeneratedBarFiles/Em
VERSION	

3.2 View the generated resources in a Web Browser window.

1. Paste the 'Service URL' in the address field on a Web Browser and open the URL:

   Note that you will have to manually correct the format of the URL (there may be additional text before the IP address).

   Click the JavaScript Client API link.


Integration Service: EmployeeSer...
192.168.211.227:7800/EmployeeService
ODM
IIB
WAS
SDS
REST

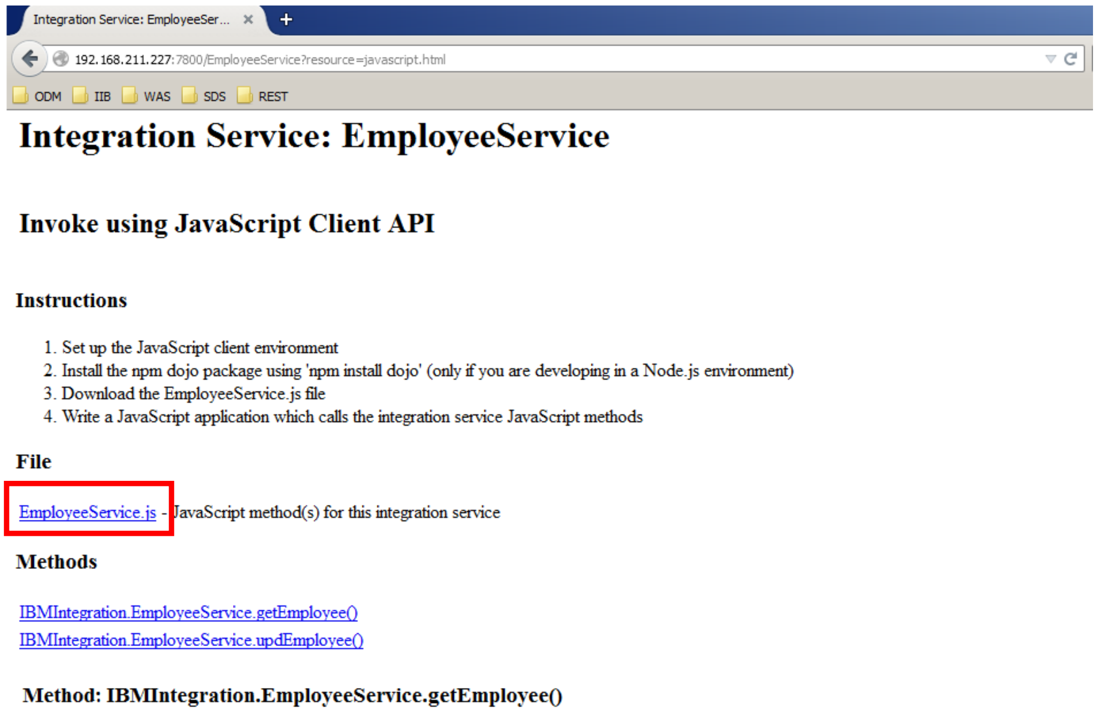
Integration Service: EmployeeService

This integration service can be invoked using:

SOAP / HTTP
JavaScript Client API

2. View the information provided. This was generated when you created the JavaScript API in the Integration Toolkit.

   When ready, right-click the EmployeeService.js and open it in a new Browser tab.



Integration Service: EmployeeSer...

192.168.211.227:7800/EmployeeService?resource=javascript.html

ODM IIB WAS SDS REST

Integration Service: EmployeeService

Invoke using JavaScript Client API

Instructions

1. Set up the JavaScript client environment
2. Install the npm dojo package using 'npm install dojo' (only if you are developing in a Node.js environment)
3. Download the EmployeeService.js file
4. Write a JavaScript application which calls the integration service JavaScript methods

File

EmployeeService.js - JavaScript method(s) for this integration service

Methods

IBMIntegration.EmployeeService.getEmployee()
IBMIntegration.EmployeeService.updEmployee()

Method: IBMIntegration.EmployeeService.getEmployee()

3. In Windows Explorer, create a file in the **c:/Program Files/nodejs** folder called **EmployeeService.js** and copy all the contents of the generated JavaScript file from the Web Browser tab into a new file.

   Please explore the file as you wish. No changes are needed.

   Note 1:

   As a good practice you will copy your '.js' API file in a separate (development) directory and then copy the file to your 'production' directory or environment.

   Note 2:

   About half way down the file you can see the host connection details – host, port number, protocol.

   If you are testing the integration service from another machine than the one on which is the IBM Integration Bus with the deployed service, please note that you **do not** need to make any changes here. The application file will override these values (port number, hostname).

```
 var ComplexError = function(errName, values) {
   this.errName = errName;
   var keys = Object.keys(values);
   for (var i=0; i<keys.length; i++) {
     this[keys[i]] = values[keys[i]];
   }
}

IBMIntegration.EmployeeService = {};
IBMIntegration.EmployeeService.IBMContext = {};
// default values that can be overridden in the client code
IBMIntegration.EmployeeService.IBMContext.hostname = "localhost";
IBMIntegration.EmployeeService.IBMContext.port     = 7800;
IBMIntegration.EmployeeService.IBMContext.protocol = "http";
IBMIntegration.EmployeeService.IBMContext.rejectUnauthorized =
true;

IBMIntegration.EmployeeService.getEmployee = {};
```

   Save the file with Ctrl-S and close it.

3.3 Prepare the JavaScript Client file

Back on the first tab in the Web Browser scroll down and copy the **‘Coding Example’** for the **‘getEmployee’** operation **only** in the grey box to another new file named **c:\Program Files\nodejs\EmpService_Test.js** (Open Notepad and paste the content. Please make sure that the extension is **.js**).

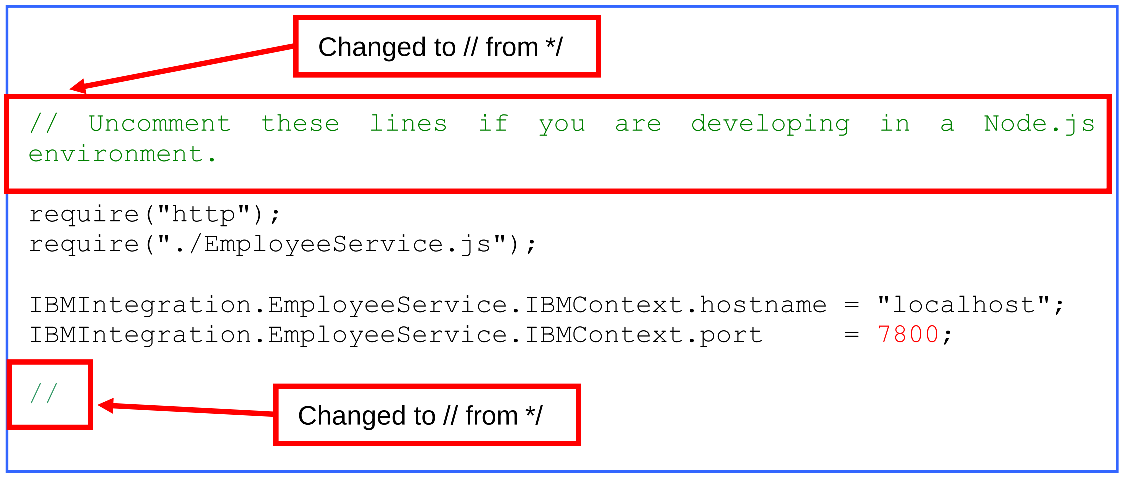
1. Change the command line near the top of the file:

```
  /* Uncomment these lines if you are developing in a Node.js
environment.
```

Replace ‘**/***’ with ‘**//**’.

Also replace ‘***/**’ after the port setting to ‘**//**’.

Once ready, this part of your file should look like this:



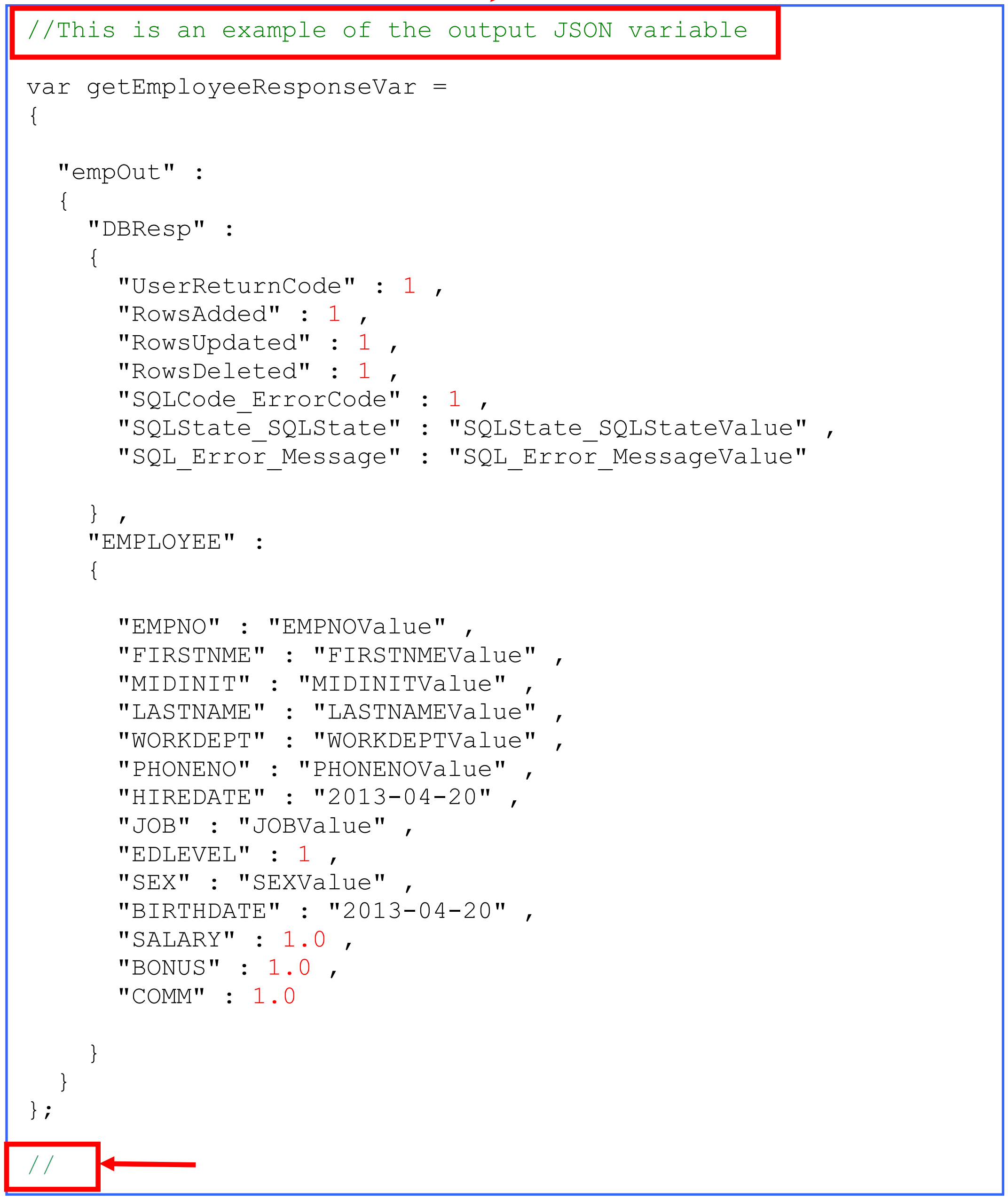
```
// Uncomment these lines if you are developing in a Node.js
environment.
require("http");
require("./EmployeeService.js");

IBMIntegration.EmployeeService.IBMContext.hostname = "localhost";
IBMIntegration.EmployeeService.IBMContext.port     = 7800;

//
```

2. Uncomment the **`getEmployeeResponse output`** variable:

```
/* This is an example of the output JSON variable.
```

Do this by changing the “/*” to “//”.

Do the same after the variable as shown below:



```
//This is an example of the output JSON variable

var getEmployeeResponseVar =
{

  "empOut" :
  {
    "DBResp" :
    {
      "UserReturnCode" : 1 ,
      "RowsAdded" : 1 ,
      "RowsUpdated" : 1 ,
      "RowsDeleted" : 1 ,
      "SQLCode_ErrorCode" : 1 ,
      "SQLState_SQLState" : "SQLState_SQLStateValue" ,
      "SQL_Error_Message" : "SQL_Error_MessageValue"

    } ,
    "EMPLOYEE" :
    {

      "EMPNO" : "EMPNOValue" ,
      "FIRSTNME" : "FIRSTNMEValue" ,
      "MIDINIT" : "MIDINITValue" ,
      "LASTNAME" : "LASTNAMEValue" ,
      "WORKDEPT" : "WORKDEPTValue" ,
      "PHONENO" : "PHONENOValue" ,
      "HIREDATE" : "2013-04-20" ,
      "JOB" : "JOBValue" ,
      "EDLEVEL" : 1 ,
      "SEX" : "SEXValue" ,
      "BIRTHDATE" : "2013-04-20" ,
      "SALARY" : 1.0 ,
      "BONUS" : 1.0 ,
      "COMM" : 1.0

    }
  }
};

//
```

*** Note *** Do not uncomment the example error response that starts with:

```
/* This is an example of the unexpected error JSON variable.
```

3. Scroll down and change only the value **EMPNO** to **'000010'** for the **getEmployeeVar** variable as shown below:

```
var getEmployeeVar =
{

  "EMPLOYEE" :
  {

    "EMPNO" : "000010" ,
    "FIRSTNME" : "FIRSTNME" ,
    "MIDINIT" : "D" ,
    "LASTNAME" : "LASTNAME" ,
    "WORKDEPT" : "BTW" ,
    "PHONENO" : "1212" ,
    "HIREDATE" : "2013-04-20" ,
    "JOB" : "JOB" ,
    "EDLEVEL" : 1 ,
    "SEX" : "M" ,
    "BIRTHDATE" : "2013-04-20" ,
    "SALARY" : 1.0 ,
    "BONUS" : 1.0 ,
    "COMM" : 1.0

  }
};
```

4. In order to see the response that has been returned in the Command Console, just before the very end of the file add the line

```
console.log(getEmployeeResponseVar);
```

after the line

```
console.log("Success for IBMIntegration.EmployeeService.getEmployee();
```

When you have made this change, it should look like this:

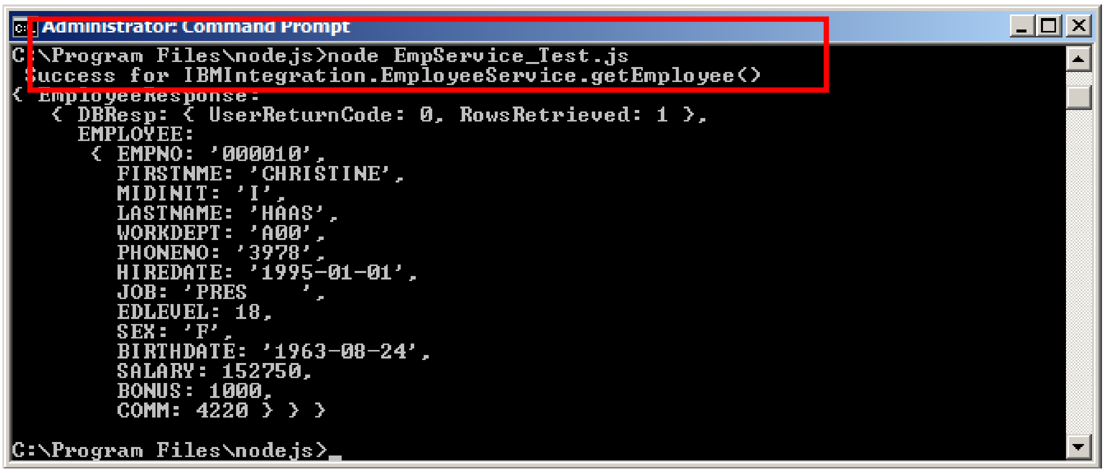
```
}
else {
 console.log("Success for IBMIntegration.EmployeeService.getEmployee() ");

 console.log(getEmployeeResponseVar);

   }

} );
```

Save the file with ‘Ctrl-S’.

4. Test the IIB JavaScript API using Node.js

1. In a command window prompt, navigate to the **c:\Program Files\nodejs** directory and execute the command:

```
node EmpService_Test.js
```



```
C:\Program Files\nodejs>node EmpService_Test.js
 Success for IBMIntegration.EmployeeService.getEmployee()
{ EmployeeResponse:
   { DBResp: { UserReturnCode: 0, RowsRetrieved: 1 },
     EMPLOYEE:
      { EMPNO: '000010',
        FIRSTNME: 'CHRISTINE',
        MIDINIT: 'I',
        LASTNAME: 'HAAS',
        WORKDEPT: 'A00',
        PHONENO: '3978',
        HIREDATE: '1995-01-01',
        JOB: 'PRES    ',
        EDLEVEL: 18,
        SEX: 'F',
        BIRTHDATE: '1963-08-24',
        SALARY: 152750,
        BONUS: 1000,
        COMM: 4220 } } }

C:\Program Files\nodejs>_
```

This will return data from the Employee Service.
You will recall that record 000010 is Christine Haas in the Employee table.

To summarize, you have just executed a JavaScript application that has used the generated API to invoke an Integration Service in IBM Integration Bus.

5. Test IIB JavaScript API in a Web browser

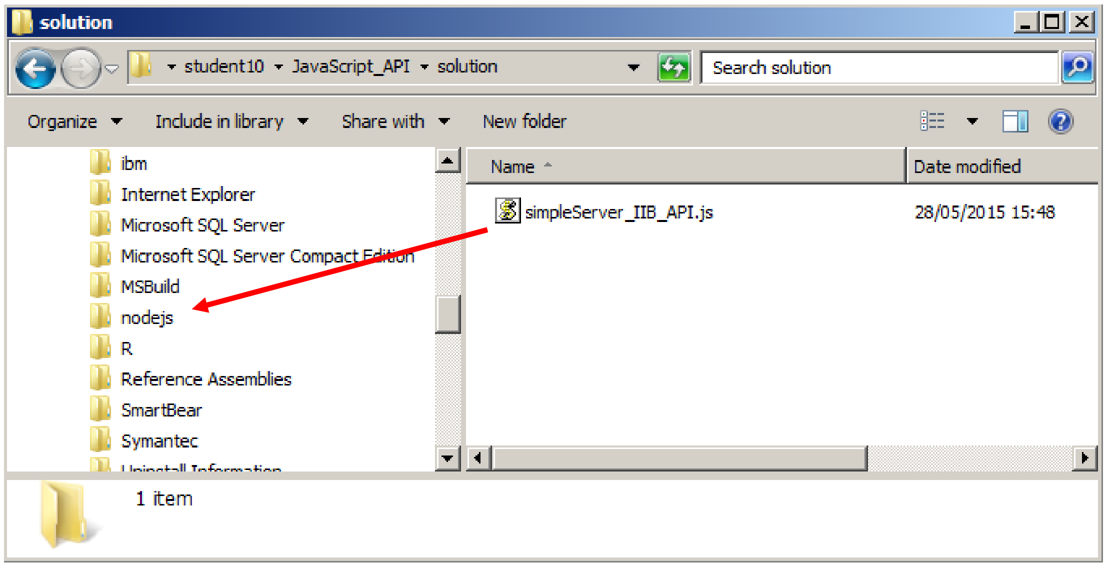
To show you how this can be executed for use in a Web Browser, we will use a browser to make a call to a JavaScript application hosted in an HTTP server. This will use the generated API to invoke the EmployeeService, which will retrieve data from a database.

For this scenario there are a number of HTTP servers that can be used. For simplicity, our example will use Node.js to start an HTTP server. This will run our JavaScript application.

1. In Windows Explorer go to **`C:\student10\JavaScript_API\solution`** folder and copy the file **`simpleServer_IIB_API.js`** to the Node.js installation directory (**c:/Program Files/nodejs**).

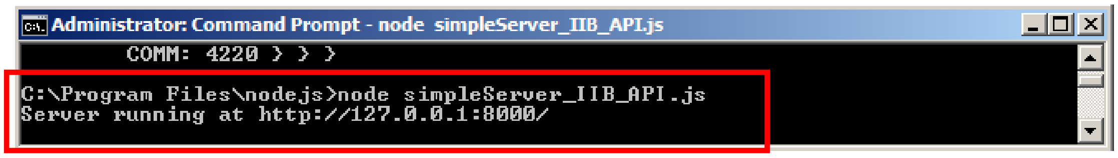
   This contains JavaScript code that builds an HTML page.

   Please make sure that the file **`EmployeeService.js`** also exists in the Node.js installation directory (this is the file that you copied in the previous section).



2. In a command window, navigate to the c:/node.js directory and execute the command:

```
node simpleServer_IIB_API.js
```



Administrator: Command Prompt - node simpleServer_IIB_API.js

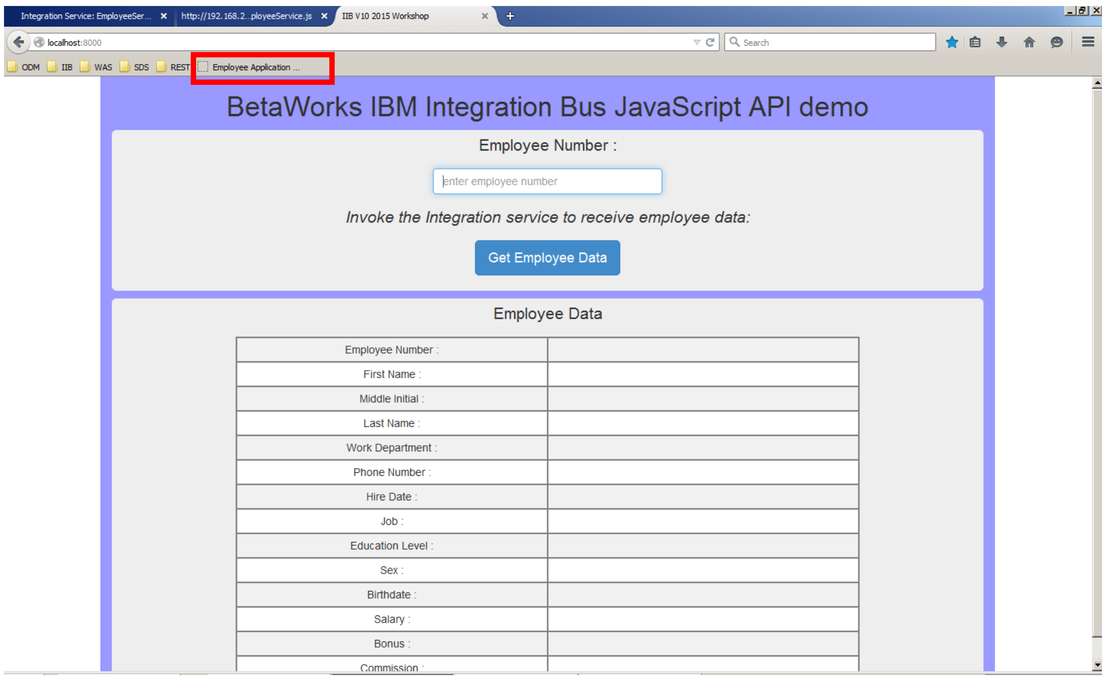
```
        COMM:  4220 > > >

C:\Program Files\nodejs>node simpleServer_IIB_API.js
Server running at http://127.0.0.1:8000/
```

The HTTP server has been started.

- **Important: Please do not close the Command Prompt, as this will stop the server. You can however, minimize the window.**

3. In a Web Browser, navigate to http://127.0.0.1:8000 (or http://localhost:8000). You can also use the 'Employee Application' bookmark available in your Browser window.

   This will open the Web page of the JavaScript API Web application, which has been developed for this lab exercise:



BetaWorks IBM Integration Bus JavaScript API demo

Employee Number :

enter employee number

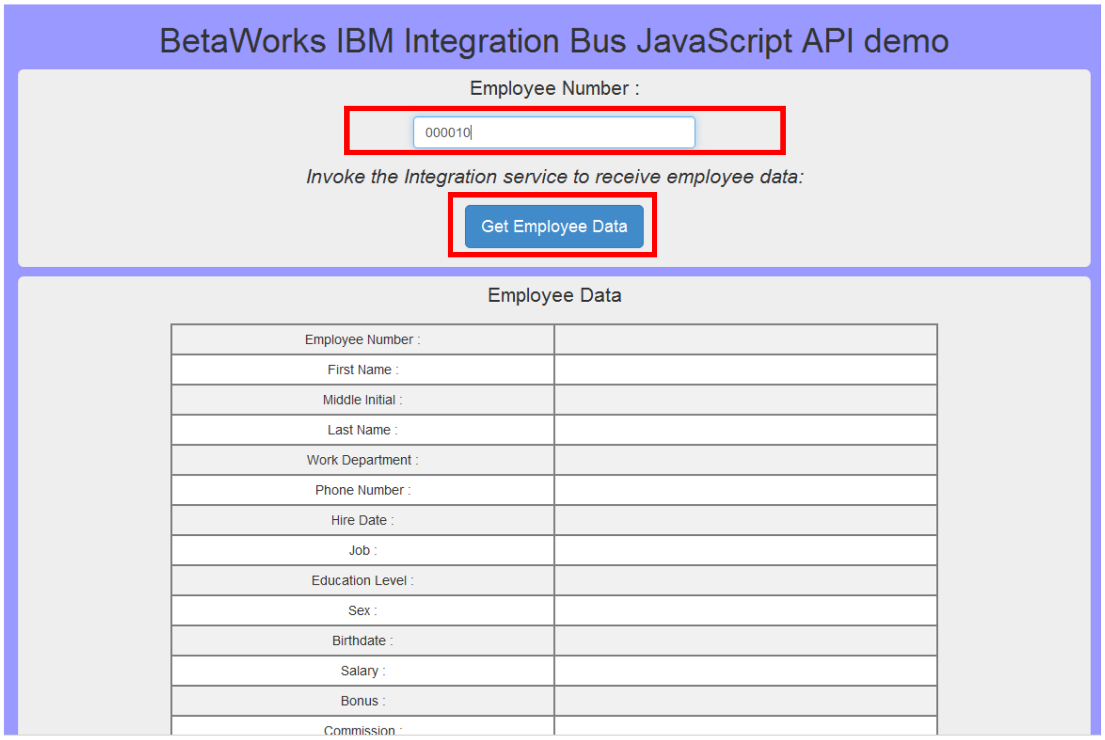
Invoke the Integration service to receive employee data:

Get Employee Data

Employee Data

Employee Number :	
First Name :	
Middle Initial :	
Last Name :	
Work Department :	
Phone Number :	
Hire Date :	
Job :	
Education Level :	
Sex :	
Birthdate :	
Salary :	
Bonus :	
Commission :	

4. Enter an employee number 000010 in the field as shown and press ‘Get Employee Data’



BetaWorks IBM Integration Bus JavaScript API demo

Employee Number :

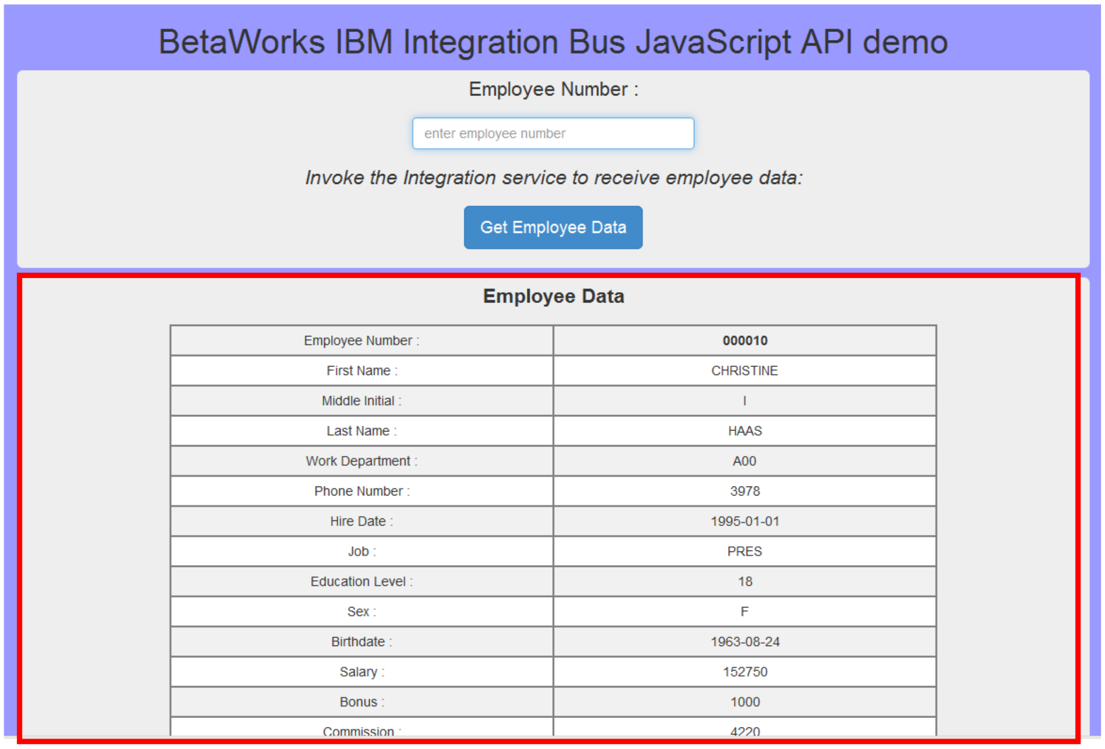
000010

Invoke the Integration service to receive employee data:

Employee Data

Employee Number :	
First Name :	
Middle Initial :	
Last Name :	
Work Department :	
Phone Number :	
Hire Date :	
Job :	
Education Level :	
Sex :	
Birthdate :	
Salary :	
Bonus :	
Commission :	

5. The data will be shown in the 'Employee Data' field. You may need to scroll down to view the bottom rows.


BetaWorks IBM Integration Bus JavaScript API demo
Employee Number :
enter employee number
Invoke the Integration service to receive employee data:
Get Employee Data
Employee Data

Employee Number :	**000010**
First Name :	CHRISTINE
Middle Initial :	I
Last Name :	HAAS
Work Department :	A00
Phone Number :	3978
Hire Date :	1995-01-01
Job :	PRES
Education Level :	18
Sex :	F
Birthdate :	1963-08-24
Salary :	152750
Bonus :	1000
Commission :	4220

Please feel free to test the application using other Employee numbers. Some examples of existing 'employees' in the Database are **000020**, **000030**, **000050**.

Note: If you enter a non-existent Employee Number, the Web Browser response will not show an exception error and will time out after a given time.
However, you will be able to see the error in the Command Console window.

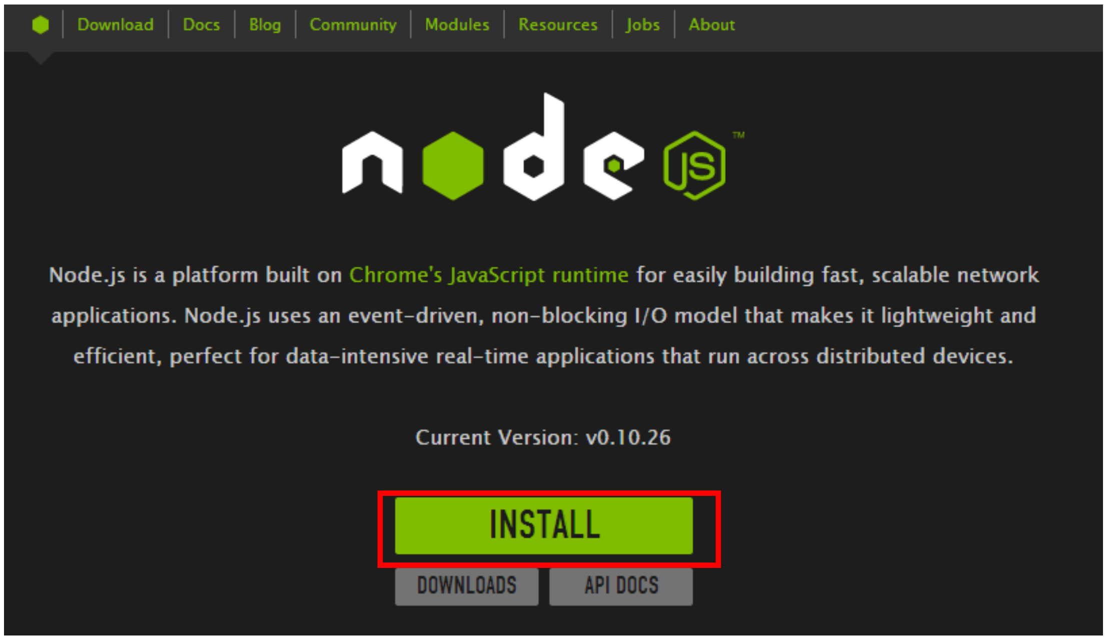
This completes the lab.

6. Appendix

6.1 Install Node.js

Node is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

1. Go to http://www.nodejs.org and install the package.

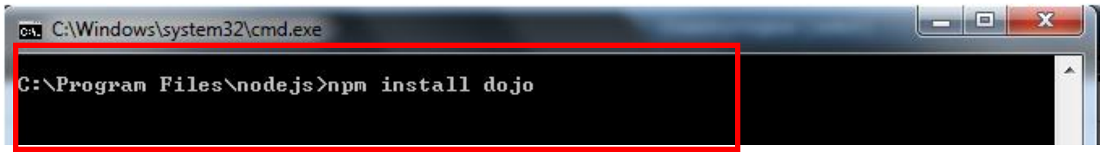


Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Current Version: v0.10.26

2. Once the package has been installed, in a command prompt window navigate to its installation directory. The default location is **`C:/Program Files/nodejs`**;

   Run the command **`npm install dojo`**



```
C:\Program Files\nodejs>npm install dojo
```

The Node.js environment is now set for your application