IBM Integration Bus

Message Flow Security

Featuring:

The JSON Client Application
IBM Security Directory Server
Securing message flows with LDAP Authentication and
Authorisation (username and password)
Using clear text and SSL connections to LDAP
Extracting credentials from X.509 certificates for authorisation

January 2016

Hands-on lab built at product
Version 10.0.0.3

1. Change History

1.1 Version 10.0.0.2

- Dependency on Business Rules ODM removed (special version of checkBonus operation provided as starting point).
- Use CreateNodes command file to create IIB nodes with ODM database connections.

1.2 Version 10.0.0.3

Important note

This lab, version 10.0.0.3, has been updated significantly from earlier versions. The following changes have been made:

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).

The database has been changed from the DB2 SAMPLE database to the DB2 HRDB database. HRDB contains two tables, EMPLOYEE and DEPARTMENT. These tables have been populated with data required for this lab. (The DDL for the HRDB is available in the student10 folder; we intend to provide corresponding DDL for Microsoft SQL/Server and Oracle over time).

The map node now retrieves multiple rows from the database, using an SQL "LIKE" function . Additionally, the map has been restructured to use a submap, located in its own shared library. The submap is invoked from the main map, also located in the shared library.

Input to the integration service is now a simple schema containing just one element, the required employee number.

As a consequence, this version of the lab, and the associated solution, can only be used with the corresponding changes in other labs. Use version 10.0.0.3 of all labs in this series of lab guides.

2. Scenarios

This lab will investigate two techniques for message flow security:

1. Authentication and authorisation with username/password, using an LDAP database for both authentication and authorisation.
    a. Authentication only, using a clear text LDAP connection
    b. Authentication and authorisation, using a clear text LDAP connection
    c. Authentication and authorisation, using a secure (SSL) LDAP connection

2. Authentication and authorisation with X.509 certificates
    a. Authentication provided by local certificate key/truststores
    b. Authorisation provided by LDAP/S, based on the Distinguished Name in the X.509 certificate

Both scenarios will use the familiar EmployeeService service and the EmployeeService_JSONClient application. Both the EmployeeService service and all the JSON Client message flows will be deployed to the same IIB server, called PROVIDER.

Message Flow authentication and authorisation will be applied in stages to the message flows contained within the EmployeeService_JSONClient application. Each of these flows will be allowed (or denied), based on the defined LDAP users, and whether these users are members of the appropriate LDAP groups.

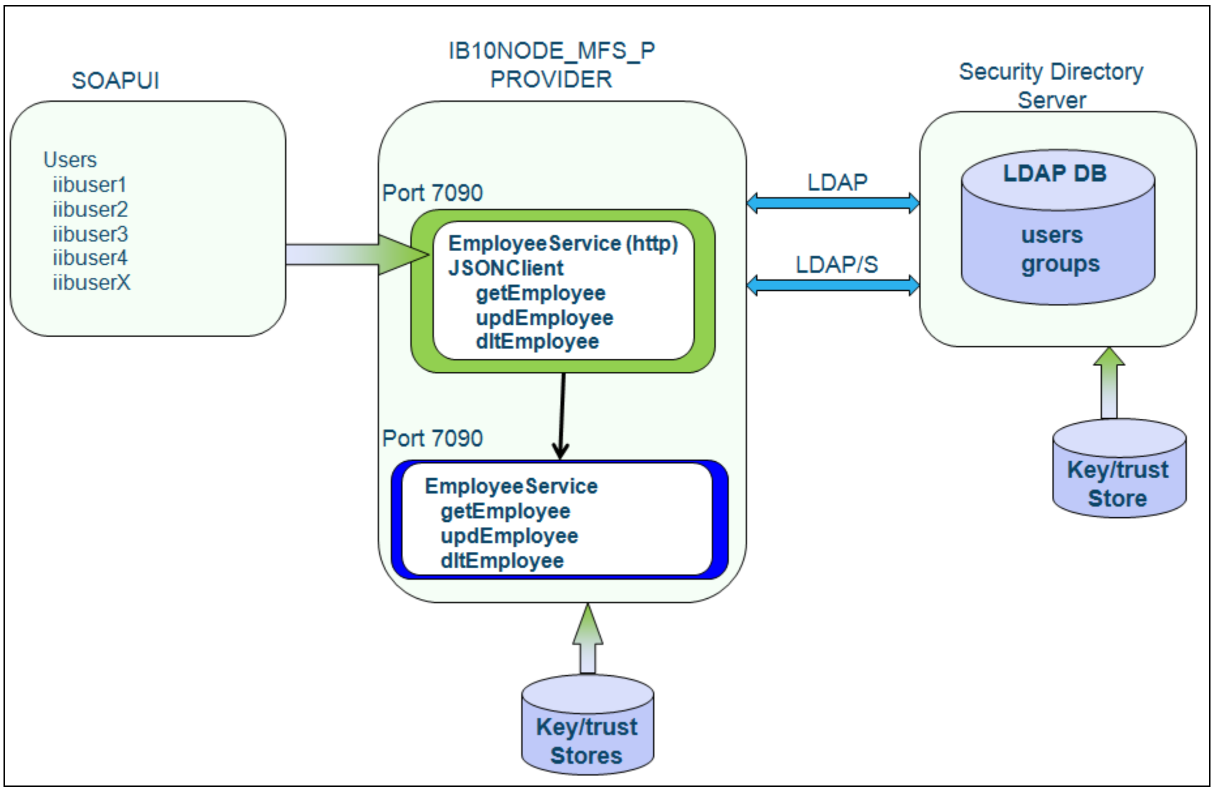
In this lab, the EmployeeService service operations will not be subject to any security protection.

2.1 Scenario 1: Message flow security with username / password

For the first scenario, the following schematic shows the overall component architecture.

Both the EmployeeService and JSONClient accept http input on the same port, 7090.

All flow tests are driven by SOAPUI; a number of SOAPUI projects are provided for these tests.



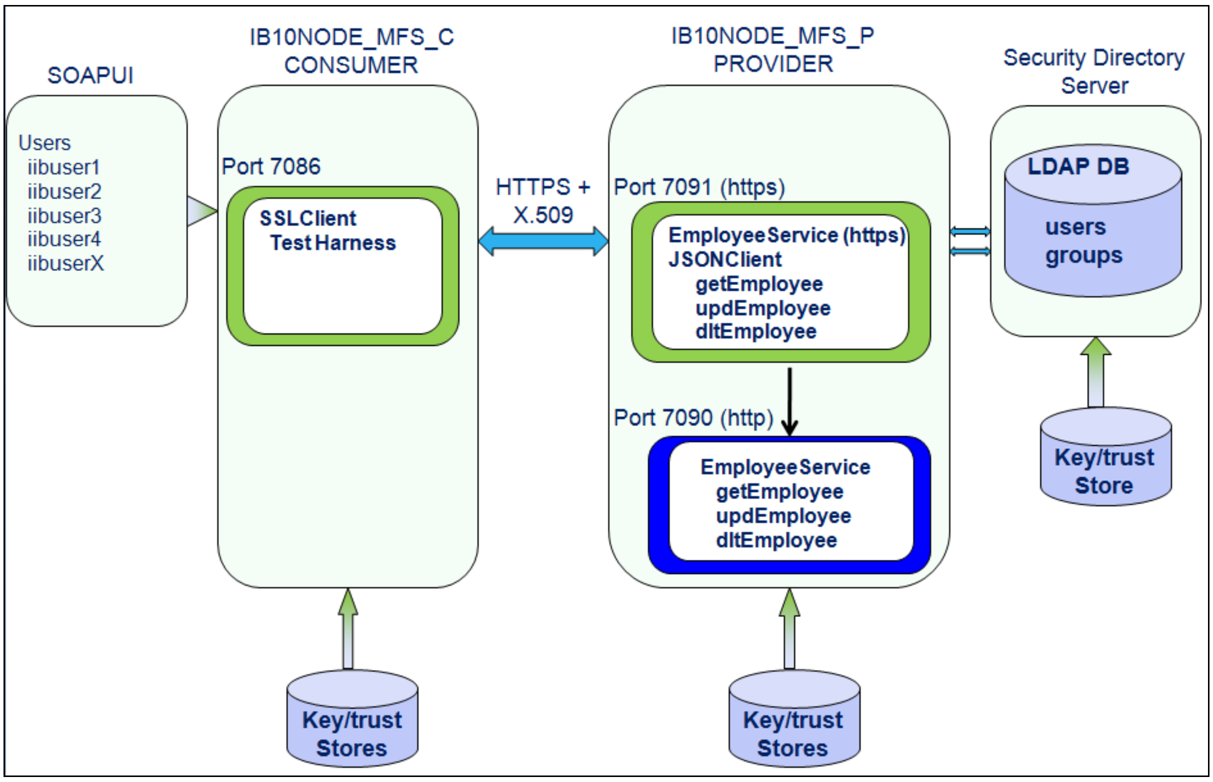
2.2 Scenario 2: Message flow security with X.509 certificates

As in scenario 1, both the EmployeeService and JSONClient applications are located in PROVIDER.

In addition, a test harness, SSLClient, is deployed in IIB server CONSUMER. This is required because this scenario includes the transmission of an X.509 certificate, which does not appear to be supported by SOAPUI, when sending a REST call.

The SSLClient message flow is invoked by SOAPUI.

Note that in this scenario, the JSONClient application runs over https, so now receives input on port 7091, an HTTPS port.



3. Prepare the Provider Service

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

3.1 Create the new IIB nodes

Since this lab will make some security changes to the IIB nodes, you will use a script to create two new nodes, avoiding any contamination of other IIB nodes.

The script will create two nodes, and the applications will be deployed as follows:

- Node = IB10NODE_MFS_P
  - Server = PROVIDER
    - EmployeeService
    - EmployeeService_JSONClient
    - EmployeeService_interface_and_maps

- Node = IB10NODE_MFS_C
  - Server = CONSUMER
    - SSLClient_TestHarness

1. Open the IIB Event Log Monitor (from the Windows Start menu).

2. In an IIB Command Console, switch to the folder:

   C:\ student10 \ Integration_service_MessageFlowSecurity \ commands

   Run the command file "**CreateNodes**". Accept the default values for node names, server names and port numbers (4432, 4431), then press Return.

   This command file will pause until the webadmin listener ports are started, probably on ports 4415 and 4416. When you see these messages on the IB Event Log Monitor, press Return to continue the script. The script will change the port numbers to those given above, and the nodes will stop and start several times during the creation.

   This command file will create the IIB nodes using embedded HTTP listeners (and HTTPS) (ie. embedded in the IIB server).

   Take a look at the command file to see the various IIB commands that are used to achieve this.

3.2 Import and deploy the IIB Applications

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 "MessageFlowSecurity", or similar.

2. Import the PI file **`c:\student10\Integration_service_MessageFlowSecurity\applications\MFS_StartingPoint.10.0.0.3.zip`**

   Ensure all projects are selected, and import them into the workspace.

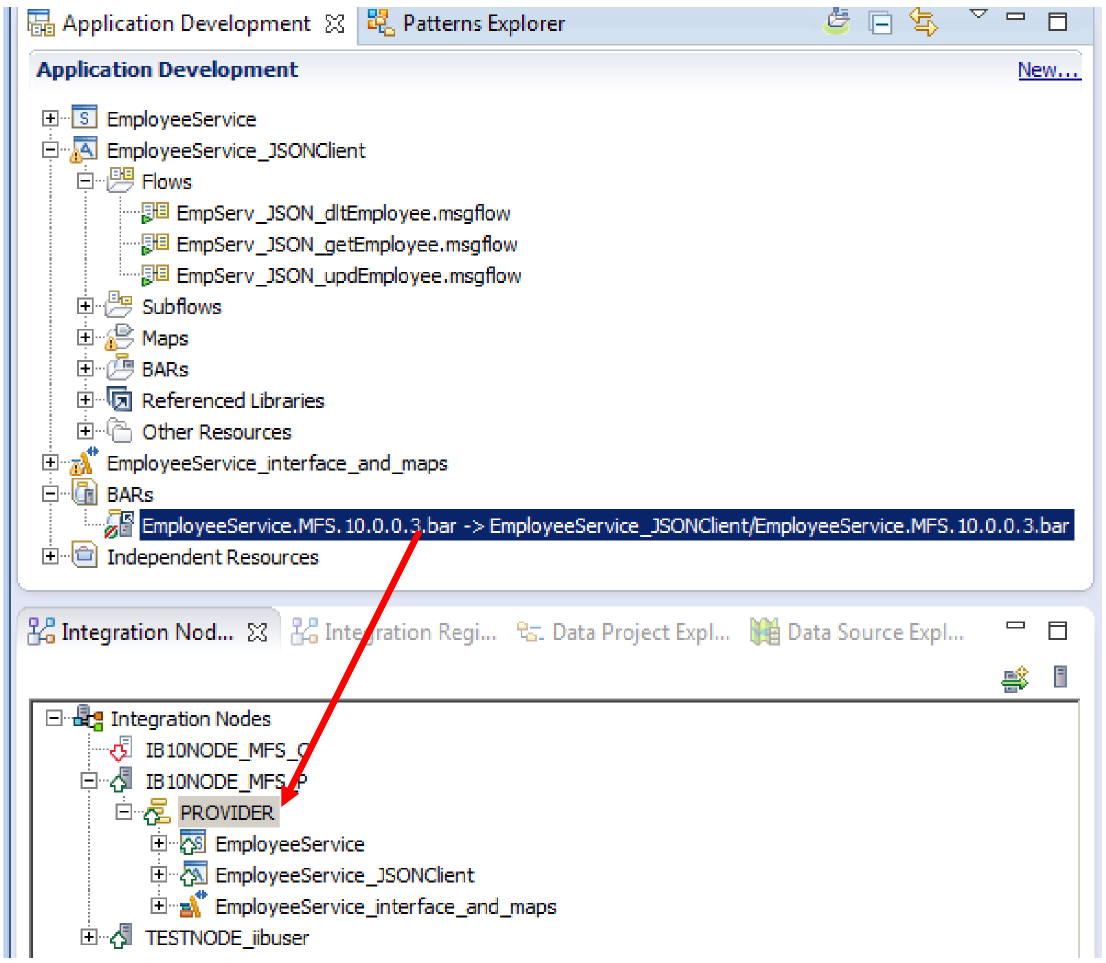
3. You may need to refresh the Integration Nodes in the Toolkit view (right-click, refresh).

   Then, deploy the following barfile to IB10NODE_MFS_P / PROVIDER:

   `EmployeeService.MFS.10.0.0.3.bar`

   Note, all barfiles in the workspace are visible under the BARs folder. This bar file contains all three projects that are required for this lab.

   The script defined an embedded listener, and disabled the node-wide listener, so all flows will use port 7090. The applications (message flows) built into the provided barfile use port 7090 for the EmployeeService operations.

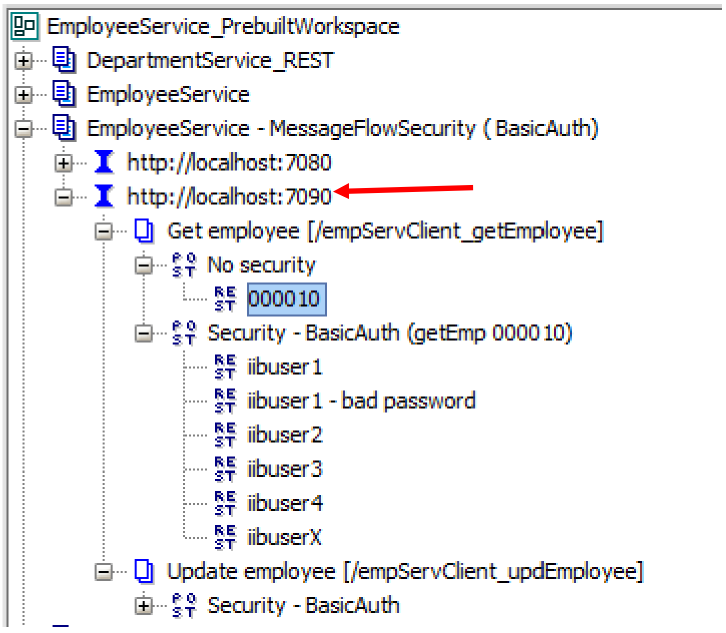


4. As a quick test, to make sure the applications are working, use SOAPUI to execute this service.

   Open SOAPUI, and open the SOAPUI workspace EmployeeService_PrebuiltWorkspace.

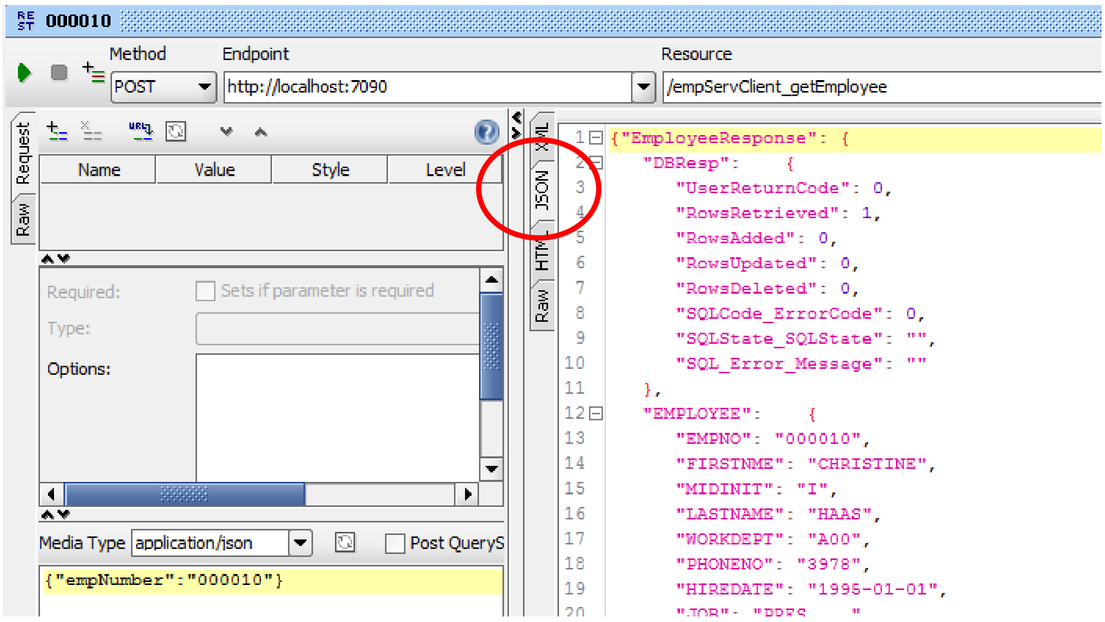
   Open the project "EmployeeService - MessageFlowSecurity, BasicAuth".

   Expand the service "http://localhost:7090", Get Employee, and the "No security" method. Open the test for 000010.



```
EmployeeService_PrebuiltWorkspace
  DepartmentService_REST
  EmployeeService
  EmployeeService - MessageFlowSecurity ( BasicAuth)
    http://localhost:7080
    http://localhost:7090
      Get employee [/empServClient_getEmployee]
        No security
          000010
        Security - BasicAuth (getEmp 000010)
          iibuser1
          iibuser1 - bad password
          iibuser2
          iibuser3
          iibuser4
          iibuserX
      Update employee [/empServClient_updEmployee]
        Security - BasicAuth
```

Clicking the green arrow should return the following data. Note the number of rows retrieved is 1.
This means that you have invoked the JSON Client application, which in turn has invoked the EmployeeService to retrieve employee details. The data has been returned in JSON format.



000010

Method: POST
Endpoint: http://localhost:7090
Resource: /empServClient_getEmployee

Name	Value	Style	Level

Required: Sets if parameter is required
Type:
Options:

Media Type: application/json Post QueryS

```
{"empNumber":"000010"}
```

```
{"EmployeeResponse": {
   "DBResp":    {
      "UserReturnCode": 0,
      "RowsRetrieved": 1,
      "RowsAdded": 0,
      "RowsUpdated": 0,
      "RowsDeleted": 0,
      "SQLCode_ErrorCode": 0,
      "SQLState_SQLState": "",
      "SQL_Error_Message": ""
   },
   "EMPLOYEE":    {
      "EMPNO": "000010",
      "FIRSTNME": "CHRISTINE",
      "MIDINIT": "I",
      "LASTNAME": "HAAS",
      "WORKDEPT": "A00",
      "PHONENO": "3978",
      "HIREDATE": "1995-01-01",
```

4. Configure and test client authentication with LDAP

In this section you will configure IIB to perform authentication of user requests, using username and password credentials from the incoming HTTP requests.

In this section, the user authentication is provided by an LDAP server and database, provided by IBM Security Directory Server, installed on the same VM as IIB.

The following tasks will be done:

- Use the IIB web administration tool to define three new Security Profiles for each of the authorisation groups.

- Use the IIB Toolkit to edit the barfile, associating the Security Profiles as follows:
    - Message Flow getEmployee uses iibauth_default
    - Message Flow updEmployee uses iibauth_update
    - Message Flow dltEmployee uses iibauth_delete

- Deploy the updated barfile

- Run various tests with different users

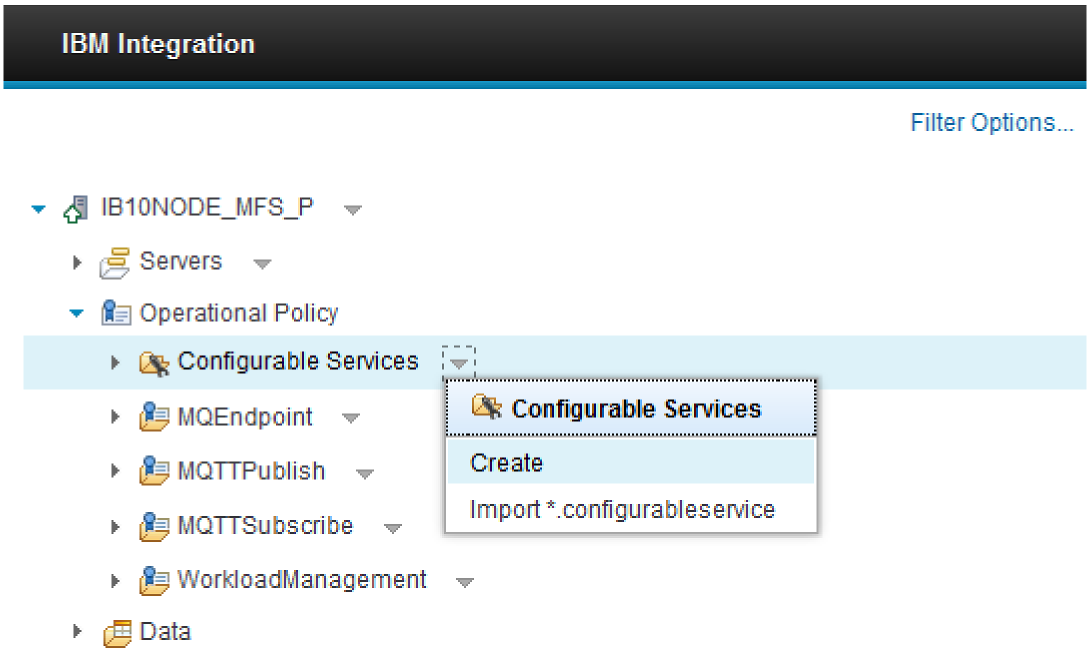
4.1 Create the Security Profiles - clear text connectivity to LDAP

1. In the Firefox web browser, login to IB10NODE_MFS_P (port 4432), using the provided shortcut in the IIB Nodes folder (https://localhost:4432).

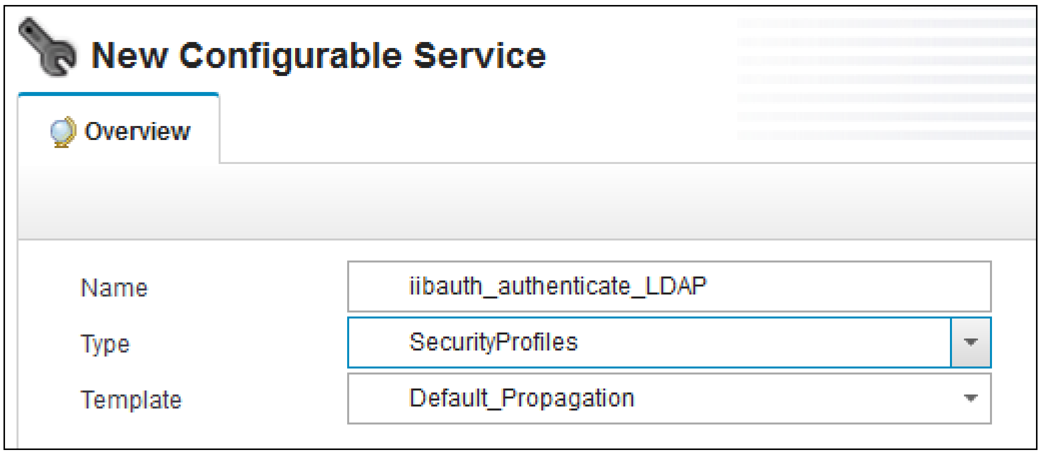
   Confirm the **Security Exception** if you get one.

   Expand Operational Policy, click the down-arrow to create a new Configurable Service.

   Click Create.



2. Name the new configurable service **iibauth_authenticate_LDAP**, and set the Type = SecurityProfiles.



Name	iibauth_authenticate_LDAP
Type	SecurityProfiles
Template	Default_Propagation

3. Set the following property values:

   - **passwordValue = MASK** (not required for the lab, but good practice to avoid the password being shown in the Properties tree).

   - **authentication = LDAP**

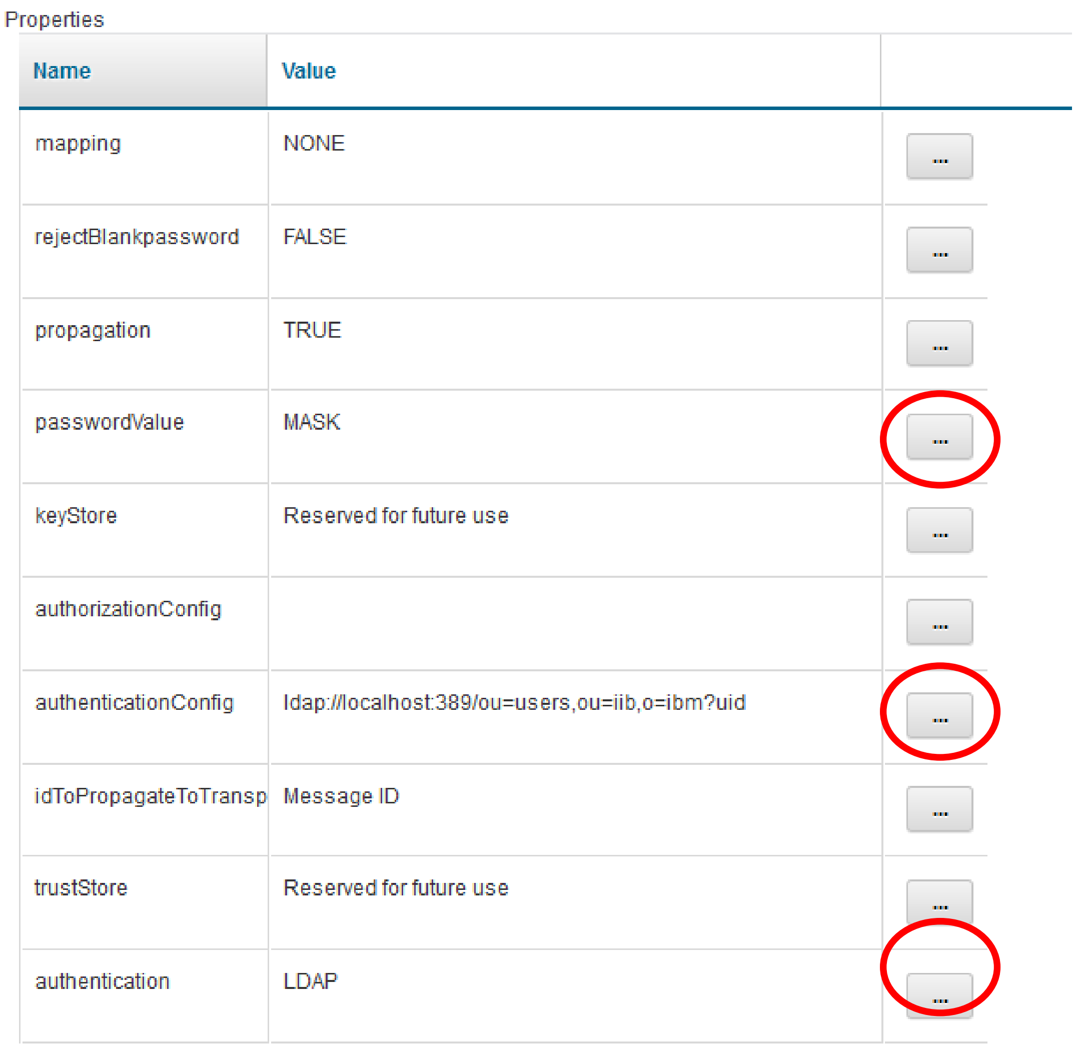
   - **authenticationConfig = ldap://localhost:389/ou=users,ou=iib,o=ibm?uid**

     This value means that the LDAP will search for the given user under the LDAP hierarchy shown - **be very careful when typing this - check your work before clicking Save !**

LDAP Notes:

- The LDAP server (IBM Security Directory Server) is running on the same system as IIB, and uses port 389 for plain text communications.
- The LDAP users are defined under the LDAP directory domain name ou=users,ou=iib,o=ibm
- The LDAP user field that is used by IIB to identify each user is "uid" (hence the ?uid suffix on the authenticationConfig property).

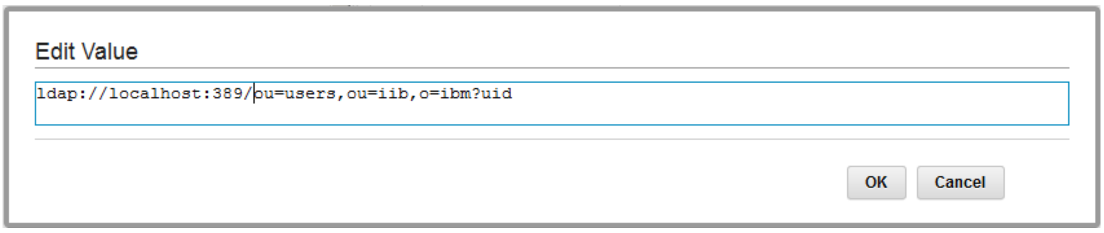
To set the values for each property, use the Edit button (three dots), or double-click the appropriate Value field.


Properties

Name	Value	
mapping	NONE	...
rejectBlankpassword	FALSE	...
propagation	TRUE	...
passwordValue	MASK	...
keyStore	Reserved for future use	...
authorizationConfig		...
authenticationConfig	ldap://localhost:389/ou=users,ou=iib,o=ibm?uid	...
idToPropagateToTransp	Message ID	...
trustStore	Reserved for future use	...
authentication	LDAP	...

4. Type the required value for each property, as shown here. Click OK when complete. This value is for the authenticationConfig property.

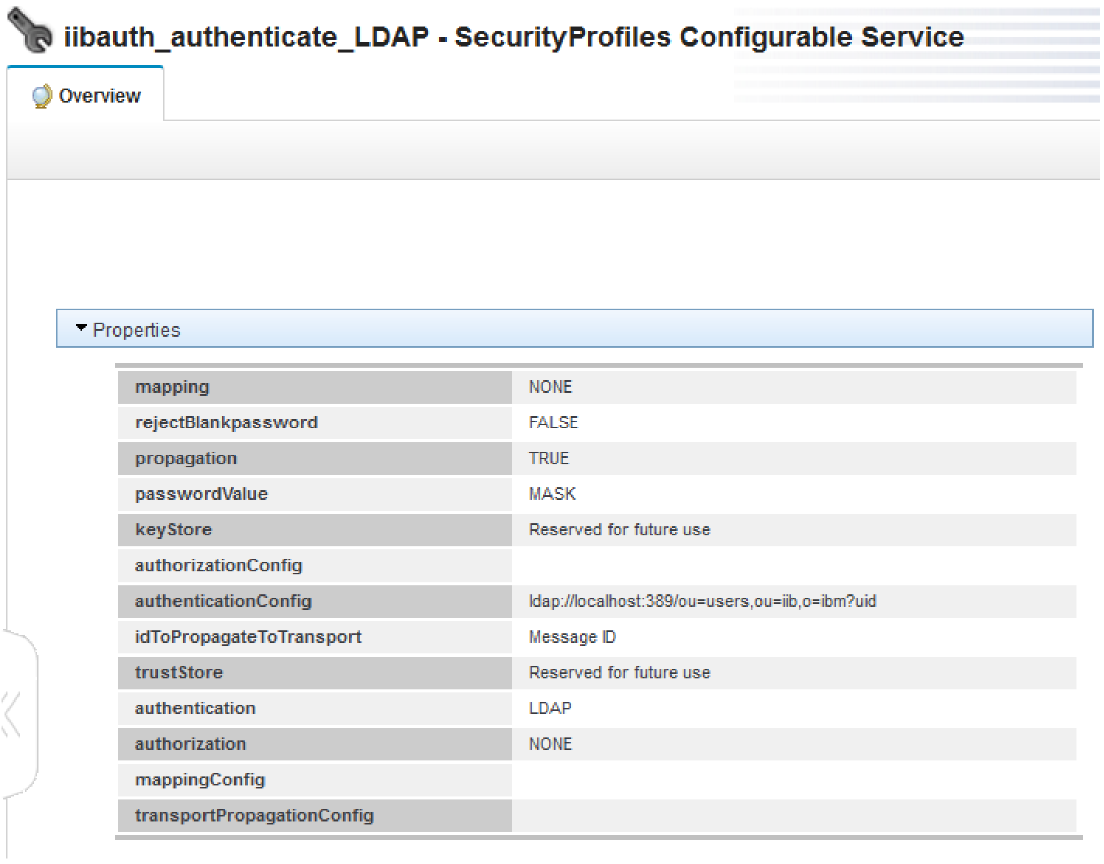
   As a reminder: **authenticationConfig = ldap://localhost:389/ou=users,ou=iib,o=ibm?uid**



```
ldap://localhost:389/ou=users,ou=iib,o=ibm?uid
```

   When you have made all the changes, click Save. The Save function will update the IB10NODE_MFS_P server in real-time. When complete, you will briefly see a success message.

5. The final result will look like this:


iibauth_authenticate_LDAP - SecurityProfiles Configurable Service
Overview
Properties

mapping	NONE
rejectBlankpassword	FALSE
propagation	TRUE
passwordValue	MASK
keyStore	Reserved for future use
authorizationConfig	
authenticationConfig	ldap://localhost:389/ou=users,ou=iib,o=ibm?uid
idToPropagateToTransport	Message ID
trustStore	Reserved for future use
authentication	LDAP
authorization	NONE
mappingConfig	
transportPropagationConfig	

4.2 Configure the IIB node to connect to the LDAP Server

Before any message flow credentials can be validated with the LDAP server, the IIB node itself must connect to the LDAP server. It must do so using a username that is defined as an administrator of the LDAP.

In our lab scenario, we have kept to the default values of the Security Directory Server as far as possible. The administrator of the LDAP system is the user "**cn=root**"**,** so be careful when you type this in the command below.

1. In an IIB Command Console, issue the command:

   ```
   mqsisetdbparms IB10NODE_MFS_P
         -n ldap::LDAP
         -u cn=root
         -p passw0rd
   ```

   Then:

   ```
   mqsistop IB10NODE_MFS_P
   mqsistart IB10NODE_MFS_P
   ```

   These commands are also provided in the **\commands\ConfigureLDAPSecurity.cmd** file.

   (Note - this will set the LDAP connection credentials for the entire IIB node; it is of course possible to specify the credentials at the server level - see the Info Centre for further details).

2. As the node starts, watch for the various HTTP listeners starting, in the IIB Event Log Monitor.

   The webadmin listener starts on 4432 (https).

   The embedded server listener for http starts on 7090.

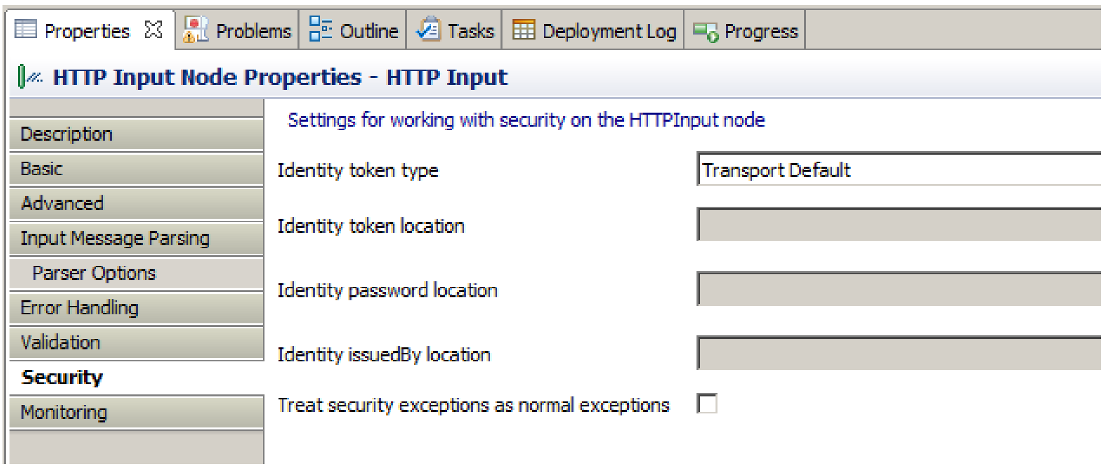
   ```
   BIP3132I: ( IB10NODE_MFS_P ) The HTTP Listener has started listening
   on port ''4432'' for ''WebAdmin https'' connections. [24/03/2015
   09:24:10]
   BIP2152I: ( IB10NODE_MFS_P.PROVIDER ) Configuration message received
   from integration node. [24/03/2015 09:24:14]
   BIP2153I: ( IB10NODE_MFS_P.PROVIDER ) About to ''Start'' an
   integration server. [24/03/2015 09:24:14]
   BIP3132I: ( IB10NODE_MFS_P.PROVIDER ) The HTTP Listener has started
   listening on port ''7090'' for ''http'' connections. [24/03/2015
   09:24:18]
   BIP2154I: ( IB10NODE_MFS_P.PROVIDER ) Integration server finished
   with Configuration message. [24/03/2015 09:24:24]
   ```

5. Configure message flows and test authentication

5.1 Review the message flow security properties

1. The security profile will be applied to the flows in the EmployeeService_JSONClient application, so expand this, and open the message flow **EmpServ_JSON_getEmployee**.

   In fact, for this first scenario, no changes are required for the message flow. In other similar scenarios, it may be necessary to change the HTTP input node to retrieve the user credentials from a non-standard part of the input message. However, on the Security tab, the default value of "Transport Default" for the Identity Token Type will be sufficient in this case.



HTTP Input Node Properties - HTTP Input

Settings for working with security on the HTTPInput node

Identity token type: Transport Default
Identity token location:
Identity password location:
Identity issuedBy location:
Treat security exceptions as normal exceptions: ☐

5.2 Configure the barfile to reference the Security Profiles

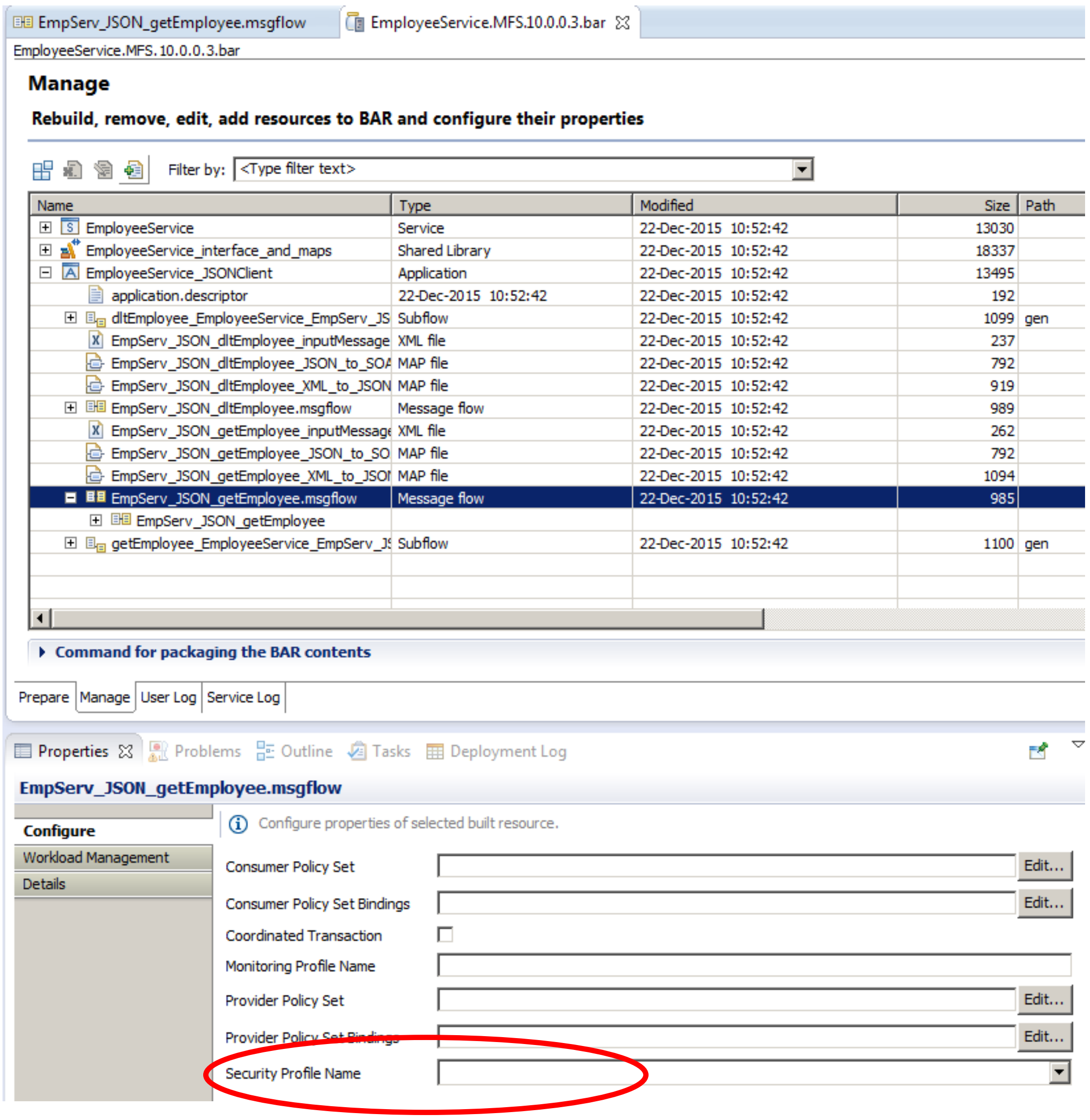
It's now necessary to associate the new Security Profile with the message flows that are contained within the application.

At this stage, since you are just configuring authentication, each message flow will use the iibauth_authenticate_LDAP security profile.

1. In the IIB Toolkit, open the EmployeeService_JSONClient.MFS.10.0.0.3.bar barfile.

   On the Manage tab, expand the EmployeeService_JSONClient application, and highlight the EmpServ_JSON_getEmployee message flow.

   In the Properties pane, on the Configure tab, you will see that the Security Profile Name is blank (or it may have been set to Default Propagation).


EmpServ_JSON_getEmployee.msgflow
EmployeeService.MFS.10.0.0.3.bar
EmployeeService.MFS.10.0.0.3.bar

Manage

Rebuild, remove, edit, add resources to BAR and configure their properties

Filter by: <Type filter text>

Name	Type	Modified	Size	Path
EmployeeService	Service	22-Dec-2015 10:52:42	13030	
EmployeeService_interface_and_maps	Shared Library	22-Dec-2015 10:52:42	18337	
EmployeeService_JSONClient	Application	22-Dec-2015 10:52:42	13495	
application.descriptor	22-Dec-2015 10:52:42	22-Dec-2015 10:52:42	192	
dltEmployee_EmployeeService_EmpServ_JS	Subflow	22-Dec-2015 10:52:42	1099	gen
EmpServ_JSON_dltEmployee_inputMessage	XML file	22-Dec-2015 10:52:42	237	
EmpServ_JSON_dltEmployee_JSON_to_SOA	MAP file	22-Dec-2015 10:52:42	792	
EmpServ_JSON_dltEmployee_XML_to_JSON	MAP file	22-Dec-2015 10:52:42	919	
EmpServ_JSON_dltEmployee.msgflow	Message flow	22-Dec-2015 10:52:42	989	
EmpServ_JSON_getEmployee_inputMessage	XML file	22-Dec-2015 10:52:42	262	
EmpServ_JSON_getEmployee_JSON_to_SO	MAP file	22-Dec-2015 10:52:42	792	
EmpServ_JSON_getEmployee_XML_to_JSOI	MAP file	22-Dec-2015 10:52:42	1094	
EmpServ_JSON_getEmployee.msgflow	Message flow	22-Dec-2015 10:52:42	985	
EmpServ_JSON_getEmployee				
getEmployee_EmployeeService_EmpServ_JS	Subflow	22-Dec-2015 10:52:42	1100	gen

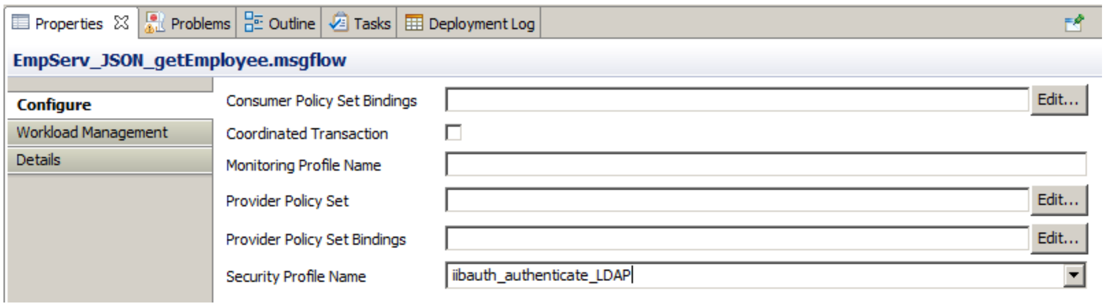
Command for packaging the BAR contents

EmpServ_JSON_getEmployee.msgflow

Configure properties of selected built resource.

Consumer Policy Set		Edit...
Consumer Policy Set Bindings		Edit...
Coordinated Transaction	☐	
Monitoring Profile Name		
Provider Policy Set		Edit...
Provider Policy Set Bindings		Edit...
Security Profile Name		

2. Change the Security Profile Name to iibauth_authenticate_LDAP

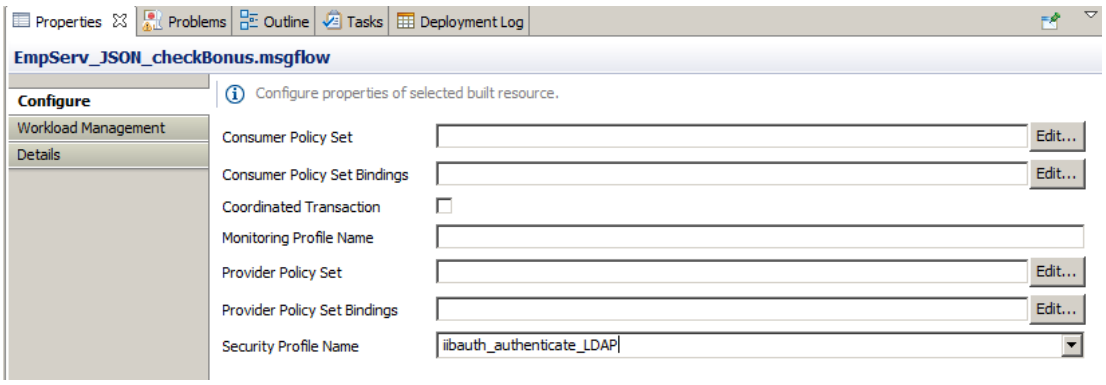

Properties | Problems | Outline | Tasks | Deployment Log

EmpServ_JSON_getEmployee.msgflow

Configure | Workload Management | Details

Consumer Policy Set Bindings	
Coordinated Transaction	☐
Monitoring Profile Name	
Provider Policy Set	
Provider Policy Set Bindings	
Security Profile Name	iibauth_authenticate_LDAP

3. Similarly, for the **EmpServ_JSON_update** and **EmpServ_JSON_delete** message flows, also change the Security Profile Name to iibauth_authenticate_LDAP.


Properties | Problems | Outline | Tasks | Deployment Log

EmpServ_JSON_checkBonus.msgflow

Configure | Workload Management | Details

Configure properties of selected built resource.

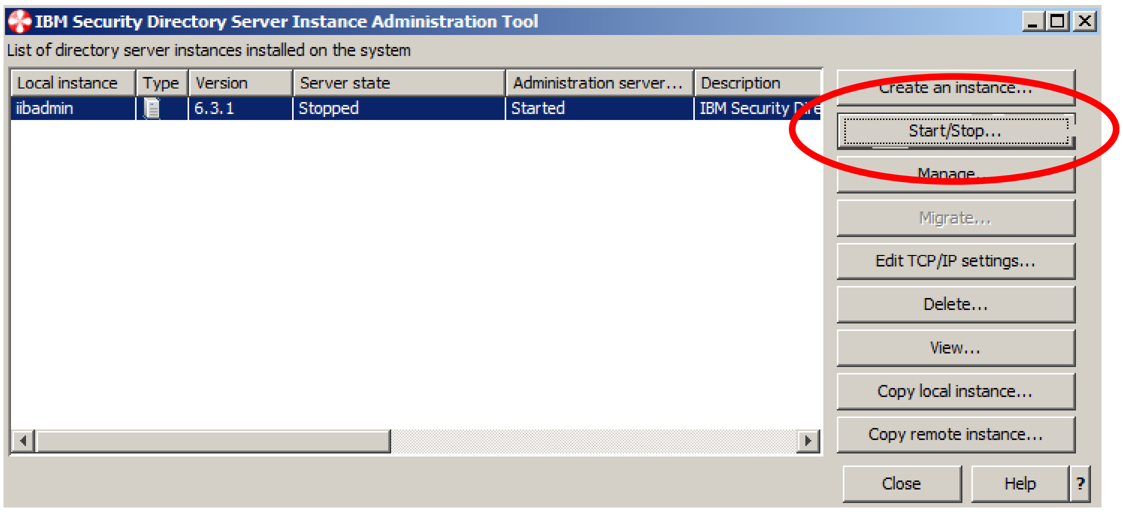
Consumer Policy Set	
Consumer Policy Set Bindings	
Coordinated Transaction	☐
Monitoring Profile Name	
Provider Policy Set	
Provider Policy Set Bindings	
Security Profile Name	iibauth_authenticate_LDAP

4. Save the updated barfile, and then deploy to IB10NODE_MFS_P / PROVIDER.

5.3 Test authentication against LDAP

1. If the LDAP (SDS) server is not running it must be started now. From the Desktop icon, open the SDS Instance Administration Tool. This icon will start SDS using the user "iibadmin". You will need to provide the password, passw0rd.

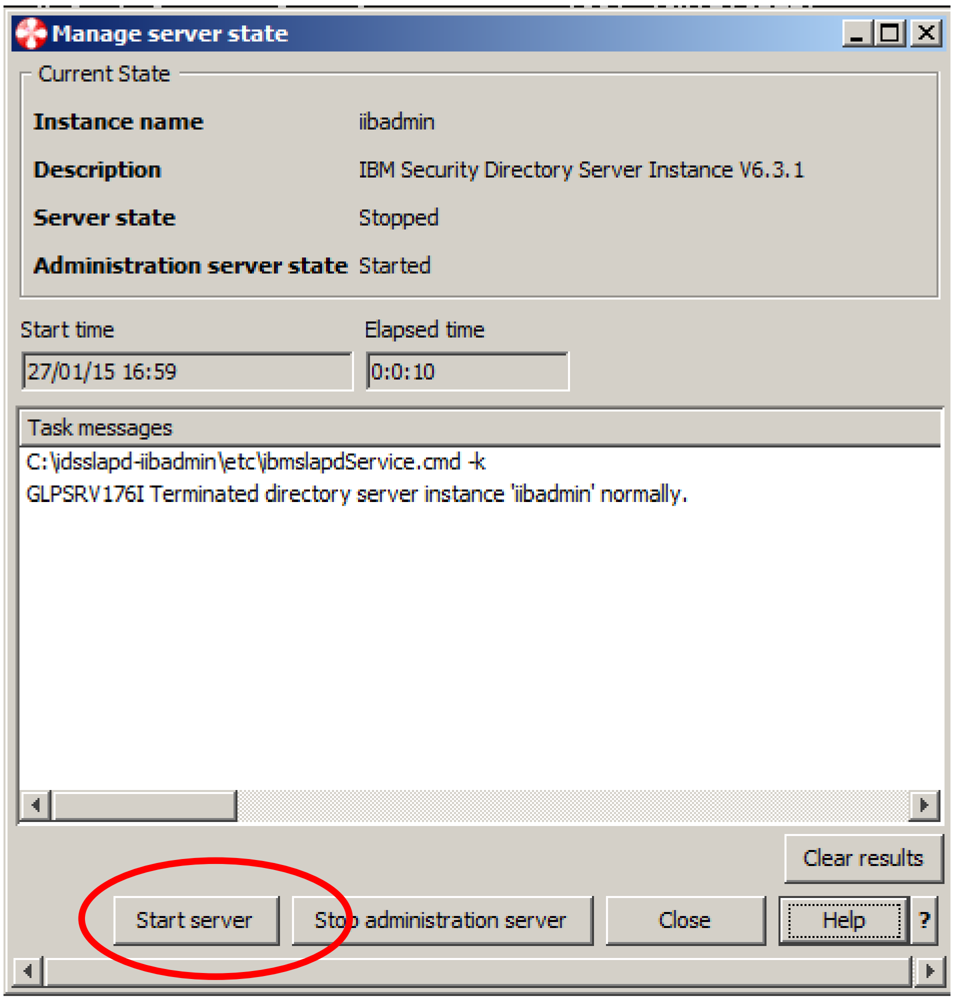
If the Administration Tool shows the instance "iibadmin" as Stopped, start it now, using the Start/Stop button, and the "Manage Server state" window which will open.


IBM Security Directory Server Instance Administration Tool

List of directory server instances installed on the system

Local instance	Type	Version	Server state	Administration server...	Description
iibadmin		6.3.1	Stopped	Started	IBM Security Dire

2. Click "Start server".



Current State

Instance name	iibadmin
Description	IBM Security Directory Server Instance V6.3.1
Server state	Stopped
Administration server state	Started

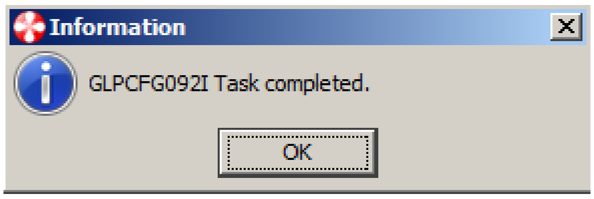
Start time: 27/01/15 16:59

Elapsed time: 0:0:10

Task messages

```
C:\idsslapd-iibadmin\etc\ibmslapdService.cmd -k
GLPSRV176I Terminated directory server instance 'iibadmin' normally.
```

After a few seconds, you will see a success window. Click OK to close, then close the server admin tool.

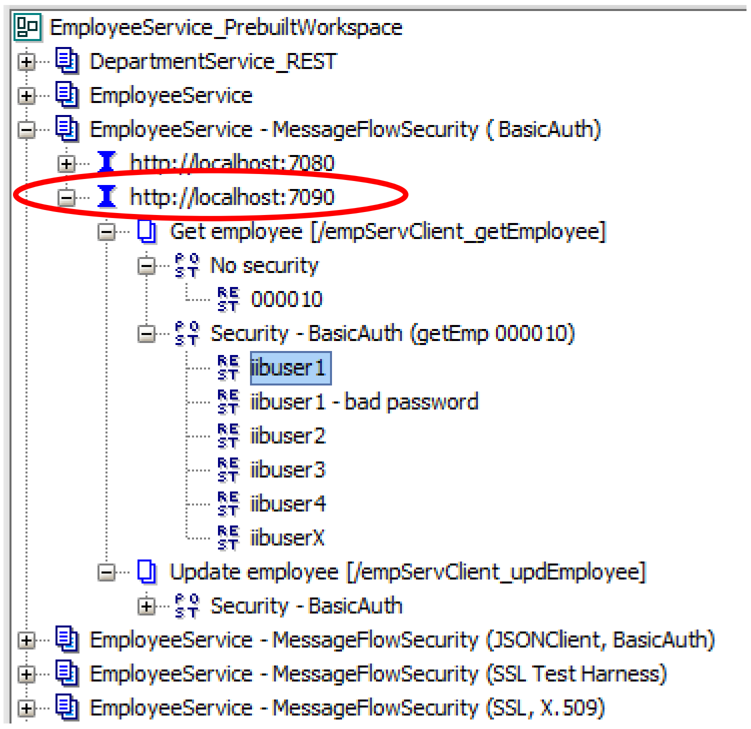


GLPCFG092I Task completed.

3. Now switch to SOAPUI and expand the project named

   `"EmployeeService - MessageFlowSecurity, BasicAuth".`

   Expand the **localhost:7090** service in this project. You will see both the "Get Employee" and "Update employee" resources are present.



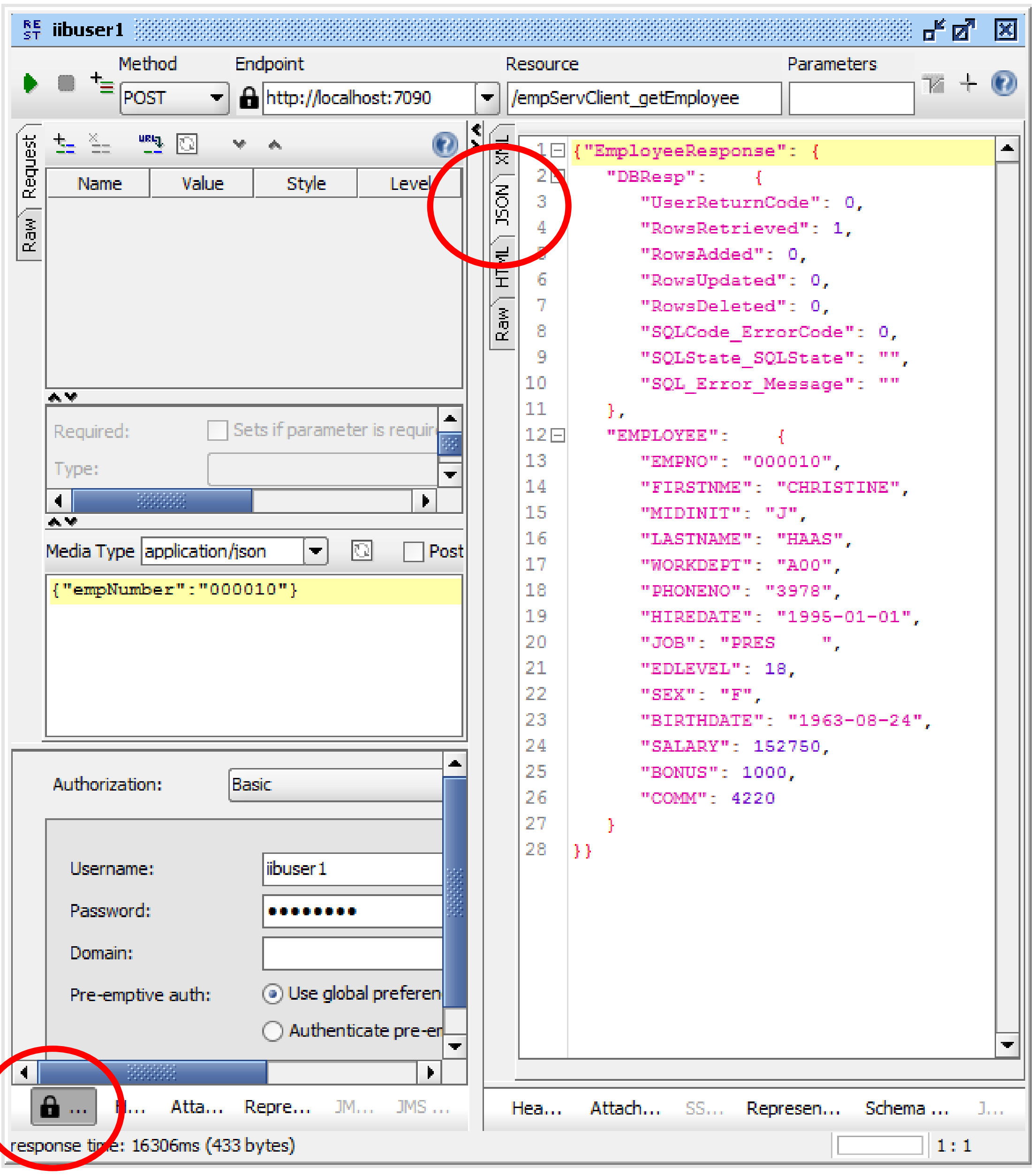
```
EmployeeService_PrebuiltWorkspace
 + DepartmentService_REST
 + EmployeeService
 - EmployeeService - MessageFlowSecurity ( BasicAuth)
    + http://localhost:7080
    - http://localhost:7090
       - Get employee [/empServClient_getEmployee]
          - No security
               000010
          - Security - BasicAuth (getEmp 000010)
               iibuser1
               iibuser1 - bad password
               iibuser2
               iibuser3
               iibuser4
               iibuserX
       - Update employee [/empServClient_updEmployee]
          + Security - BasicAuth
 + EmployeeService - MessageFlowSecurity (JSONClient, BasicAuth)
 + EmployeeService - MessageFlowSecurity (SSL Test Harness)
 + EmployeeService - MessageFlowSecurity (SSL, X.509)
```

4. In the Get Employee resource, expand the "Security - BasicAuth" method, and open the **iibuser1** request.

   Click the padlock icon to show the credentials associated with this request (iibuser1/passw0rd).

   Run this request (click the green Send arrow).

   The test should work, which means that the iibuser1 user has been authenticated against the LDAP database. You will see the returned data in the output pane of the test.



Method: POST
Endpoint: http://localhost:7090
Resource: /empServClient_getEmployee
Parameters:

Required: Sets if parameter is requir
Type:

Media Type: application/json Post

```
{"empNumber":"000010"}
```

Authorization: Basic
Username: iibuser1
Password: ••••••••
Domain:
Pre-emptive auth: (●) Use global preferen ( ) Authenticate pre-en

```
{"EmployeeResponse": {
   "DBResp":    {
      "UserReturnCode": 0,
      "RowsRetrieved": 1,
      "RowsAdded": 0,
      "RowsUpdated": 0,
      "RowsDeleted": 0,
      "SQLCode_ErrorCode": 0,
      "SQLState_SQLState": "",
      "SQL_Error_Message": ""
   },
   "EMPLOYEE":    {
      "EMPNO": "000010",
      "FIRSTNME": "CHRISTINE",
      "MIDINIT": "J",
      "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
   }
}}
```

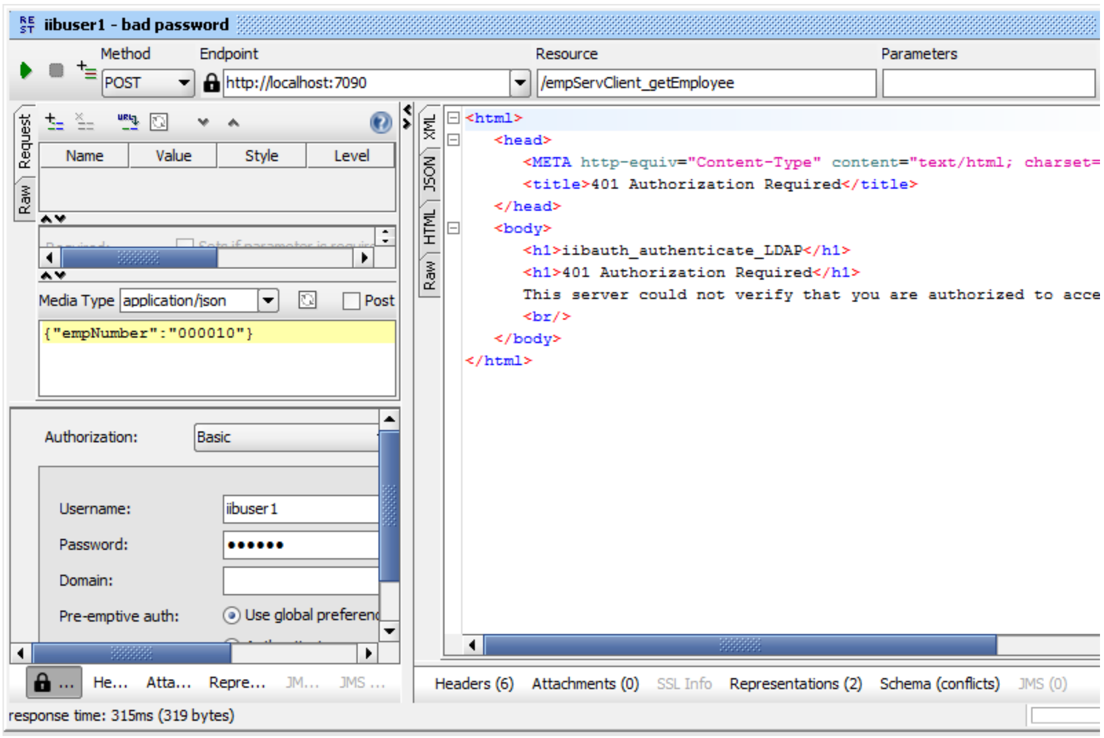
response time: 16306ms (433 bytes)

Repeat the test with the users iibuser2 and iibuser3. These should also be authenticated correctly.

5. Open the request "iibuser1 - bad password", and run the test. This should fail, and a 401 response will be seen in the SOAPUI response.

   Note that the message is a generic security message. Even though the message is an authorisation message, this particular error occurs because of an authentication failure.

   Note that the name of the IIB Security Profile, iibauth_authenticate_LDAP, is shown in the body of the response.



Method: POST
Endpoint: http://localhost:7090
Resource: /empServClient_getEmployee

Media Type: application/json

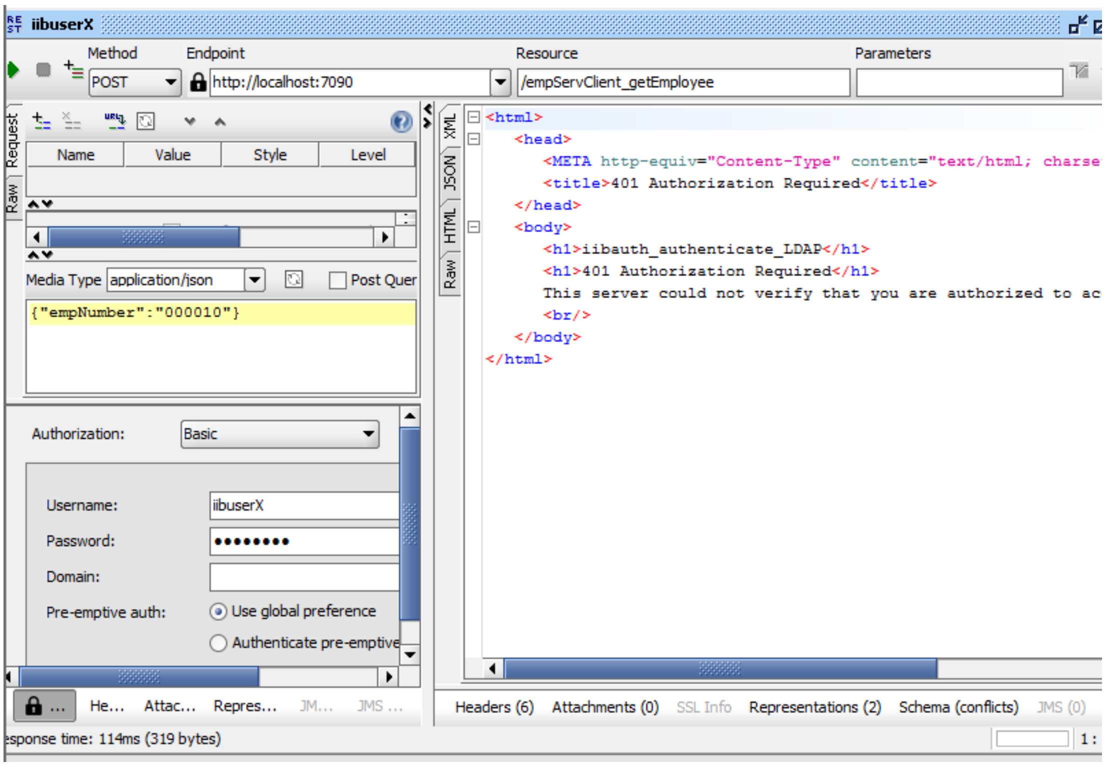
```
{"empNumber":"000010"}
```

Authorization: Basic
Username: iibuser1
Password: ••••••
Domain:
Pre-emptive auth: (•) Use global preferenc

```
<html>
  <head>
    <META http-equiv="Content-Type" content="text/html; charset=
    <title>401 Authorization Required</title>
  </head>
  <body>
    <h1>iibauth_authenticate_LDAP</h1>
    <h1>401 Authorization Required</h1>
    This server could not verify that you are authorized to acce
    <br/>
  </body>
</html>
```

response time: 315ms (319 bytes)

6. Open the request "**iibuserX**", and run the test. This user does not exist in the LDAP database, so should again fail.



```
{"empNumber":"000010"}
```

```
<html>
   <head>
      <META http-equiv="Content-Type" content="text/html; charse
      <title>401 Authorization Required</title>
   </head>
   <body>
      <h1>iibauth_authenticate_LDAP</h1>
      <h1>401 Authorization Required</h1>
      This server could not verify that you are authorized to ac
      <br/>
   </body>
</html>
```

7. Other reasons for getting a 401 authorisation failure:

The IIB node will return a 401 failure for many reasons. During development of this lab, the following scenarios resulted in a security failure, so these should be on your checklist when doing this type of security checking:

- LDAP (SDS) server not available - not started
- LDAP server (SDS) started in Config-only mode
- IIB node LDAP username invalid or not set (cn=root in this scenario)
- IIB node LDAP username password wrong or not set (cn=root / passw0rd in this scenario)
- Username / password of message flow request not specified properly

6. Configure and test authorization with LDAP

We will now move on to add authorization to the security policy for these message flows.

The following users are defined in the LDAP and are members of the authorisation groups as shown::

LDAP Group -->	iibauth_default	iibauth_update	iibauth_delete	None
	iibuser1	iibuser2	iibuser3	iibuser4
	iibuser2	iibuser3		
	iibuser3			

6.1 Import further security profiles

So far, you have only defined one security profile, iibauth_authenticate_LDAP, and this was configured just for authentication.

For the rest of the lab, we have provided a set of security profile configurable services for you. These will be used for both authentication and authorization, in different configurations.

You will import these into the IIB IB10NODE_MFS_P configuration.

Security profiles for plain text LDAP connection

• Message flow EmpServ_JSON_getEmployee	Profile = iibauth_default
• Message flow EmpServ_JSON_updEmployee	Profile = iibauth_update
• Message flow EmpServ_JSON_dltEmployee	Profile = iibauth_delete

Security profiles for secure LDAP connection (used in the next chapter of the lab)

• Message flow EmpServ_JSON_getEmployee	Profile = iibauth_default_s
• Message flow EmpServ_JSON_updEmployee	Profile = iibauth_update_s
• Message flow EmpServ_JSON_dltEmployee	Profile = iibauth_delete_s

Security profiles for authorization only (used in chapter 7 - X.509 certificates)

• Message flow EmpServ_JSON_getEmployee	Profile = iibauth_authorizationonly_default
• Message flow EmpServ_JSON_updEmployee	Profile = iibauth_authorizationonly_update
• Message flow EmpServ_JSON_dltEmployee	Profile = iibauth_authorizationonly_delete

1. In an IIB Command Console, change directory to

   `c:\student10\Integration_service_MessageFlowSecurity\config_services`

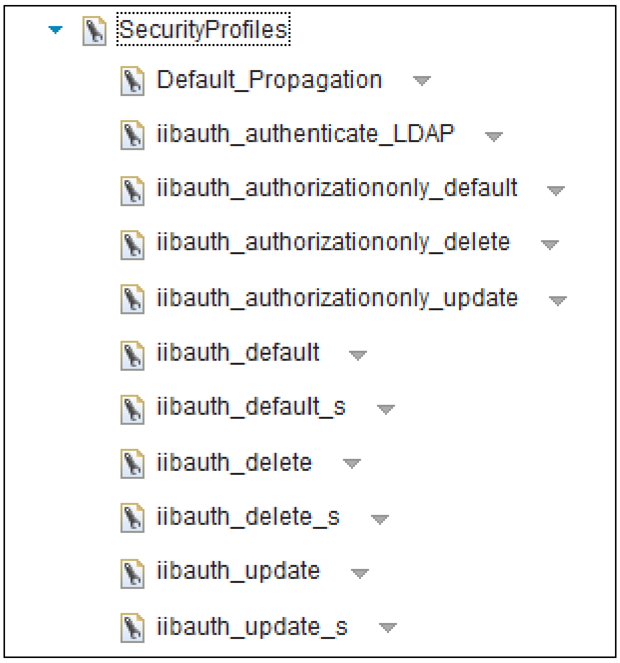
   Run the command

   `create_iibauth_config_services`

   Accept the default values for the node.

2. When the nine configurable services have been imported, switch to the web admin browser for IB10NODE_MFS_P.

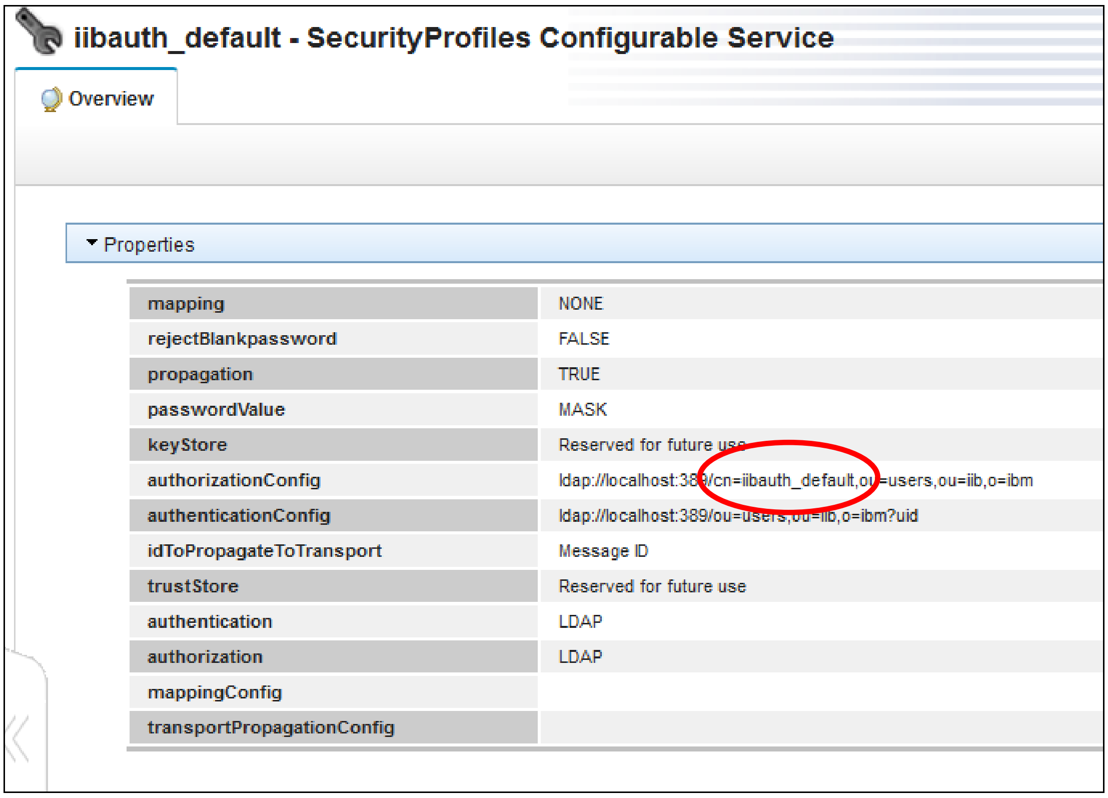
   Expand Operational Policy, Configurable Services, Security Profiles. You should see the following security profiles:



SecurityProfiles
- Default_Propagation
- iibauth_authenticate_LDAP
- iibauth_authorizationonly_default
- iibauth_authorizationonly_delete
- iibauth_authorizationonly_update
- iibauth_default
- iibauth_default_s
- iibauth_delete
- iibauth_delete_s
- iibauth_update
- iibauth_update_s

3. As an example, take a look at the **iibauth_default** profile. Note that both the LDAP authentication and authorisation configs have been provided.

   The authorizationConfig for the profile below means that a user who attempts to access a message flow which has this security profile must be in the group "iibauth_default" in the LDAP database.



iibauth_default - SecurityProfiles Configurable Service

Overview

Properties

mapping	NONE
rejectBlankpassword	FALSE
propagation	TRUE
passwordValue	MASK
keyStore	Reserved for future use
authorizationConfig	ldap://localhost:389/cn=iibauth_default,ou=users,ou=iib,o=ibm
authenticationConfig	ldap://localhost:389/ou=users,ou=iib,o=ibm?uid
idToPropagateToTransport	Message ID
trustStore	Reserved for future use
authentication	LDAP
authorization	LDAP
mappingConfig	
transportPropagationConfig	

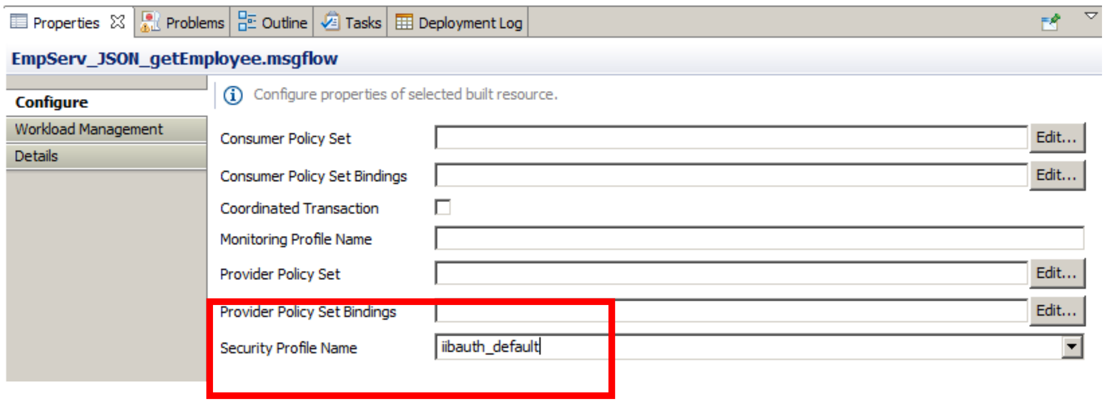
4. Now you will associate the message flows with the appropriate security profiles.

   In the Toolkit, open the EmployeeService_JSONClient.MFS.10.0.0.3.bar in the barfile editor.

   In the editor, on the Manage tab, expand the EmployeeService_JSONClient application.

   Select the EmpServ_JSON_getEmployee.msgflow.

   Change the Security Profile Name to iibauth_default.

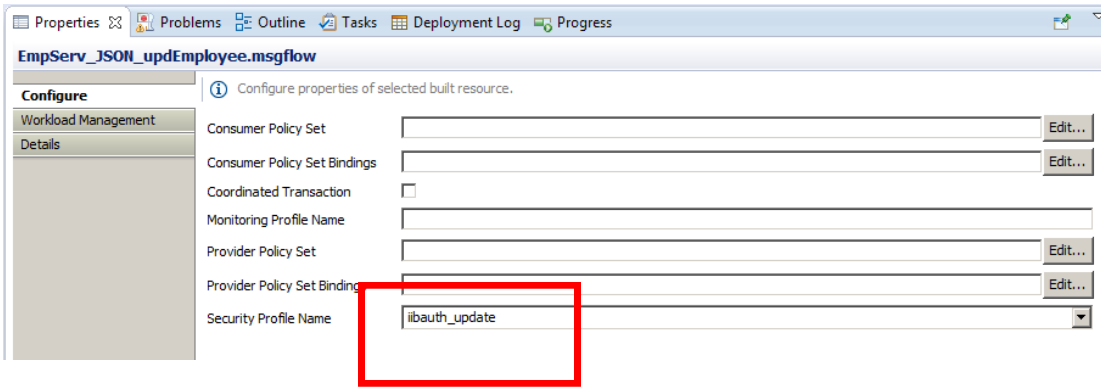


EmpServ_JSON_getEmployee.msgflow

Configure properties of selected built resource.

Consumer Policy Set	
Consumer Policy Set Bindings	
Coordinated Transaction	☐
Monitoring Profile Name	
Provider Policy Set	
Provider Policy Set Bindings	
Security Profile Name	iibauth_default

5. For the updEmployee message flow, set the profile name to "iibauth_update".


EmpServ_JSON_updEmployee.msgflow

Configure properties of selected built resource.

Consumer Policy Set	
Consumer Policy Set Bindings	
Coordinated Transaction	☐
Monitoring Profile Name	
Provider Policy Set	
Provider Policy Set Bindings	
Security Profile Name	iibauth_update

6. For the dltEmployee message flow, set profile name to "iibauth_delete".

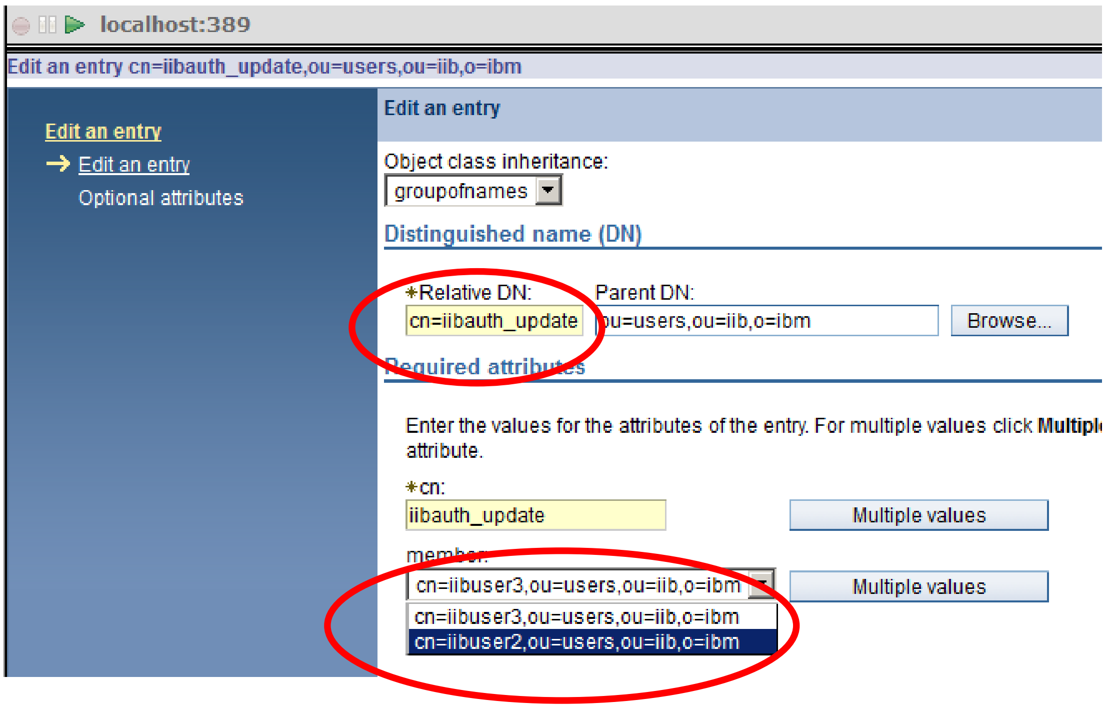
7. Save and redeploy the barfile to IB10NODE_MFS_P / PROVIDER.

6.2 Test authorisation with LDAP

1. As an example of the LDAP authorisation groups, this screen capture from the IBM Security Directory Server Administration tool shows the group **iibauth_update** group (object name = GroupOfNames).

   Note that the users iibuser2 and iibuser3 are members of this group, but iibuser1 is not.

It's not recommended that you login to SDS in this lab. If you want to do this, you will have to start the WAS component, to login through the web browser.



localhost:389

Edit an entry cn=iibauth_update,ou=users,ou=iib,o=ibm

Edit an entry

Object class inheritance: groupofnames

Distinguished name (DN)

*Relative DN: cn=iibauth_update

Parent DN: ou=users,ou=iib,o=ibm

Required attributes

Enter the values for the attributes of the entry. For multiple values click Multipl attribute.

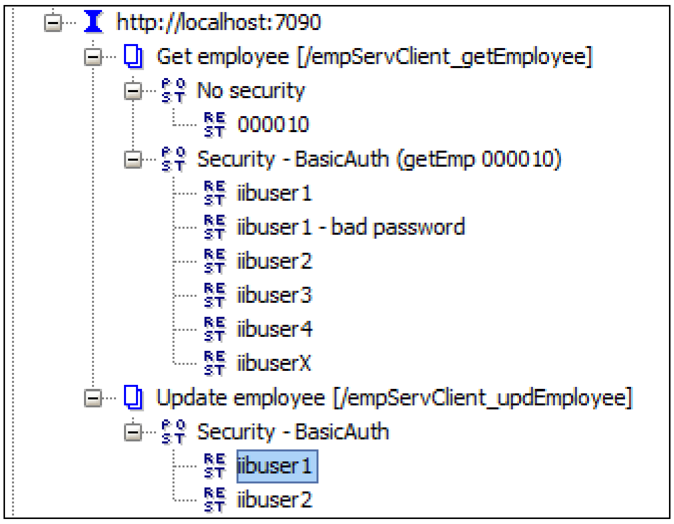
*cn: iibauth_update

member: cn=iibuser3,ou=users,ou=iib,o=ibm

cn=iibuser3,ou=users,ou=iib,o=ibm
cn=iibuser2,ou=users,ou=iib,o=ibm

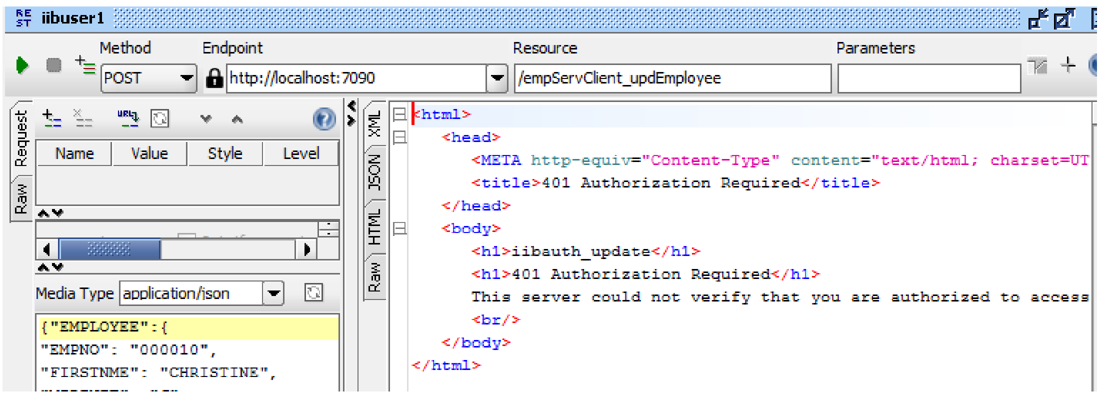
2. In SOAPUI, in the same project as earlier, expand the "Update employee" resource, and "Security - Basic Auth".

   Open the **iibuser1** request.



```
http://localhost:7090
  Get employee [/empServClient_getEmployee]
    No security
      000010
    Security - BasicAuth (getEmp 000010)
      iibuser1
      iibuser1 - bad password
      iibuser2
      iibuser3
      iibuser4
      iibuserX
  Update employee [/empServClient_updEmployee]
    Security - BasicAuth
      iibuser1
      iibuser2
```

3. iibuser1 is not a member of the iibauth_update group, so should not be authorised to run this message flow. IIB will return a 401 Authorisation Failure message, as shown below.



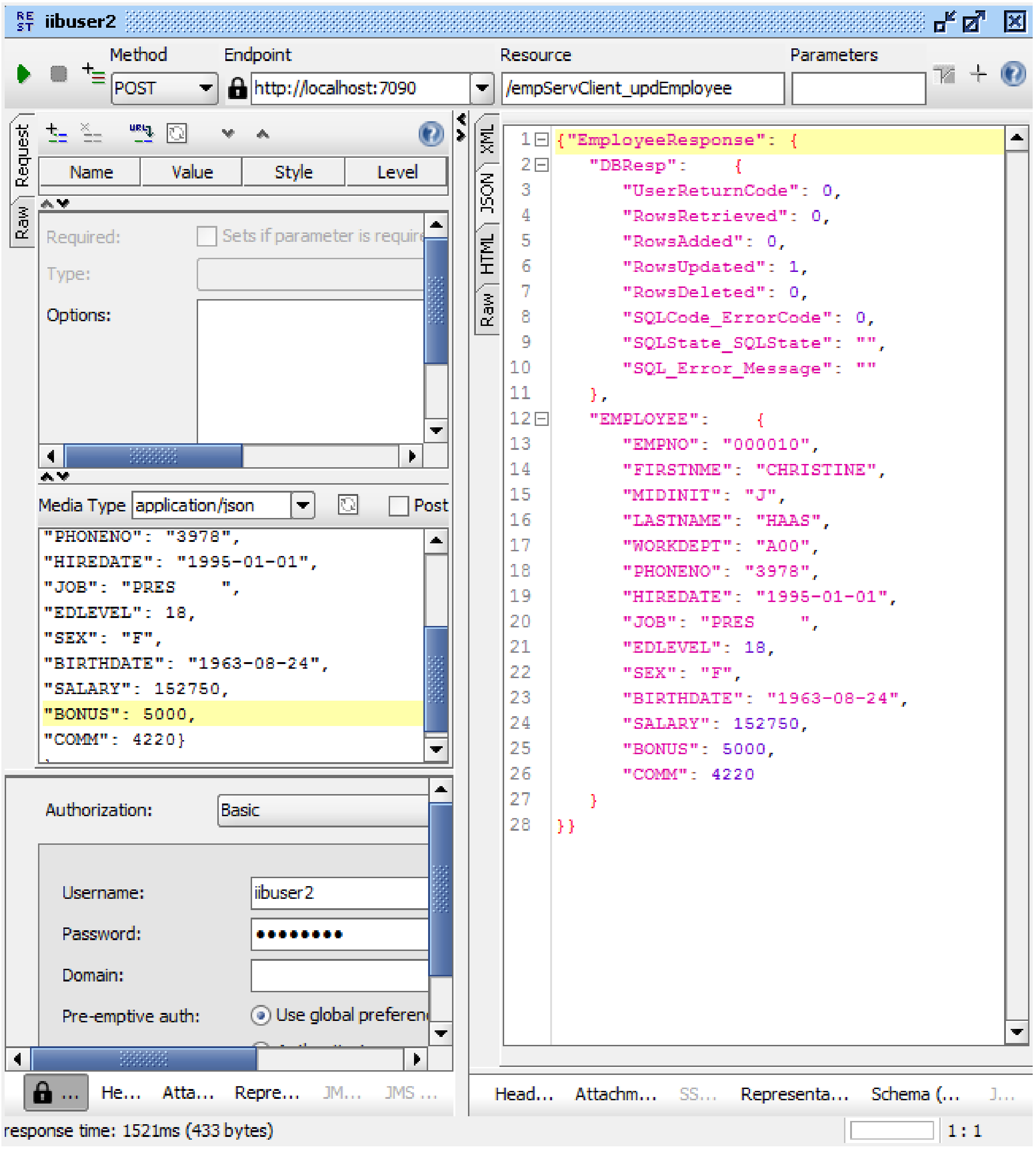
Method: POST
Endpoint: http://localhost:7090
Resource: /empServClient_updEmployee
Media Type: application/json

```
{"EMPLOYEE":{
"EMPNO": "000010",
"FIRSTNME": "CHRISTINE",
```

```
<html>
  <head>
    <META http-equiv="Content-Type" content="text/html; charset=UT
    <title>401 Authorization Required</title>
  </head>
  <body>
    <h1>iibauth_update</h1>
    <h1>401 Authorization Required</h1>
    This server could not verify that you are authorized to access
    <br/>
  </body>
</html>
```

4. iibuser2 is in the iibauth_update group, so is permitted to execute this flow.

Click the JSON tab to see the full returned data.
The BONUS value will have been updated to 5000 (hardcoded in the input data).



Method: POST
Endpoint: http://localhost:7090
Resource: /empServClient_updEmployee
Media Type: application/json

```
"PHONENO": "3978",
"HIREDATE": "1995-01-01",
"JOB": "PRES     ",
"EDLEVEL": 18,
"SEX": "F",
"BIRTHDATE": "1963-08-24",
"SALARY": 152750,
"BONUS": 5000,
"COMM": 4220}
```

Authorization: Basic
Username: iibuser2
Password: ••••••••
Domain:
Pre-emptive auth: Use global preferen

```
{"EmployeeResponse": {
   "DBResp":     {
      "UserReturnCode": 0,
      "RowsRetrieved": 0,
      "RowsAdded": 0,
      "RowsUpdated": 1,
      "RowsDeleted": 0,
      "SQLCode_ErrorCode": 0,
      "SQLState_SQLState": "",
      "SQL_Error_Message": ""
   },
   "EMPLOYEE":     {
      "EMPNO": "000010",
      "FIRSTNME": "CHRISTINE",
      "MIDINIT": "J",
      "LASTNAME": "HAAS",
      "WORKDEPT": "A00",
      "PHONENO": "3978",
      "HIREDATE": "1995-01-01",
      "JOB": "PRES     ",
      "EDLEVEL": 18,
      "SEX": "F",
      "BIRTHDATE": "1963-08-24",
      "SALARY": 152750,
      "BONUS": 5000,
      "COMM": 4220
   }
}}
```

response time: 1521ms (433 bytes)

7. Securing the LDAP Connection with SSL

In the previous chapter, the username and password credentials of the user were passed to the LDAP server for authentication in clear text. This means that username and password would be exposed to any network sniffing technology, such as Wireshark.

This section adds protection of the connection between the IIB node and the LDAP server. It does this by using new security profiles which use a "ldaps" connection.

7.1 Review LDAPS Security Profiles

Three of the new security profiles that were imported earlier have been defined to use a secure connection to the LDAP server:

- iibauth_default_s
- iibauth_update_s
- iibauth_delete_s

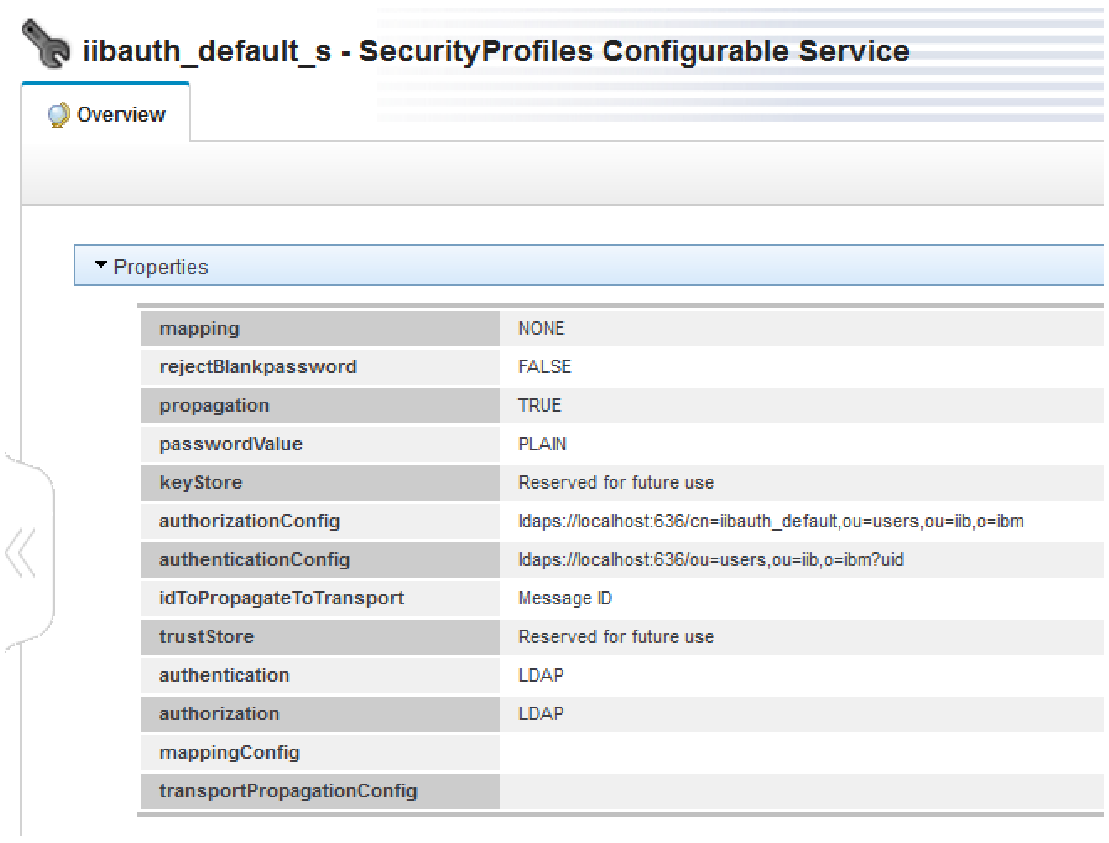
In the IIB web admin tool for IB10NODE_MFS_P (port 4432), open one of the security profiles with the "_s" suffix.

For example, the iibauth_default_s profile will look like this.

Note that the authentication and authorization Configs connect to the LDAP server using a url of the form

```
ldaps:..localhost:636/....
```

Port 636 is the default port for SSL connections to an LDAP server.



mapping	NONE
rejectBlankpassword	FALSE
propagation	TRUE
passwordValue	PLAIN
keyStore	Reserved for future use
authorizationConfig	ldaps://localhost:636/cn=iibauth_default,ou=users,ou=iib,o=ibm
authenticationConfig	ldaps://localhost:636/ou=users,ou=iib,o=ibm?uid
idToPropagateToTransport	Message ID
trustStore	Reserved for future use
authentication	LDAP
authorization	LDAP
mappingConfig	
transportPropagationConfig	

7.2 Define PKI for IIB connection to LDAP server

Before these security profiles can be used, you must configure the IIB node to use an appropriate PKI configuration that enables an SSL connection to the LDAP server.

For ease of configuration, we have provided prebuilt keystores and truststores for both IIB and SDS(LDAP). These contain Personal and Signer certificates on both sides to enable mutual authentication of IIB with SDS.

For reference, the following table shows the certificates that have been provided in the various keystores and truststores. Note that these tables include the certificates that are used in the final section of this lab, namely the PKI for the CONSUMER/PROVIDER connection.

We have provided a full definition of user certificates, issued by the "rootca" certificate authority. "rootca" is a certificate authority defined locally, and enables certificates to be located with a full certificate path.

	Personal certificates	**Signer certificates**
PROVIDER keystore (type JKS) ProviderKeyStore.jks	providercert	rootca
PROVIDER truststore (type JKS) ProviderTrustStore.jks	-	consumercert iibuser1cert iibuser2cert iibuser3cert iibuser4cert sdscert rootca
SDS keystore/truststore (type CMS) SDS.kdb	sdscert	providercert rootCA
CONSUMER keystore (type JKS) ConsumerKeystore.jks	consumercert iibuser1cert iibuser2cert iibuser3cert iibuser4cert	providercert rootca
CONSUMER truststore (type JKS) ConsumerTruststore.jks	-	providercert

You can use ikeyman to take a further look at the key/truststores and the certificates they contain. For example, here is a screen capture of the ProviderTrustStore.jks database.

ikeyman is available on the Windows Start menu.

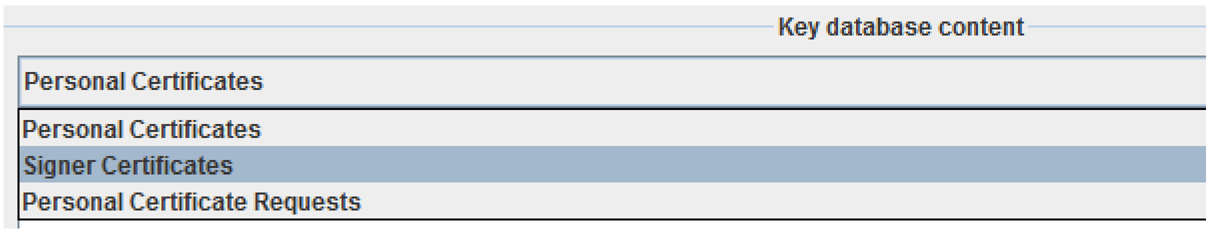
To view this, using ikeyman:

- Click "Key Database File", then Open.
- Set the "Key database type" to **JKS** (the default is CMS, which will cause an error)
- Specify the FileName :

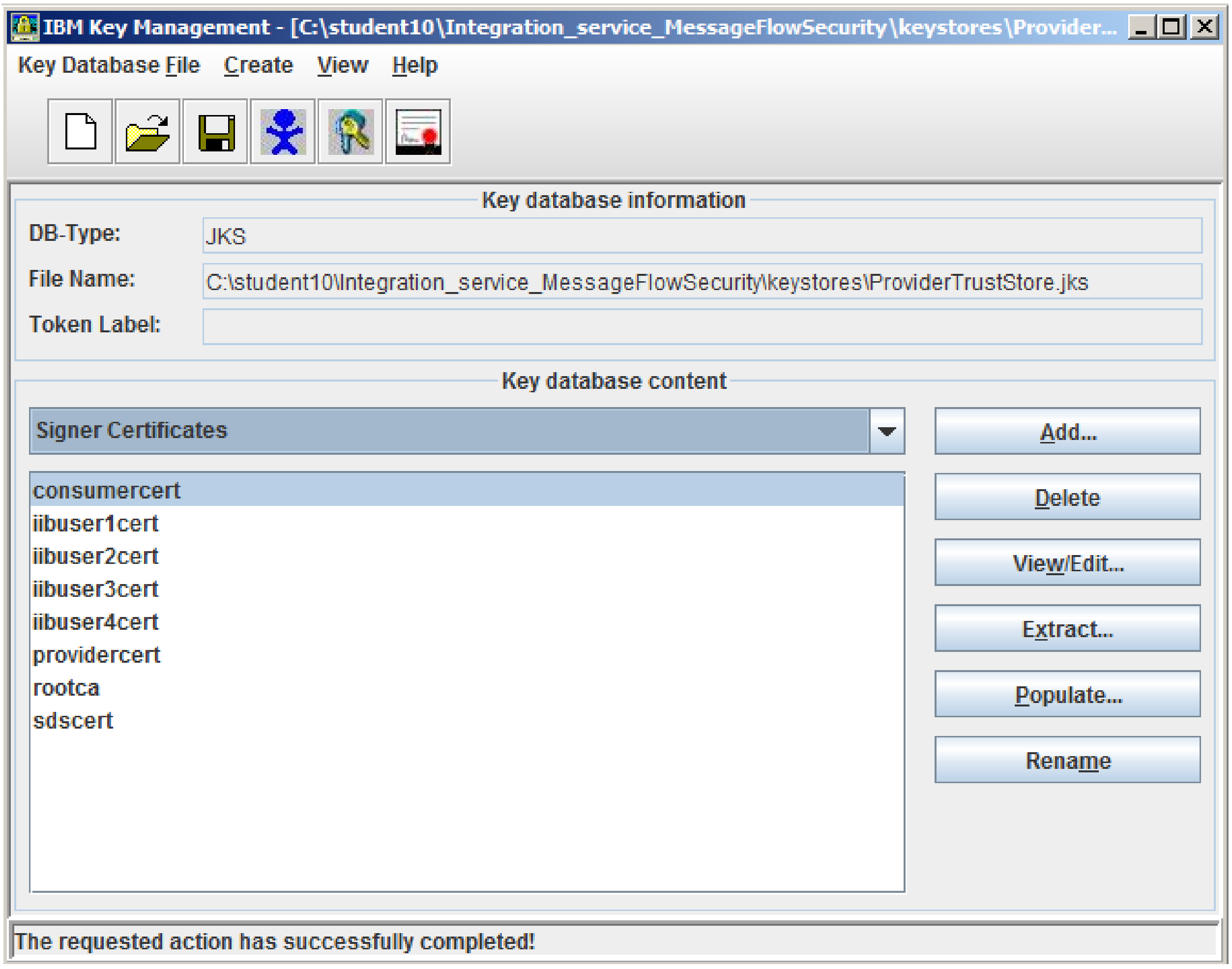
  c:\ student10 \ integration_service_MessageFlowSecurity \ keystores \ ProviderTrustStore.jks

- When prompted, the password is "**passw0rd**"

The default display for ikeyman is to show the Personal Certificates. This particular database does not have any personal certificates, so click on this and select Signer Certificates. These are the certificates that are used by the provider system to validate incoming requests.



The Signer certificates are now shown as follows:


IBM Key Management - [C:\student10\Integration_service_MessageFlowSecurity\keystores\Provider...

Key database information

DB-Type: JKS

File Name: C:\student10\Integration_service_MessageFlowSecurity\keystores\ProviderTrustStore.jks

Token Label:

Key database content

Signer Certificates

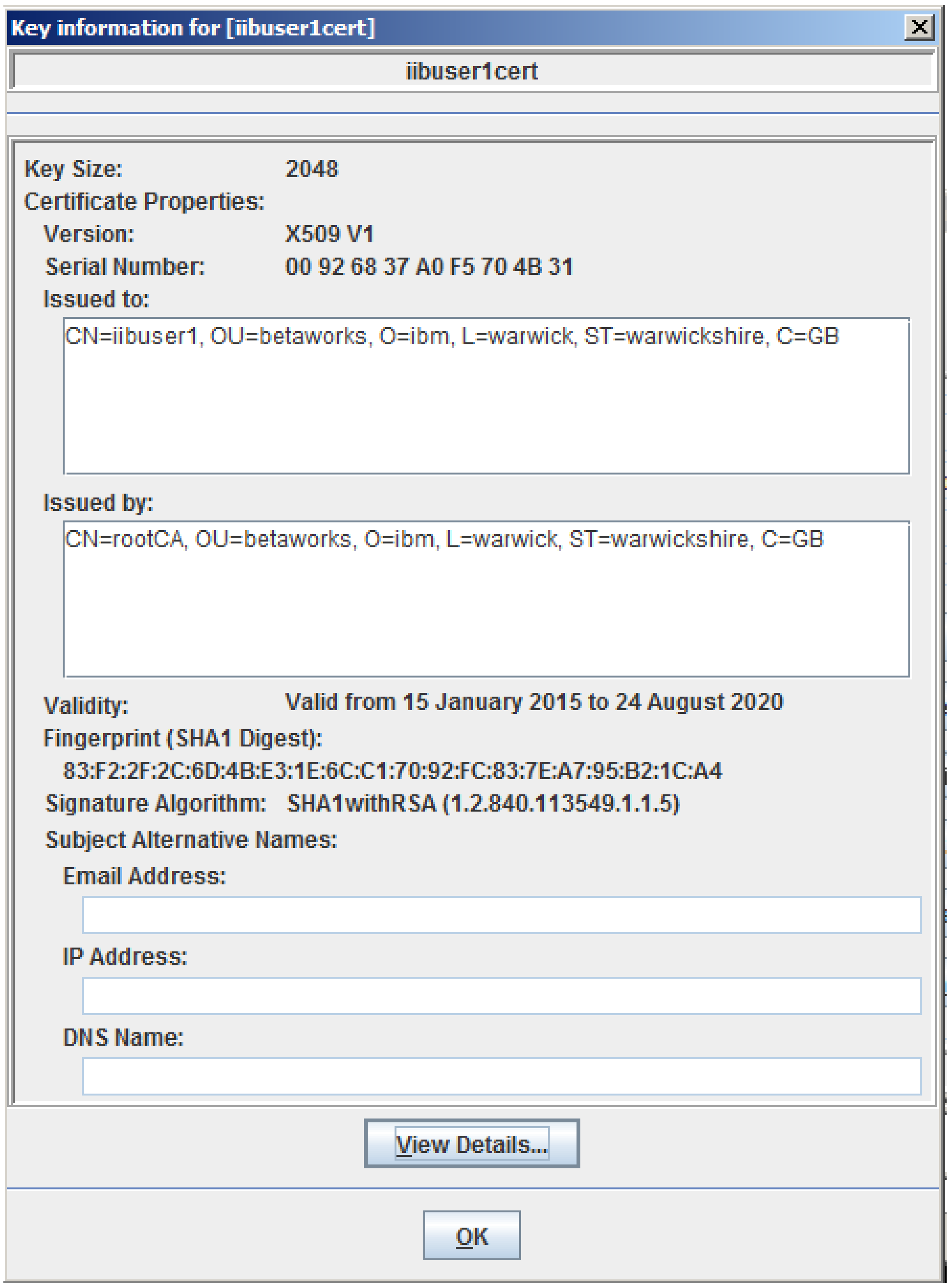
consumercert
iibuser1cert
iibuser2cert
iibuser3cert
iibuser4cert
providercert
rootca
sdscert

The requested action has successfully completed!

Select one of these certificates and click View/Edit. This will show the key information for this user.

Note that both the "Issued to" and "Issued by" have a full rootCA path.

Click OK to close.



Key Size:	2048
Certificate Properties:	
Version:	X509 V1
Serial Number:	00 92 68 37 A0 F5 70 4B 31
Issued to:	CN=iibuser1, OU=betaworks, O=ibm, L=warwick, ST=warwickshire, C=GB
Issued by:	CN=rootCA, OU=betaworks, O=ibm, L=warwick, ST=warwickshire, C=GB
Validity:	Valid from 15 January 2015 to 24 August 2020
Fingerprint (SHA1 Digest):	83:F2:2F:2C:6D:4B:E3:1E:6C:C1:70:92:FC:83:7E:A7:95:B2:1C:A4
Signature Algorithm:	SHA1withRSA (1.2.840.113549.1.1.5)
Subject Alternative Names:	
Email Address:	
IP Address:	
DNS Name:	

1. When Security Profiles use an LDAPS connection, you have to provide an appropriate set of keystores and truststores for the IIB server, and for the Security Directory Server.

   A script has been provided to do this for you. In an IIB Command Console, switch to the folder:

   ```
       C:\student10\Integration_service_MessageFlowSecurity\commands
   ```

   Run the command **`SetupPKI_forEmbeddedListeners`**. Accept the default values for the node name and server name.

   This will create the PKI for the connection to the LDAP server. At the same time, since an IIB server can only have one keystore and one truststore, it will also define the PKI for the X.509 scenarios that will be described chapter 7.

   This will run the following commands, or you can run them manually yourself (case is important).

   Commands for IB10NODE_MFS_P

   Define location of keystore

   ```
         mqsichangeproperties IB10NODE_MFS_P
               -e PROVIDER
               -o ComIbmJVMManager
               -n keystoreFile
               -v c:\student10\Integration_service_MessageFlowSecurity
               \keystores\ProviderKeyStore.jks
   ```

   Define location of truststore

   ```
         mqsichangeproperties IB10NODE_MFS_P
               -e PROVIDER
               -o ComIbmJVMManager
               -n truststoreFile
               -v c:\student10\Integration_service_MessageFlowSecurity
               \keystores\ProviderTrustStore.jks
   ```

   Specify password of keystore

   ```
         mqsisetdbparms IB10NODE_MFS_P
               -n brokerKeystore::password
               -u ignore
               -p passw0rd
   ```

   Specify password of truststore

   ```
         mqsisetdbparms IB10NODE_MFS_P
               -n brokerTruststore::password
               -u ignore
               -p passw0rd
   ```

   Stop and restart

   ```
         mqsistop IB10NODE_MFS_P
         mqsistart IB10NODE_MFS_P
   ```

   Review settings

   ```
         mqsireportproperties IB10NODE_MFS_P
               -o ComIbmJVMManager
               -a
               -e PROVIDER
   ```

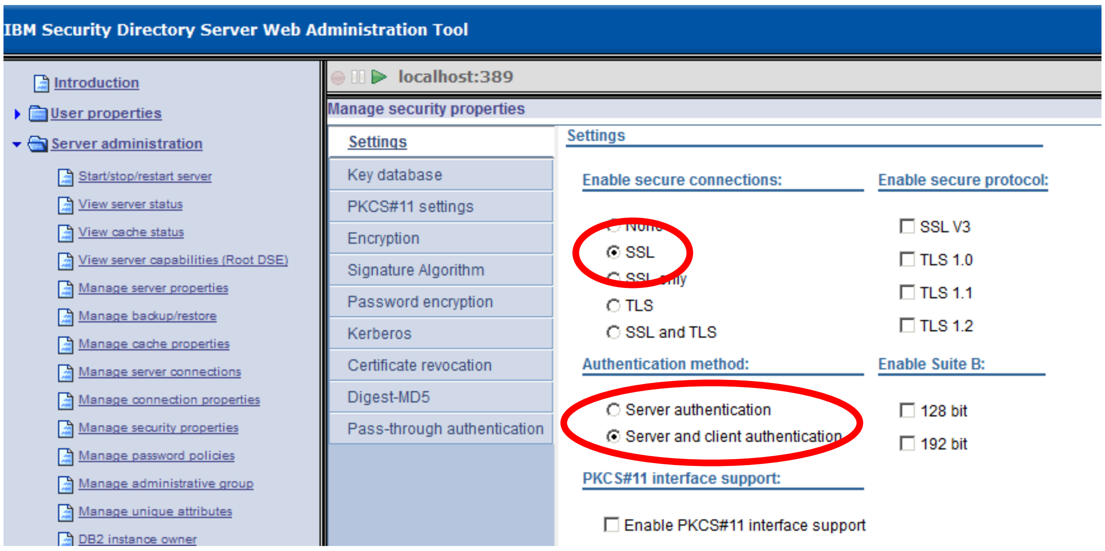
   The corresponding commands need to be run for IB10NODE_MFS_C / CONSUMER (for the X/509 scenario). The script will have done this for you.

2. The equivalent configuration in the LDAP server (SDS) has been done for you. For reference, here are two screen captures of the Security Settings tab.

It's not recommended that you login to SDS in this lab. More detail is provided in the detailed SDS set up lab guide - please ask if you need this.

In the "Server administration" section, you can select "Manage security properties".

On the Settings tab, SSL has been selected, and "Server and client authentication".


IBM Security Directory Server Web Administration Tool

Introduction
User properties
Server administration
Start/stop/restart server
View server status
View cache status
View server capabilities (Root DSE)
Manage server properties
Manage backup/restore
Manage cache properties
Manage server connections
Manage connection properties
Manage security properties
Manage password policies
Manage administrative group
Manage unique attributes
DB2 instance owner

localhost:389

Manage security properties

Settings
Key database
PKCS#11 settings
Encryption
Signature Algorithm
Password encryption
Kerberos
Certificate revocation
Digest-MD5
Pass-through authentication

Settings

Enable secure connections:
None
[✓ SSL]
SSL only
TLS
SSL and TLS

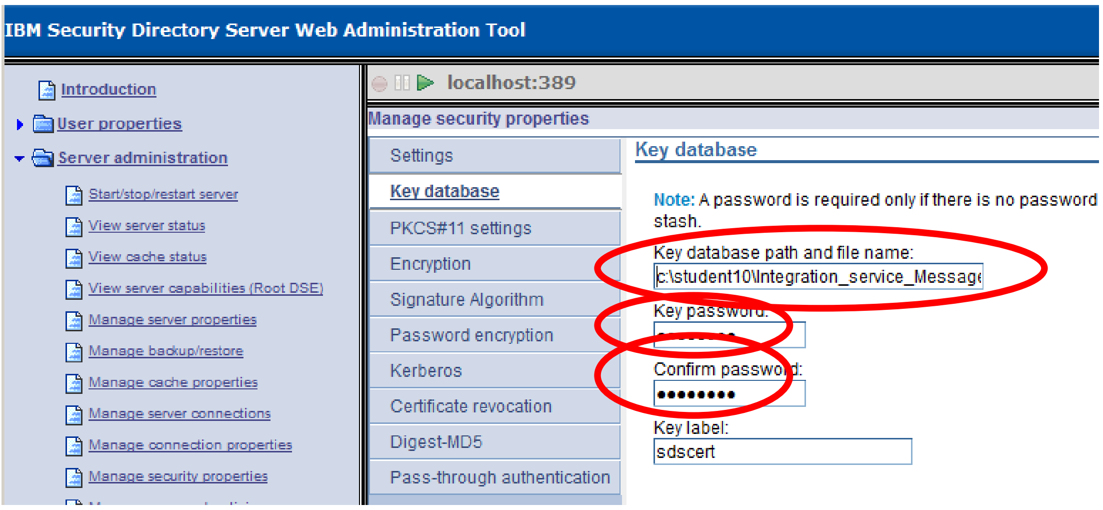
Enable secure protocol:
☐ SSL V3
☐ TLS 1.0
☐ TLS 1.1
☐ TLS 1.2

Authentication method:
Server authentication
[✓ Server and client authentication]

Enable Suite B:
☐ 128 bit
☐ 192 bit

PKCS#11 interface support:
☐ Enable PKCS#11 interface support

And on the "Key database" tab, the name of the matching key and truststore file is specified, along with the required password and the "Key label" (sdscert), which has to match the name of the SDS certificate in the SDS keystore.
Note that the screen capture does not show the full length of the keystore file name.


IBM Security Directory Server Web Administration Tool

Introduction
User properties
Server administration
Start/stop/restart server
View server status
View cache status
View server capabilities (Root DSE)
Manage server properties
Manage backup/restore
Manage cache properties
Manage server connections
Manage connection properties
Manage security properties

localhost:389

Manage security properties

Settings
Key database
PKCS#11 settings
Encryption
Signature Algorithm
Password encryption
Kerberos
Certificate revocation
Digest-MD5
Pass-through authentication

Key database

Note: A password is required only if there is no password stash.

Key database path and file name:
c:\student10\Integration_service_Message

Key password:
••••••••

Confirm password:
••••••••

Key label:
sdscert

Tip: if you are performing similar configuration with SDS, and the keystore file does not exist in the specified folder, or if "Key label" cannot be found in the keystore, the SDS configuration will start in "Config only" mode. IIB will not be able to use the LDAP for authenticating or authorising users.

To resolve, either provide the keystore file in the expected location, make sure the key label is named correctly, or disable the SSL connection in SDS.

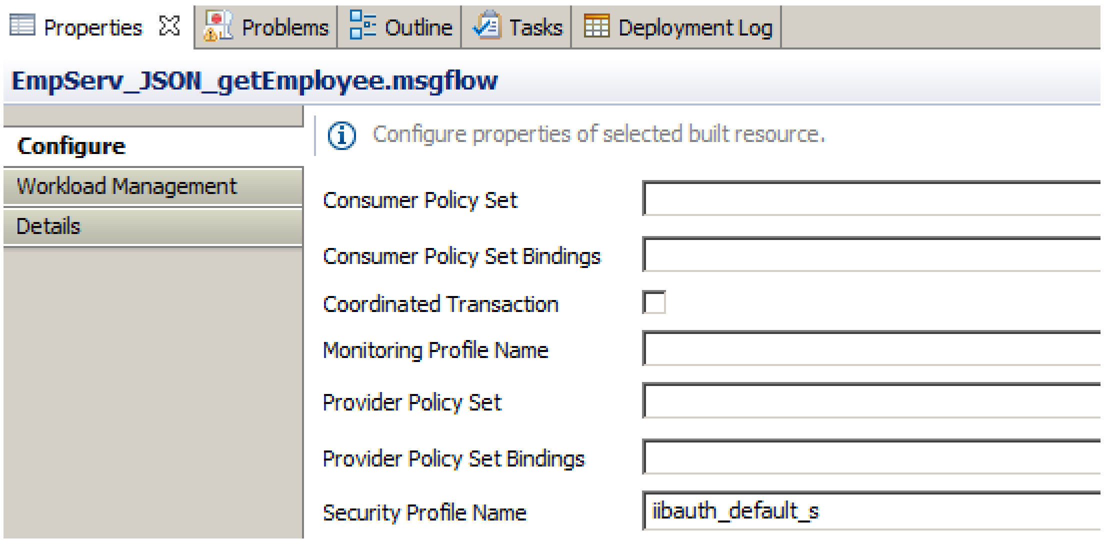
7.3 Update JSON Client to use Security Profiles LDAPS

You have now defined all of the items that are needed to use a secure connection to the LDAP server.

The final task is to configure the JSON Client message flows to reference the new security profiles. This could be done by using barfile overrides and automated processes to redeploy. However, this lab will use the barfile editor to make these changes.

1. In the Toolkit, edit the EmployeeService_JSONClient.MFS.10.0.0.3.bar again.

   For the EmpServ_JSON_getEmployee message flow, set the Security Profile Name to **iibauth_default_s**.

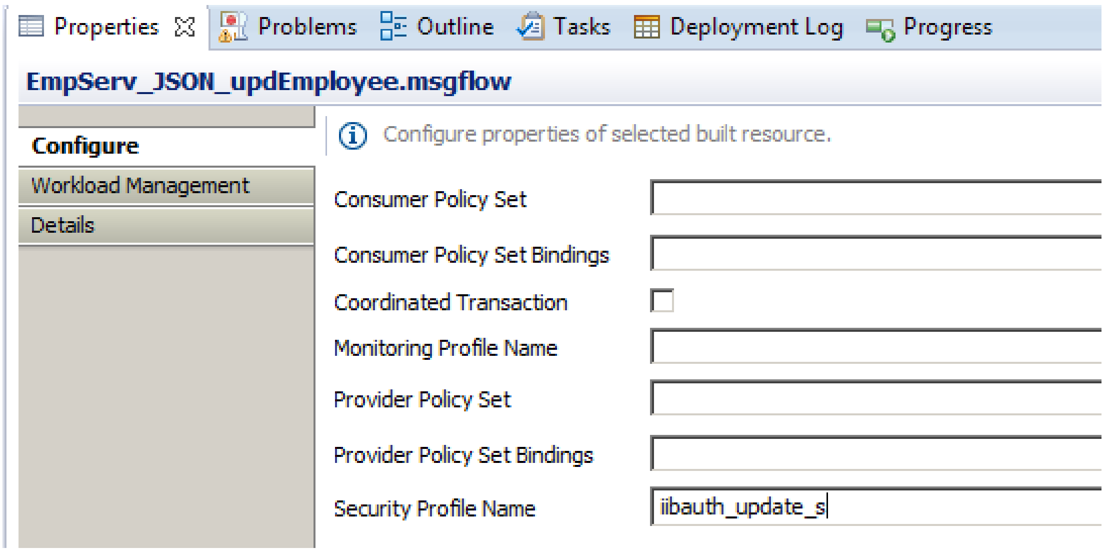


EmpServ_JSON_getEmployee.msgflow

Configure properties of selected built resource.

Consumer Policy Set	
Consumer Policy Set Bindings	
Coordinated Transaction	☐
Monitoring Profile Name	
Provider Policy Set	
Provider Policy Set Bindings	
Security Profile Name	iibauth_default_s

2. Similarly, for the updEmployee flow, set the Profile name **to iibauth_update_s**.



EmpServ_JSON_updEmployee.msgflow

Configure properties of selected built resource.

Consumer Policy Set	
Consumer Policy Set Bindings	
Coordinated Transaction	☐
Monitoring Profile Name	
Provider Policy Set	
Provider Policy Set Bindings	
Security Profile Name	iibauth_update_s

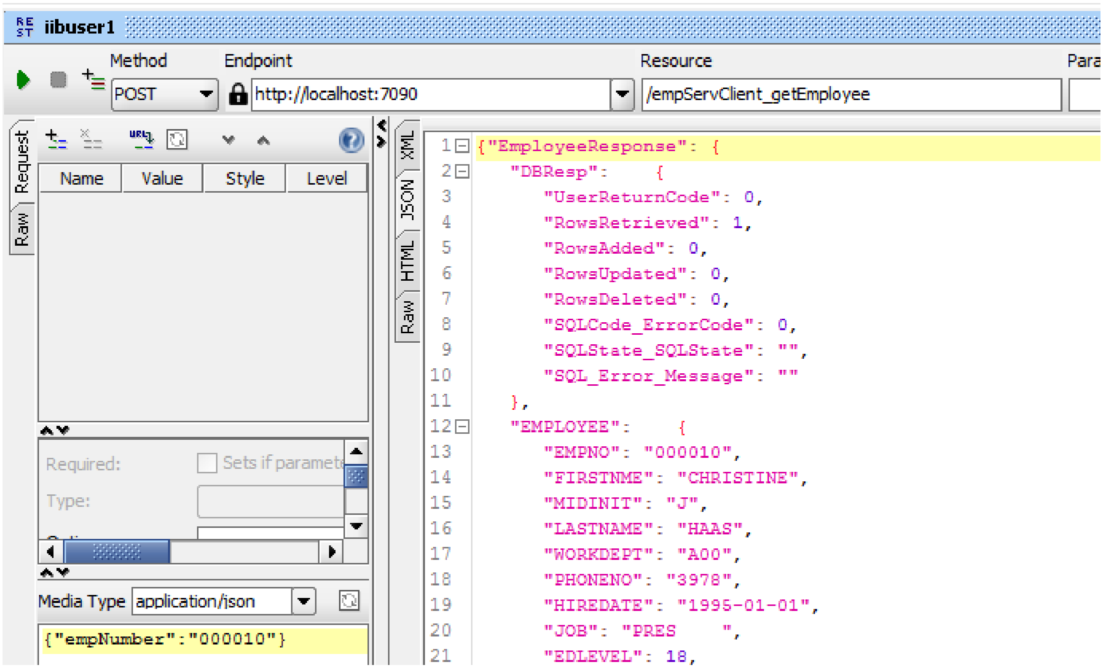
3. Save and redeploy the barfile.

7.4 Test Authentication using the secure LDAP connection

Repeat the tests for basicAuth (previously run in section 4.3).

1.
    - Host - localhost:7090
    - Function - Get Employee
    - Resource - Security - Basic Auth
    - Request = iibuser1

The request will succeed.



POST http://localhost:7090 /empServClient_getEmployee

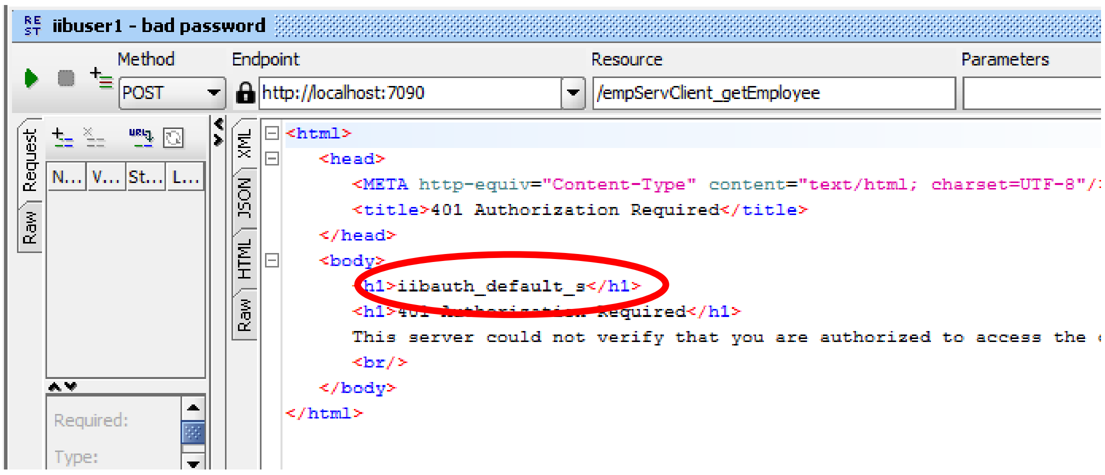
Media Type application/json

```
{"empNumber":"000010"}
```

```
{"EmployeeResponse": {
   "DBResp":     {
      "UserReturnCode": 0,
      "RowsRetrieved": 1,
      "RowsAdded": 0,
      "RowsUpdated": 0,
      "RowsDeleted": 0,
      "SQLCode_ErrorCode": 0,
      "SQLState_SQLState": "",
      "SQL_Error_Message": ""
   },
   "EMPLOYEE":     {
      "EMPNO": "000010",
      "FIRSTNME": "CHRISTINE",
      "MIDINIT": "J",
      "LASTNAME": "HAAS",
      "WORKDEPT": "A00",
      "PHONENO": "3978",
      "HIREDATE": "1995-01-01",
      "JOB": "PRES     ",
      "EDLEVEL": 18,
```

2. Request = iibuser1 - bad password

Note that in the case of security failures, a 401 message will be shown as before. The name of the associated Security Profile will also be shown. In this case, you will see that the profile will have the suffix "_s", indicating that the connection to SDS was indeed a secure one.



POST http://localhost:7090 /empServClient_getEmployee

```
<html>
   <head>
      <META http-equiv="Content-Type" content="text/html; charset=UTF-8"/
      <title>401 Authorization Required</title>
   </head>
   <body>
      <h1>iibauth_default_s</h1>
      <h1>401 Authorization Required</h1>
      This server could not verify that you are authorized to access the
      <br/>
   </body>
</html>
```

8. Client security with X.509 Certificates

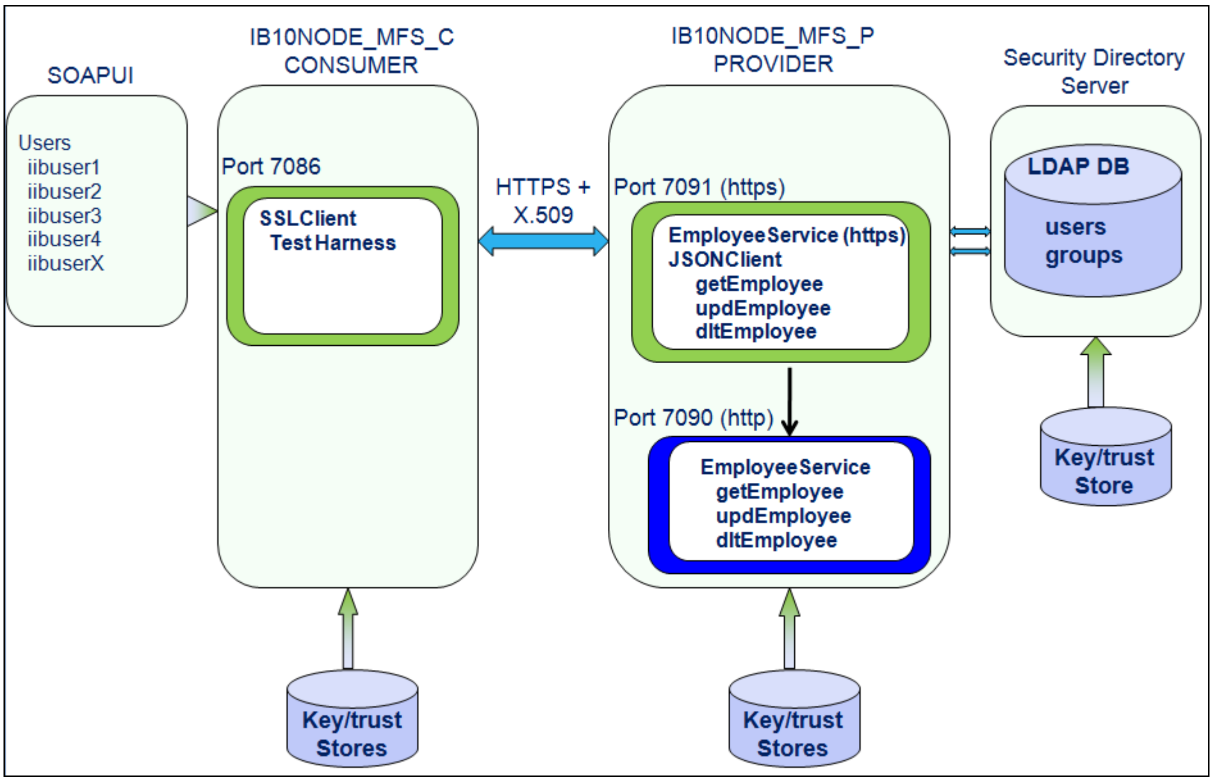
This final section will demonstrate the following capabilities:

- Use X.509 certificates for client authentication
- Extract Common Name (CN) from the X.509 certificate using a map node with XPath
- Use the extracted CN to authorise access to the requested message flow

SOAPUI is not capable of attaching an X.509 certificate to a plain REST request, so you will use a test harness to provide the client system. This client will be an IIB message flow, running in the IIB node called IB10NODE_MFS_C, server = CONSUMER.

This schematic shows the system context for this section of the lab.

- CONSUMER contains a message flow called SSLClient
- CONSUMER listens on port 7086
- SSLClient is invoked by SOAPUI - passing various client parameters
- SSLClient receives these parameters and constructs a REST request which is sent to PROVIDER over HTTPS. This also includes an X.509 certificate, based on the key alias parameter passed from SOAPUI
- PROVIDER application now has a Security PEP node which uses the embedded CN to authorise access to the flow.



8.1 Define the Private Key Infrastructure (PKI)

In this scenario, the CONSUMER and PROVIDER nodes need to establish an SSL connection between them. This is achieved by using the PROVIDER and CONSUMER key and trust stores, and by configuring the IIB nodes and servers.

Because you have earlier run the command file **SetupPKI_forEmbeddedListeners**, the PKI infrastructure has already been defined.

As a reminder, in addition to the PROVIDER PKI configuration, the following definitions were also made for the CONSUMER side:

Define location of keystore

```
      mqsichangeproperties IB10NODE_MFS_C
            -e CONSUMER
            -o ComIbmJVMManager
            -n keystoreFile
            -v c:\student10\Integration_service_MessageFlowSecurity
            \keystores\ConsumerKeyStore.jks
```

Define location of truststore

```
      mqsichangeproperties IB10NODE_MFS_C
            -e CONSUMER
            -o ComIbmJVMManager
            -n truststoreFile
            -v c:\student10\Integration_service_MessageFlowSecurity
            \keystores\ConsumerTrustStore.jks
```

Specify password of keystore

```
      mqsisetdbparms IB10NODE_MFS_C
            -n brokerKeystore::password
            -u ignore
            -p passw0rd
```

Specify password of truststore

```
      mqsisetdbparms IB10NODE_MFS_C
            -n brokerTruststore::password
            -u ignore
            -p passw0rd
```

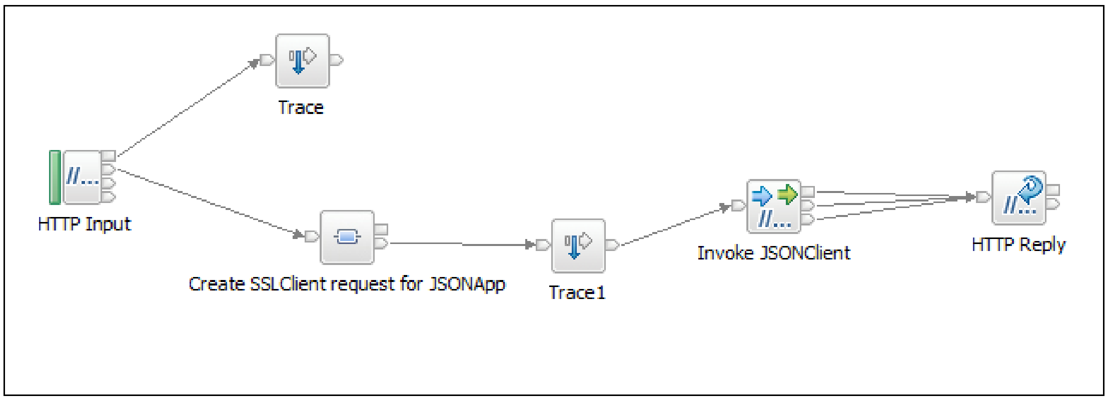
You can review the current settings with commands like:

```
      mqsireportproperties IB10NODE_MFS_C
            -e CONSUMER
            -o ComIbmJVMManager
            -a
```

8.2 Review the SSLClient test harness application

1. In the Toolkit, import the PI file SSLClient.zip from the \applications folder.

   Open the SSLTestHarness message flow.



2. You are not required to make any changes to this message flow. However, you may find it instructive to review the mapping node logic in this flow. The following transformations are provided in the "Create SSLClientRequest for JSON App" map.

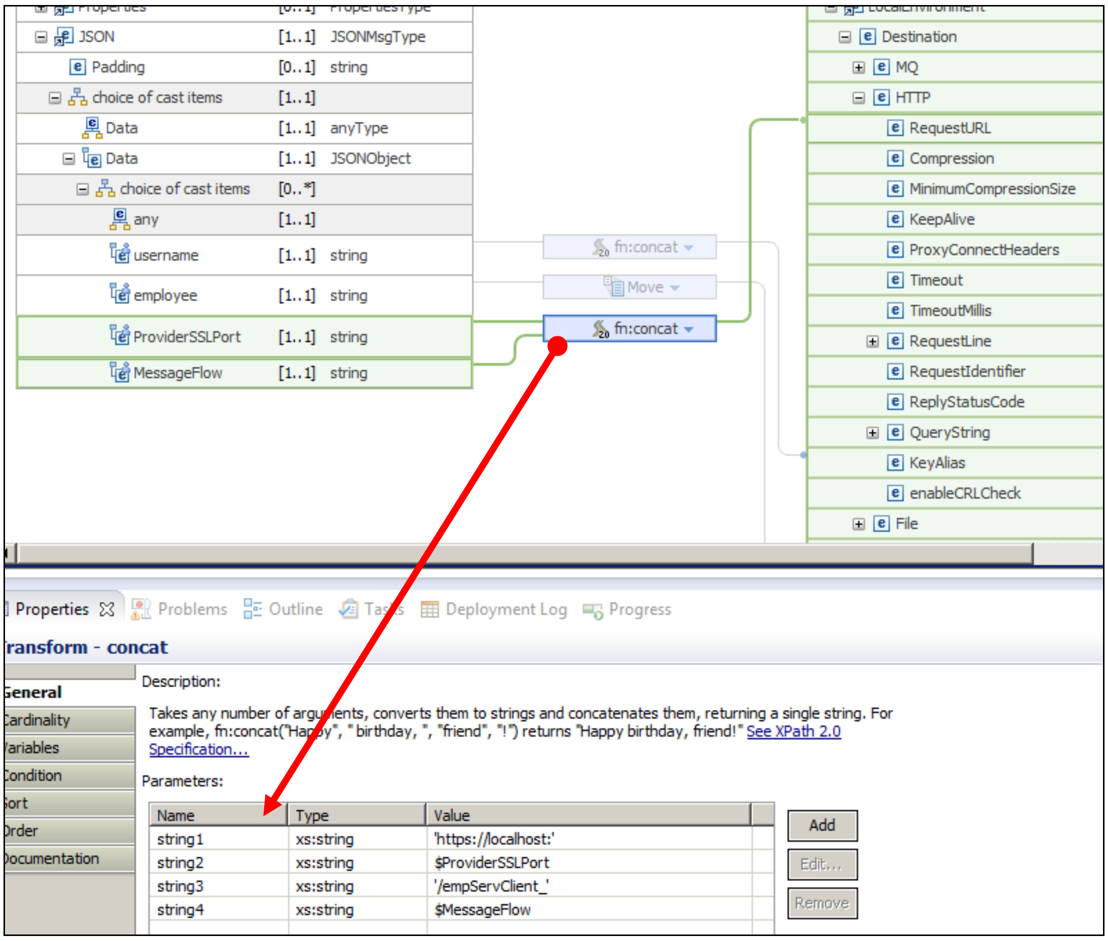
   The following elements are mapped. The input elements are defined as JSON object elements. The output elements are all contained within the LocalEnvironment/HTTP folder. These Local Environment elements are used by the subsequent "Invoke JSONClient" SOAP Request node.

   Input Output
   - username ==> KeyAlias (this is used to reference the matching SSL certificate)
   - employee ==> empNumber (primary input to the message flow)
   - ProviderSSLPort ==> RequestURL (part of)
   - MessageFlow ==> RequestURL (part of)

   As an example, this screen capture shows the mapping of the RequestURL. This element will be used to invoke the getEmployee message flow on the port specified by the SOAPUI input. This element uses a Concat function to fully construct the target URL, as follows:

   `http://localhost:port/empServClient/getEmployee`

   String1 and string3 are hardcoded values "http://localhost" and "/empServClient".



Description:

Takes any number of arguments, converts them to strings and concatenates them, returning a single string. For example, fn:concat("Happy", " birthday, ", "friend", "!") returns "Happy birthday, friend!" See XPath 2.0 Specification...

Parameters:

Name	Type	Value
string1	xs:string	'https://localhost:'
string2	xs:string	$ProviderSSLPort
string3	xs:string	'/empServClient_'
string4	xs:string	$MessageFlow

8.3 Update the JSON Client message flows

1. Because the message flows in PROVIDER are now going to accept an incoming X.509 certificate, you need to make several changes to the message flows (described in point 2).

   - The HTTP input node
     - Change to accept HTTPS
     - Change to specify Identity Token Type = X.509

   - Change JSON_TO_SOAP mapping node to extract the Distinguished Name from the X.509 certificate, and copy it to the Local Environment.

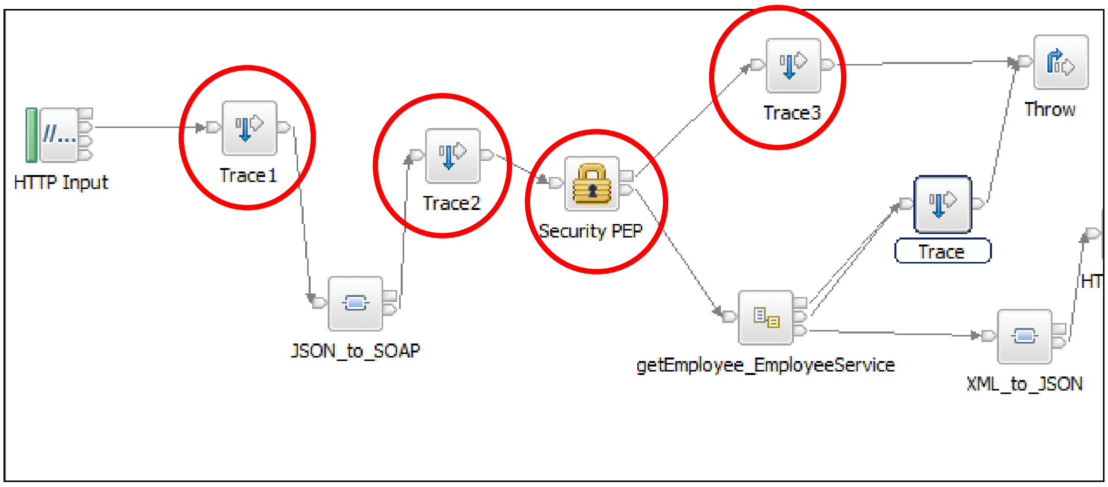
   - Add a Trace node after this updated mapping node (aids problem determination)

   - Add a Security PEP node to use the extracted DN for authorisation against LDAP

   This lab will update the getEmployee scenario. Similar changes would be necessary for the other flows in the application.

   In the Toolkit, open the EmpServ_JSON_getEmployee message flow.

   Drop three Trace nodes and a Security PEP node onto the flow and connect as shown.



2. Set the following node properties

   - HTTP Input
     - Basic
       - Use HTTPS = ticked
     - Security - Identity Token Type = X.509 Certificate

   - All Trace nodes
     - Destination = Local Error Log
     - Pattern

       ${LocalEnvironment}
       ${Root}

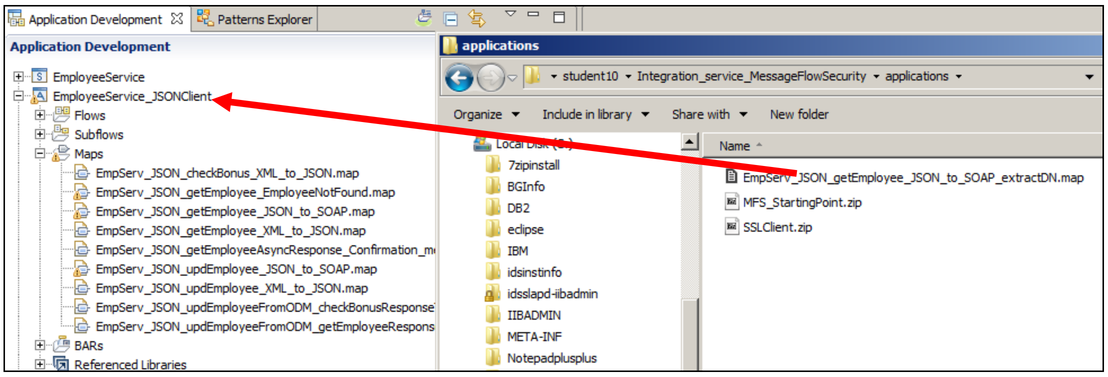
   - Security PEP node
     - Basic
       - Identity Token Type = username

       - Identity token location = **$LocalEnvironment/IdentityToAuthorise** (make sure you type this exactly as shown. This element will be referenced by the mapping node, which you will update in the next step)

3. The JSON_to_SOAP Mapping node needs to be updated to extract the Distinguished Name of the client from the accompanying X.509 certificate.

   Since the XPath statement could be mistyped, we have provided a prebuilt replacement mapping node, EmpServ_JSON_to_SOAP_extractDN_10_0_0_3.map.

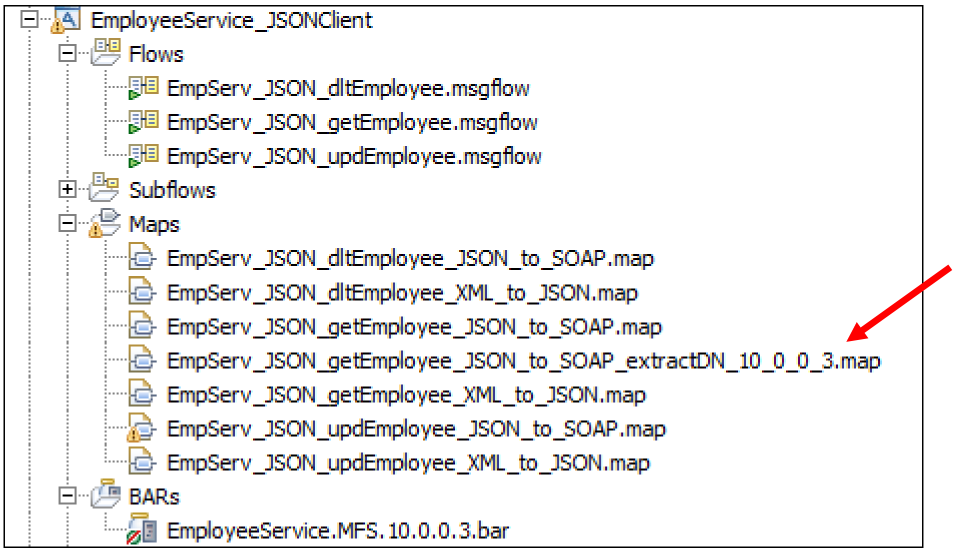
   Copy / paste this map from the \applications folder (in Windows Explorer) onto the EmployeeService_JSONClient application (in the Toolkit). The map will automatically be placed under the \maps folder.



```
Application Development   Patterns Explorer
Application Development
+ EmployeeService
- EmployeeService_JSONClient
  + Flows
  + Subflows
  - Maps
      EmpServ_JSON_checkBonus_XML_to_JSON.map
      EmpServ_JSON_getEmployee_EmployeeNotFound.map
      EmpServ_JSON_getEmployee_JSON_to_SOAP.map
      EmpServ_JSON_getEmployee_XML_to_JSON.map
      EmpServ_JSON_getEmployeeAsyncResponse_Confirmation_m
      EmpServ_JSON_updEmployee_JSON_to_SOAP.map
      EmpServ_JSON_updEmployee_XML_to_JSON.map
      EmpServ_JSON_updEmployeeFromODM_checkBonusResponse
      EmpServ_JSON_updEmployeeFromODM_getEmployeeRespons
  + BARs
  + Referenced Libraries
```

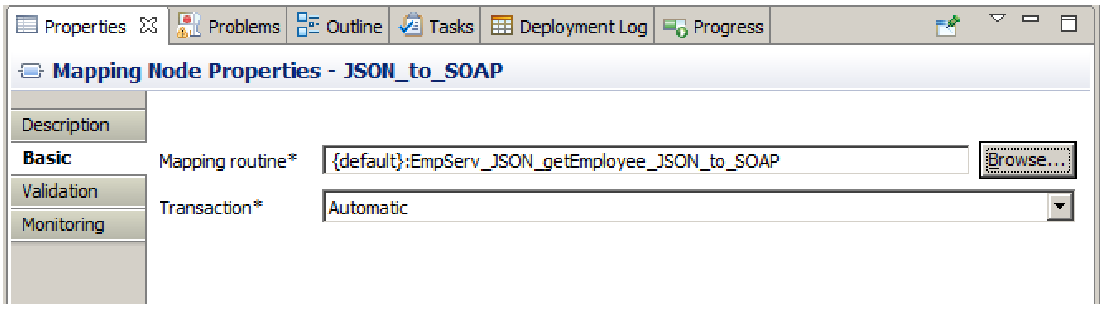
```
applications
student10 ▸ Integration_service_MessageFlowSecurity ▸ applications
Organize   Include in library   Share with   New folder

Local Disk (C:)          Name
  7zipinstall            EmpServ_JSON_getEmployee_JSON_to_SOAP_extractDN.map
  BGInfo                 MFS_StartingPoint.zip
  DB2                    SSLClient.zip
  eclipse
  IBM
  idsinstinfo
  idsslapd-iibadmin
  IIBADMIN
  META-INF
  Notepadplusplus
```

Result:



```
- EmployeeService_JSONClient
  - Flows
      EmpServ_JSON_dltEmployee.msgflow
      EmpServ_JSON_getEmployee.msgflow
      EmpServ_JSON_updEmployee.msgflow
  + Subflows
  - Maps
      EmpServ_JSON_dltEmployee_JSON_to_SOAP.map
      EmpServ_JSON_dltEmployee_XML_to_JSON.map
      EmpServ_JSON_getEmployee_JSON_to_SOAP.map
      EmpServ_JSON_getEmployee_JSON_to_SOAP_extractDN_10_0_0_3.map
      EmpServ_JSON_getEmployee_XML_to_JSON.map
      EmpServ_JSON_updEmployee_JSON_to_SOAP.map
      EmpServ_JSON_updEmployee_XML_to_JSON.map
  - BARs
      EmployeeService.MFS.10.0.0.3.bar
```

4. In the Properties of the JSON_to_SOAP mapping node on the flow, you will see the current name of the map file.


Properties | Problems | Outline | Tasks | Deployment Log | Progress

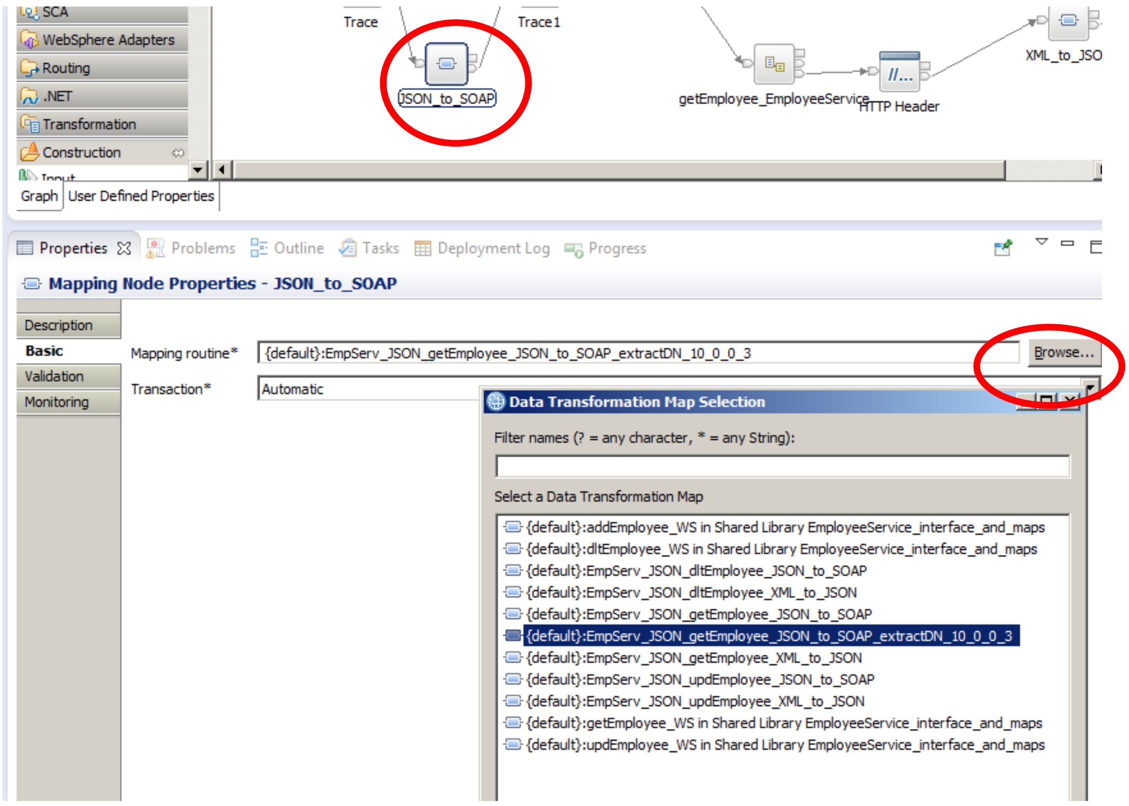
Mapping Node Properties - JSON_to_SOAP

Description
Basic
Validation
Monitoring

Mapping routine* {default}:EmpServ_JSON_getEmployee_JSON_to_SOAP Browse...
Transaction* Automatic

Use the Browse button to set the map name to EmpServ_JSON_to_SOAP_extractDN_10_0_0_3.map.

Click OK.



Properties | Problems | Outline | Tasks | Deployment Log | Progress

Mapping Node Properties - JSON_to_SOAP

Description
Basic
Validation
Monitoring

Mapping routine* {default}:EmpServ_JSON_getEmployee_JSON_to_SOAP_extractDN_10_0_0_3 Browse...
Transaction* Automatic

Data Transformation Map Selection

Filter names (? = any character, * = any String):

Select a Data Transformation Map

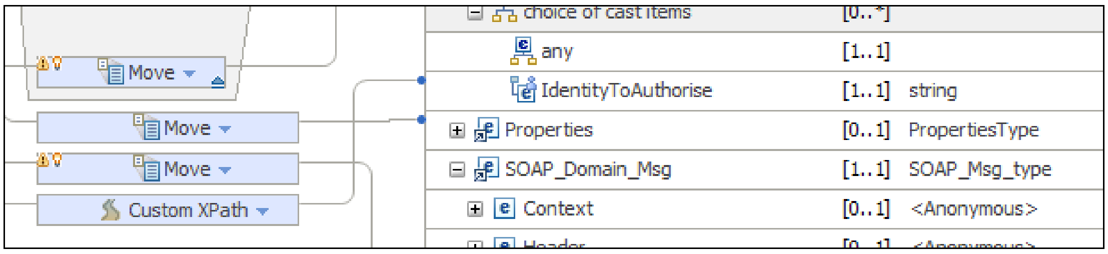
{default}:addEmployee_WS in Shared Library EmployeeService_interface_and_maps
{default}:dltEmployee_WS in Shared Library EmployeeService_interface_and_maps
{default}:EmpServ_JSON_dltEmployee_JSON_to_SOAP
{default}:EmpServ_JSON_dltEmployee_XML_to_JSON
{default}:EmpServ_JSON_getEmployee_JSON_to_SOAP
{default}:EmpServ_JSON_getEmployee_JSON_to_SOAP_extractDN_10_0_0_3
{default}:EmpServ_JSON_getEmployee_XML_to_JSON
{default}:EmpServ_JSON_updEmployee_JSON_to_SOAP
{default}:EmpServ_JSON_updEmployee_XML_to_JSON
{default}:getEmployee_WS in Shared Library EmployeeService_interface_and_maps
{default}:updEmployee_WS in Shared Library EmployeeService_interface_and_maps

5. Open the new map.

   Expand the input message, and expand Properties. You will see that the element IdentitySourceToken has been connected to a Custom XPath transform.

   Also observe that this transform has been mapped to the output element:

   `LocalEnvironment / IdentityToAuthorise`



choice of cast items [0..*]
any [1..1]
IdentityToAuthorise [1..1] string
Properties [0..1] PropertiesType
SOAP_Domain_Msg [1..1] SOAP_Msg_type
Context [0..1] <Anonymous>
Header [0..1] <Anonymous>
Move
Move
Move
Custom XPath

   Observe that the Custom XPath transform has been defined with the following transform (click on the transform to bring its properties into view).

```
fn:substring-after(fn:substring-before($IdentitySourceToken,",OU="),"CN=")
```

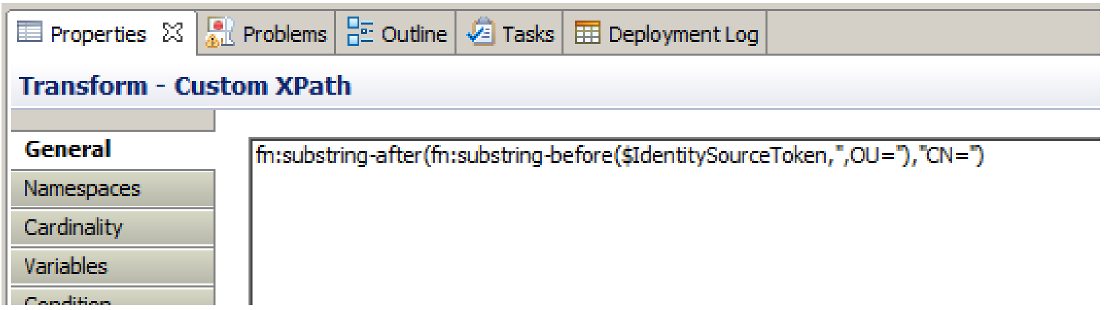
   This is an example of a nested XPath statement. This is required because the typical format of a Distinguished Name is like this:

   `CN=iibuser1,OU=users,OU=iib,O=IBM`

   The **fn:substring-before** will extract all input prior to the string ",OU=", so the interim result will be

   `CN=iibuser1`

   The **fn:substring-after** will then strip off the first three characters, "CN=", leaving the "iibuser1" string, which will be passed to the Security PEP node for authorisation.

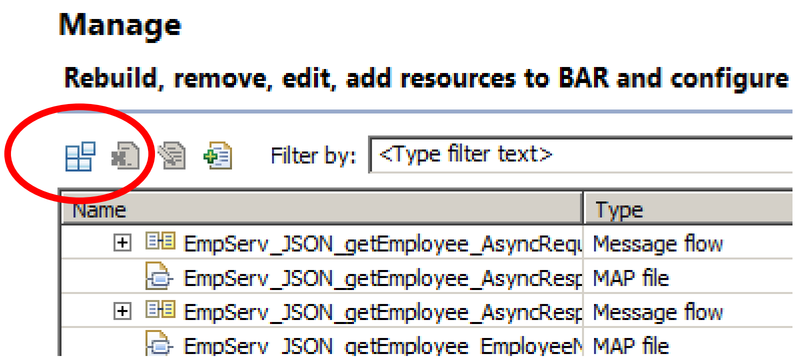


```
fn:substring-after(fn:substring-before($IdentitySourceToken,",OU="),"CN=")
```

   Close the map.

   Save the message flow.

6. Open the EmployeeService_JSONClient.MFS.10.0.0.3.bar, and rebuild the barfile. If you are on the Manage tab in the editor, you can use the Rebuild and Save BAR button.


Manage

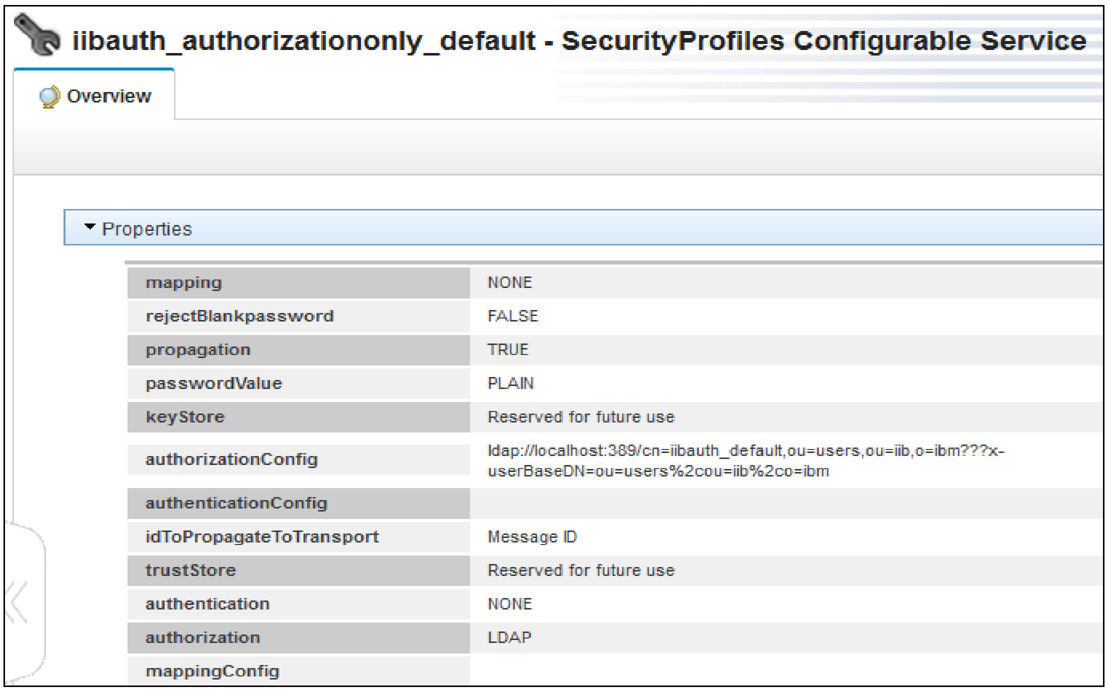
Rebuild, remove, edit, add resources to BAR and configure

Name	Type
EmpServ_JSON_getEmployee_AsyncRequ	Message flow
EmpServ_JSON_getEmployee_AsyncResp	MAP file
EmpServ_JSON_getEmployee_AsyncResp	Message flow
EmpServ_JSON_getEmployee_EmployeeN	MAP file

8.4 Review and specify the Authorization Security Profile

1. Earlier in the lab, you defined three security profiles to authorise access requests for users who were authenticated with the X.509 certificates.

   The following screen capture shows the iibauth_authorizationonly_default profile. You can look at this in the web admin browser.



iibauth_authorizationonly_default - SecurityProfiles Configurable Service

mapping	NONE
rejectBlankpassword	FALSE
propagation	TRUE
passwordValue	PLAIN
keyStore	Reserved for future use
authorizationConfig	ldap://localhost:389/cn=iibauth_default,ou=users,ou=iib,o=ibm???x-userBaseDN=ou=users%2cou=iib%2co=ibm
authenticationConfig	
idToPropagateToTransport	Message ID
trustStore	Reserved for future use
authentication	NONE
authorization	LDAP
mappingConfig	

Points to note:

- The authentication property is set to NONE
- The authenticationConfig is blank
- The authorization property is set to LDAP
- The authorizationConfig is set to **`ldap://localhost:389/cn=iibauth_default,ou=users,ou=iib,o=ibm???x-userBaseDN=ou=users%2cou=iib%2co=ibm`**

Read next part carefully
More information is available in the Knowledge Centre, topic ap04141

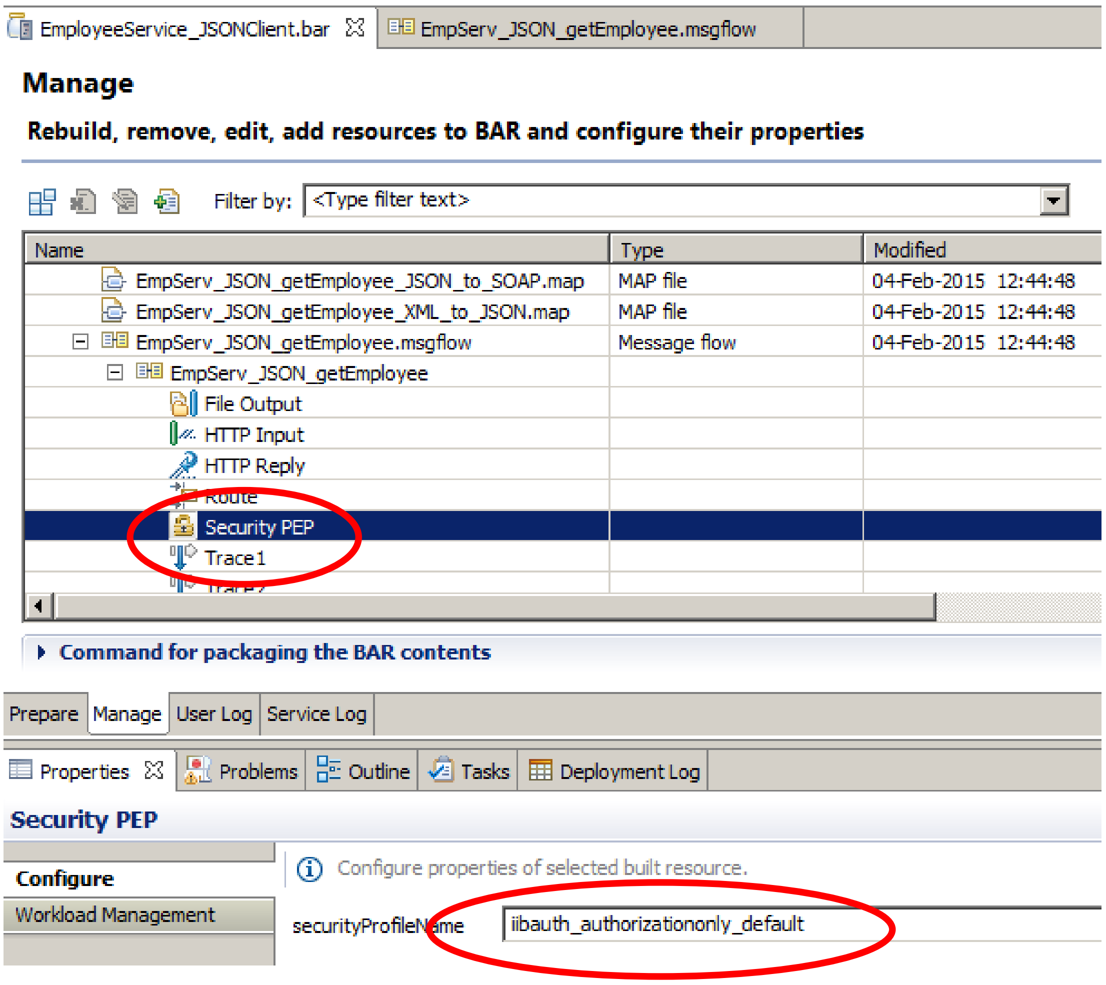
The authorizationConfig string comprises three logical parts. These parts are concatenated into a single string:

- LDAP server connection details
  - **`ldap://localhost:389`**
- Fully qualified DN of the relevant LDAP authorization group
  - **`/cn=iibauth_default,ou=users,ou=iib,o=ibm`**
- String defining the base distinguished name of all users in the directory. Must be preceded by the string "**`x-userBaseDN`**"
  - **`???x-userBaseDN=ou=users%2cou=iib%2co=ibm`**

Note - embedded commas must be replaced by the hex string "%2c", as in the example above.

2. In the EmployeeService_JSONClient.MFS.10.0.0.3.bar, on the Manage tab, expand the EmpServ_JSON_getEmployee message flow, and select the Security PEP node.

   Set securityProfileName = iibauth_authorizationonly_default.



EmployeeService_JSONClient.bar | EmpServ_JSON_getEmployee.msgflow

Manage

Rebuild, remove, edit, add resources to BAR and configure their properties

Filter by: <Type filter text>

Name	Type	Modified
EmpServ_JSON_getEmployee_JSON_to_SOAP.map	MAP file	04-Feb-2015 12:44:48
EmpServ_JSON_getEmployee_XML_to_JSON.map	MAP file	04-Feb-2015 12:44:48
EmpServ_JSON_getEmployee.msgflow	Message flow	04-Feb-2015 12:44:48
EmpServ_JSON_getEmployee		
File Output		
HTTP Input		
HTTP Reply		
Route		
Security PEP		
Trace1		

Command for packaging the BAR contents

Prepare | Manage | User Log | Service Log

Properties | Problems | Outline | Tasks | Deployment Log

Security PEP

Configure

Workload Management

Configure properties of selected built resource.

securityProfileName: iibauth_authorizationonly_default

3. Remove the Security Profile from the message flows (eg. getEmployee).

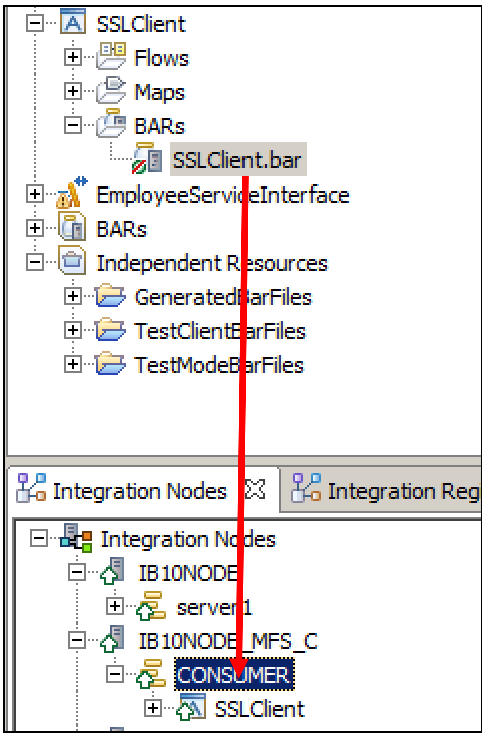
8.5 Deploy the applications

1. Save and redeploy the barfile to the PROVIDER server.

   Because this flow is now running with a HTTPS Input node, you should see the https listener start in the PROVIDER server, using port 7091.

```
BIP2155I: ( IB10NODE_MFS_P.PROVIDER ) About to ''create '' the
deployed resource ''EmpServ_JSON_checkBonus'' of type ''.MSGFLOW''.
[27/01/2015 14:51:47]
BIP3132I: ( IB10NODE_MFS_P.PROVIDER ) The HTTP Listener has started
listening on port ''7091'' for ''https'' connections. [27/01/2015
14:51:55]
BIP2154I: ( IB10NODE_MFS_P.PROVIDER ) Execution group finished with
Configuration message. [27/01/2015 14:51:56]
```

2. Deploy the SSLClient.bar to IB10NODE_MFS_C / CONSUMER.



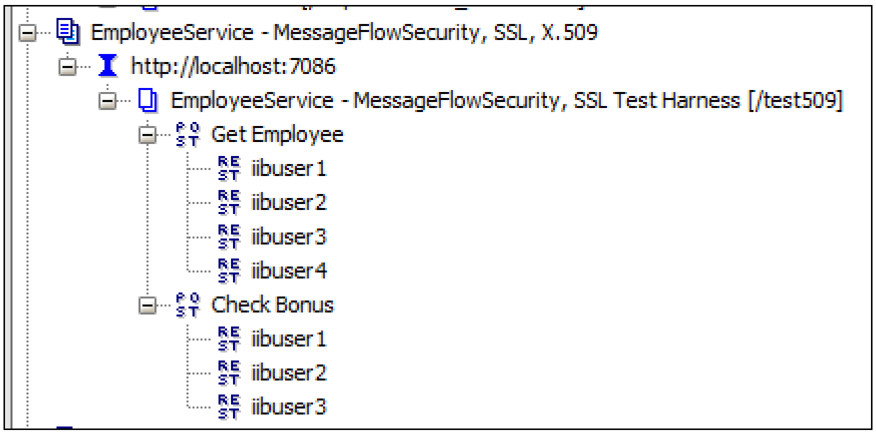
8.6 Test the application with X.509 certificates

1. In SOAPUI, ensure the Prebuilt workspace is open.

   Expand the project **EmployeeService - MessageFlowSecurity, SSL, X.509.**

   For ease of testing, the project is split into several sections. The Get Employee method has several requests, each one representing different users. All requests in this method invoke the same message flow, and read the same employee record.

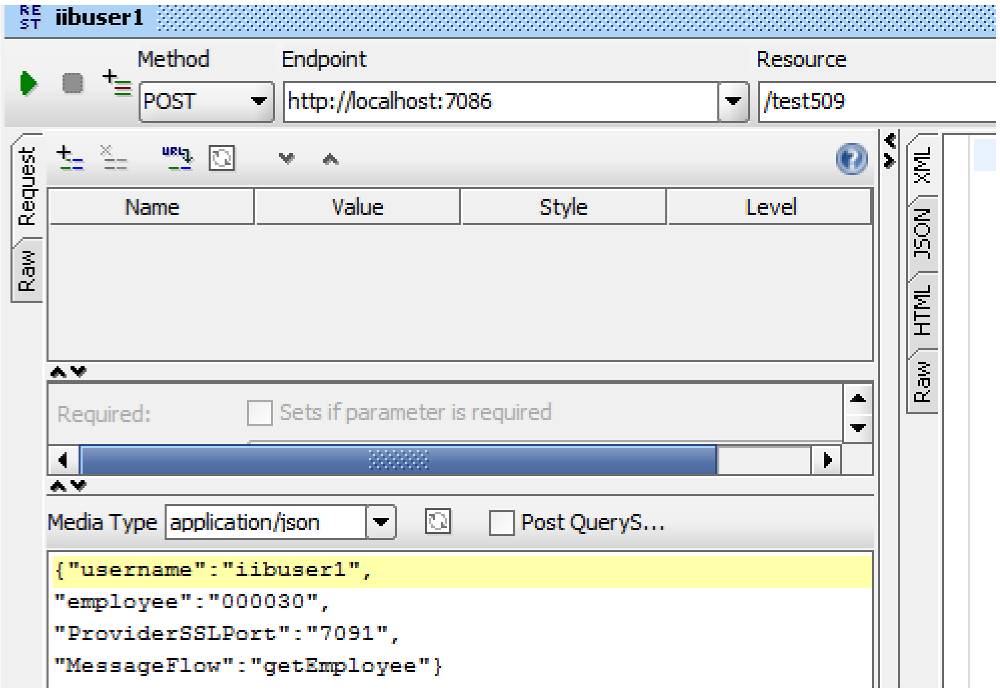
   Similarly, the Check Bonus method has several requests for different users. All request invoke the same message flow.



```
EmployeeService - MessageFlowSecurity, SSL, X.509
  http://localhost:7086
    EmployeeService - MessageFlowSecurity, SSL Test Harness [/test509]
      Get Employee
        iibuser1
        iibuser2
        iibuser3
        iibuser4
      Check Bonus
        iibuser1
        iibuser2
        iibuser3
```

2. Open the "iibuser1" request in Get Employee.

   The input data shows the elements that will be sent to the SSLClient flow. Although you can edit these, there should be enough prebuilt requests for the various tests that are needed.



iibuser1

Method: POST
Endpoint: http://localhost:7086
Resource: /test509

Name	Value	Style	Level

Required: Sets if parameter is required

Media Type: application/json
Post QueryS...

```
{"username":"iibuser1",
"employee":"000030",
"ProviderSSLPort":"7091",
"MessageFlow":"getEmployee"}
```

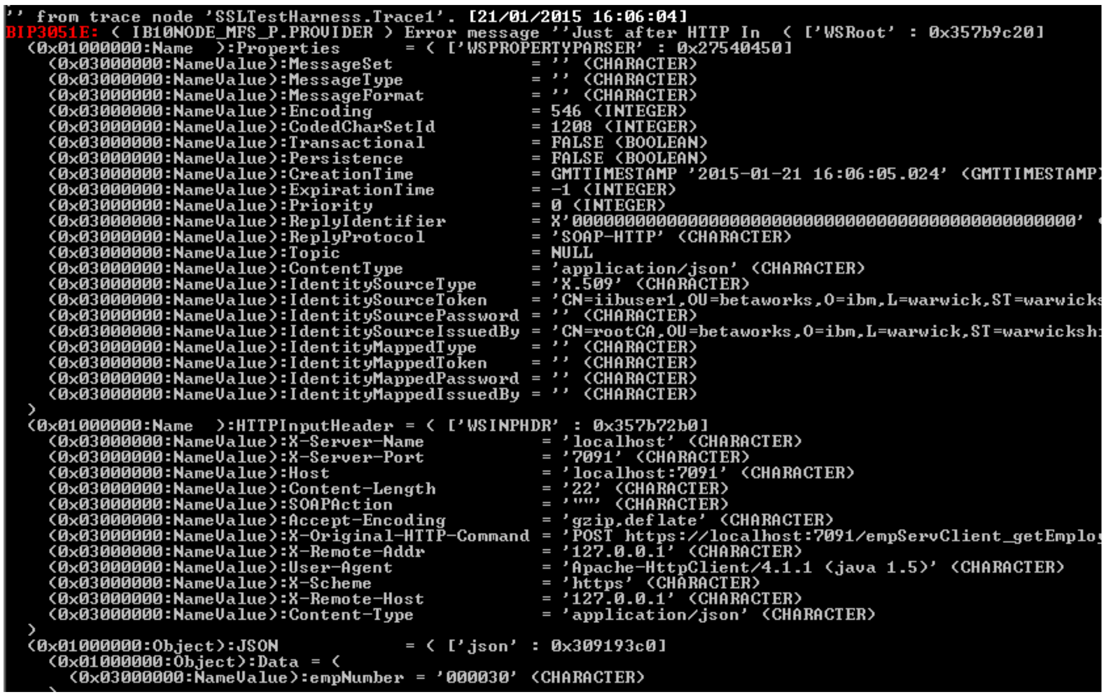
3. Ensure the IIB Event Log Viewer is open (icon on the Start menu).

   Click the green arrow to run the test. The output from the Trace nodes is instructive to review at this point.

   You should see something like this, from the first trace node.

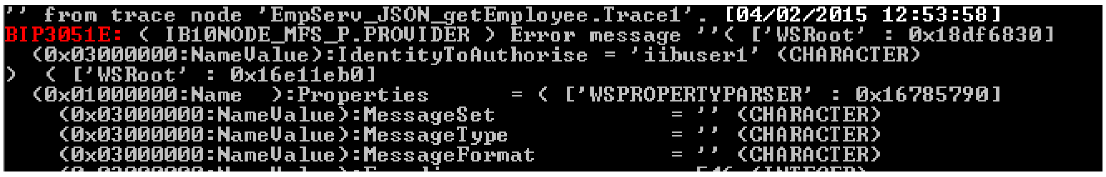
   Key points to note:
   - IdentitySourceType = X.509
   - IdentitySourceToken = "CN=iibuser1,OU=betaworks,O=ibm,L=warwick........"
   - IdentitySourceIssuedBy = "CN=rootCA,OU=betaworks,O=ibm,L=warwick ......"
   - Server-Port = 7091

   And in the JSON message payload:
   - Data/empNumber = "000030"



```
'' from trace node 'SSLTestHarness.Trace1'. [21/01/2015 16:06:04]
BIP3051E: ( IB10NODE_MFS_P.PROVIDER ) Error message ''Just after HTTP In  ( ['WSRoot' : 0x357b9c20]
  (0x01000000:Name  ):Properties       = ( ['WSPROPERTYPARSER' : 0x27540450]
    (0x03000000:NameValue):MessageSet               = '' (CHARACTER)
    (0x03000000:NameValue):MessageType              = '' (CHARACTER)
    (0x03000000:NameValue):MessageFormat            = '' (CHARACTER)
    (0x03000000:NameValue):Encoding                 = 546 (INTEGER)
    (0x03000000:NameValue):CodedCharSetId           = 1208 (INTEGER)
    (0x03000000:NameValue):Transactional            = FALSE (BOOLEAN)
    (0x03000000:NameValue):Persistence              = FALSE (BOOLEAN)
    (0x03000000:NameValue):CreationTime             = GMTTIMESTAMP '2015-01-21 16:06:05.024' (GMTTIMESTAMP
    (0x03000000:NameValue):ExpirationTime           = -1 (INTEGER)
    (0x03000000:NameValue):Priority                 = 0 (INTEGER)
    (0x03000000:NameValue):ReplyIdentifier          = X'000000000000000000000000000000000000000000000000'
    (0x03000000:NameValue):ReplyProtocol            = 'SOAP-HTTP' (CHARACTER)
    (0x03000000:NameValue):Topic                    = NULL
    (0x03000000:NameValue):ContentType              = 'application/json' (CHARACTER)
    (0x03000000:NameValue):IdentitySourceType       = 'X.509' (CHARACTER)
    (0x03000000:NameValue):IdentitySourceToken      = 'CN=iibuser1,OU=betaworks,O=ibm,L=warwick,ST=warwicks
    (0x03000000:NameValue):IdentitySourcePassword   = '' (CHARACTER)
    (0x03000000:NameValue):IdentitySourceIssuedBy   = 'CN=rootCA,OU=betaworks,O=ibm,L=warwick,ST=warwicksh
    (0x03000000:NameValue):IdentityMappedType       = '' (CHARACTER)
    (0x03000000:NameValue):IdentityMappedToken      = '' (CHARACTER)
    (0x03000000:NameValue):IdentityMappedPassword   = '' (CHARACTER)
    (0x03000000:NameValue):IdentityMappedIssuedBy   = '' (CHARACTER)
  )
  (0x01000000:Name  ):HTTPInputHeader = ( ['WSINPHDR' : 0x357b72b0]
    (0x03000000:NameValue):X-Server-Name             = 'localhost' (CHARACTER)
    (0x03000000:NameValue):X-Server-Port             = '7091' (CHARACTER)
    (0x03000000:NameValue):Host                      = 'localhost:7091' (CHARACTER)
    (0x03000000:NameValue):Content-Length            = '22' (CHARACTER)
    (0x03000000:NameValue):SOAPAction                = '""' (CHARACTER)
    (0x03000000:NameValue):Accept-Encoding           = 'gzip,deflate' (CHARACTER)
    (0x03000000:NameValue):X-Original-HTTP-Command   = 'POST https://localhost:7091/empServClient_getEmplo
    (0x03000000:NameValue):X-Remote-Addr             = '127.0.0.1' (CHARACTER)
    (0x03000000:NameValue):User-Agent                = 'Apache-HttpClient/4.1.1 (java 1.5)' (CHARACTER)
    (0x03000000:NameValue):X-Scheme                  = 'https' (CHARACTER)
    (0x03000000:NameValue):X-Remote-Host             = '127.0.0.1' (CHARACTER)
    (0x03000000:NameValue):Content-Type              = 'application/json' (CHARACTER)
  )
  (0x01000000:Object):JSON             = ( ['json' : 0x309193c0]
    (0x01000000:Object):Data = (
      (0x03000000:NameValue):empNumber = '000030' (CHARACTER)
```

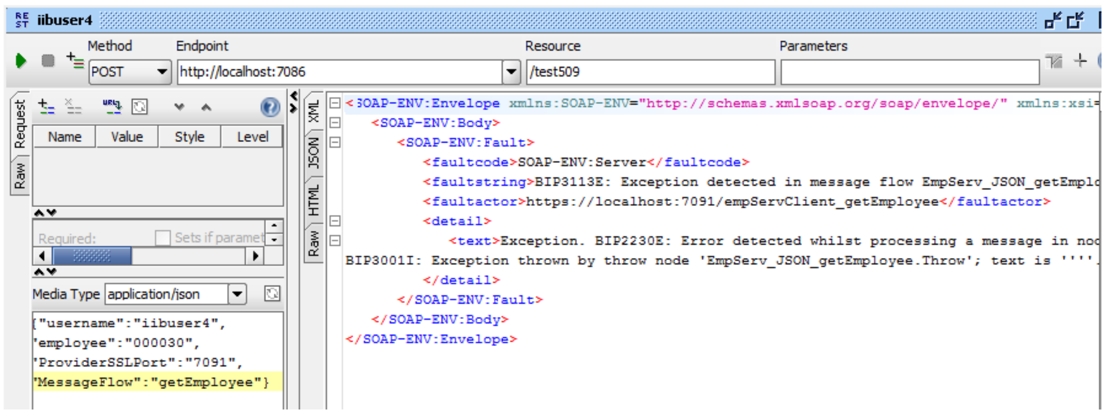
4. From the second trace node, you will see that the mapping node has extracted the Distinguished Name and placed it in the Local Environment/IdentityToAuthorise element.



```
'' from trace node 'EmpServ_JSON_getEmployee.Trace1'. [04/02/2015 12:53:58]
BIP3051E: ( IB10NODE_MFS_P.PROVIDER ) Error message ''( ['WSRoot' : 0x18df6830]
  (0x03000000:NameValue):IdentityToAuthorise = 'iibuser1' (CHARACTER)
)  ( ['WSRoot' : 0x16e11eb0]
  (0x01000000:Name  ):Properties       = ( ['WSPROPERTYPARSER' : 0x16785790]
    (0x03000000:NameValue):MessageSet               = '' (CHARACTER)
    (0x03000000:NameValue):MessageType              = '' (CHARACTER)
    (0x03000000:NameValue):MessageFormat            = '' (CHARACTER)
```

5. Now try the test using "iibuser4". This user is defined as a valid user in the LDAP database. However, it is not a member of the LDAP group "iibauth_default", so it should not be allowed to execute the getEmployee message flow.

   In the SOAPUI output, you should see something like this:

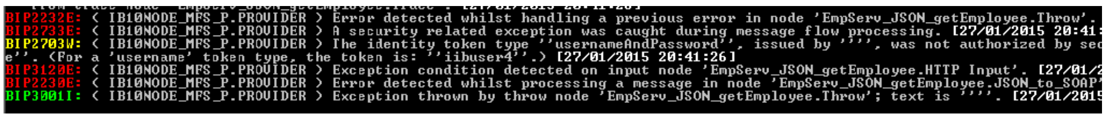


Method: POST; Endpoint: http://localhost:7086; Resource: /test509; Media Type: application/json

```
{"username":"iibuser4",
"employee":"000030",
"ProviderSSLPort":"7091",
"MessageFlow":"getEmployee"}
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi=
   <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode>SOAP-ENV:Server</faultcode>
         <faultstring>BIP3113E: Exception detected in message flow EmpServ_JSON_getEmplo
         <faultactor>https://localhost:7091/empServClient_getEmployee</faultactor>
         <detail>
            <text>Exception. BIP2230E: Error detected whilst processing a message in noo
BIP3001I: Exception thrown by throw node 'EmpServ_JSON_getEmployee.Throw'; text is ''''.
         </detail>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

   And in the IIB Event Log Viewer, you should see this. Scroll further right to see the full detail. The messages indicate that the LDAP has been unable to authorise access to the required message flow for user "iibuser4".



```
BIP2232E: ( IB10NODE_MFS_P.PROVIDER ) Error detected whilst handling a previous error in node 'EmpServ_JSON_getEmployee.Throw'.
BIP2733E: ( IB10NODE_MFS_P.PROVIDER ) A security related exception was caught during message flow processing. [27/01/2015 20:41:
BIP2703W: ( IB10NODE_MFS_P.PROVIDER ) The identity token type ''usernameAndPassword'', issued by '''', was not authorized by sec
e''. (For a 'username' token type, the token is: ''iibuser4''.) [27/01/2015 20:41:26]
BIP3120E: ( IB10NODE_MFS_P.PROVIDER ) Exception condition detected on input node 'EmpServ_JSON_getEmployee.HTTP Input'. [27/01/2
BIP2230E: ( IB10NODE_MFS_P.PROVIDER ) Error detected whilst processing a message in node 'EmpServ_JSON_getEmployee.JSON_to_SOAP'
BIP3001I: ( IB10NODE_MFS_P.PROVIDER ) Exception thrown by throw node 'EmpServ_JSON_getEmployee.Throw'; text is ''''. [27/01/2015
```

```
BIP2703W: ( IB10NODE_MFS_P.PROVIDER ) The identity token type
''username'', issued by '''', was not authorized by security
provider ''LDAP'' to access message flow
'EmpServ JSON_getEmployee''. (For a 'username' token type, the token
is: ''iibuser4''.) [04/02/2015 12:56:25]
```

END OF LAB GUIDE