

# **IBM Integration Bus**

Web Services Security: Authentication and Encryption Using X.509 Certificates

Featuring:

- Policy Sets and Policy Set Bindings
- Full and Partial Message Encryption
- Client Authentication and TLS

January 2016 Hands-on lab built at product Version 10.0.0.3

1.	INTRODUCTION	4
2.	PREPARE THE ENVIRONMENT	7
	<ul> <li>2.1.1 The "Consumer"</li></ul>	7 7 7 8 9
3.	PREPARE THE CONSUMER AND PROVIDER	12
	<ul> <li>3.1 PREPARE THE PROVIDER</li></ul>	13 13 16 17 19
4.	SET UP IIB TO USE A KEY AND TRUST STORE	21
5.	SCENARIO: TEST THE SERVICE (NO ACTIVE SECURITY)	22
6.	POLICY SETS AND BINDINGS IN IIB	24
7.	SCENARIO: AUTHENTICATION USING WS-SECURITY	25
	<ul> <li>7.1 POLICY SET AND BINDINGS FOR AUTHENTICATION</li></ul>	25 26 27 28 28 29 29 29 31
8.	MESSAGE LEVEL ENCRYPTION	33
	<ul> <li>8.1 PREPARE THE PKI FOR ENCRYPTION</li></ul>	34 35 35 36 37 37 37 38 39 41 41
	8.3.2 Reset the TCP/IP Monitor	42

9.2.4 Varify the Energy price in the TCD/ID Maniter	43
9. TRANSPORT LEVEL SECURITY (TLS)	45
9.1.1 Configure the provider to use https	45
9.2 CALL THE PROVIDER USING HTTPS AND CLIENT AUTHENTICATION	49
9.3 TEST TLS SCENARIO	52
9.3.1 Reconfigure the TCP/IP Monitor properties	
9.3.2 Send a message using SOAPUI	
9.3.3 Verify the test in the TCP/IP Monitor	54
END OF LAB	54
10. APPENDIX	55
10.1 DEFINING THE SELF SIGNED CERTIFICATES USED IN THIS LAB GUIDE	55
10.1.1 Preparing your openssl environment:	
10.1.2 Create the RootCA:	
10.1.3 Create the signed CONSUMER certificate:	
10.1.4 Import the keys and certificates:	

# 1. Introduction

This lab guide covers how IBM Integration Bus Integration Services can be secured using (message level) Web Services security and Transport Level Security (TLS) using X.509 certificates.

In all scenarios throughout this lab guide a Provider service, "EmployeeService" is called by a JSON client.

The TCP/IP Monitor tool, provided in the IIB Toolkit, is used to see the effects of the message level authentication and encryption that WS-Security configuration has on SOAP messages. A final scenario shows how the service can be further secured by using Transport Level Security (TLS) by defining the service to use https.

The guide is split into the following scenarios:

1) Initially the environment is prepared ready for the security scenarios. The JSON client and EmployeeService are deployed and tested without being configured to use security. The Public Key Infrastructure (PKI) is defined in this section, however no applications or services are configured to use the certificates defined in the PKI. In an ideal situation it is easier to resolve non security related problems without security working. At the end of this part the environment will look like this:



2) The next part covers the configuration necessary to enable Authentication so that successful communication between the JSON client and the EmployeeService is only possible given successful authentication between the two components. The following diagram shows at a high level the configuration required:



3) Encryption covers the configuration necessary to enable Encryption of the message content between the JSON client and the EmployeeService using X.509 certificates. The following diagram shows at a high level the configuration required:



In this part you will see how IIB can be configured to support:

- a) Full request and response message encryption
- b) Partial message encryption (for example encryption of part of the message response containing sensitive data).
- c) The addition of https Transport Level Security configuration between the JSON Client and the EmployeeService

## 2. Prepare the Environment

In the following sections you will prepare your environment to handle Web Services security using a public key infrastructure based on self-signed certificates. You will use a Provider based on the solution for the Integration Services Lab guide (EmployeeService), with the Consumer being based on the JSON client. The client is run on a separate Integration Node and Server and is used to call the EmployeeService, both these services are provided as a completed solution, there will be no need to develop these components.

The Consumer IIB environment will host the JSON Client application. The provider environment will host the EmployeeService.

## 2.1.1 The "Consumer"

A JSON-based client called EmployeeService\_JSONClient will be used to call "the provider". The provider will retrieve employee details from the EMPLOYEE table, and return them to the EmployeeService\_JSONClient.

The client will run in an Integration Server called CONSUMER running on node IB10NODEC.

### 2.1.2 The "Provider"

The "EmployeeService" is an Integration Service that will be used to obtain details registered in the EMPLOYEE (db2) table for a given key.

EmployeeService will run in an Integration Server called PROVIDER on node IB10NODEP.

### 2.1.3 Configuration scripts

Each part of this lab uses a number of scripts (Windows CMD files) to configure various parts of the IIB nodes. These use "mqsi" commands to perform the configuration. These are provided for simplicity of running the lab scenario, but you are encouraged to look in detail at the contents of these script files, and in particular the mqsi commands that are used.

## 2.2 Set up IIB Nodes

In this section you will prepare the IIB nodes IB10NODEP and IB10NODEC. These nodes will be used to deploy the Provider web service and Consumer application.

- 1. Use the Integration Nodes view in the Toolkit to stop all Integration nodes that you have running.
- 2. Use "File > Switch Workspace > other" to create a new workspace, call it WSSEC. (*The Integration Toolkit will restart*).
- 3. Open an Integration Console and navigate to

```
"C:\student10\WSSecurity\commands"
```

4. Run the command:

#### 00CreateNodes.10.0.0.3.cmd

Accept the defaults when prompted to start the script.

The script will create the IIB nodes used in this lab guide and configure them to run the Consumer and Provider web services you will use in this guide.

Note: the script will attempt to delete the two nodes that it is adding before adding them. If the nodes do not exist in your environment you will see an error message (BIP8013E) saying that each node does not exist; you can ignore this message.

5. When prompted to establish if the node "IB10NODEP" is fully started, check the status of the nodes using the Toolkit (you may need to refresh the node in the Integration Nodes view to see the latest status).

Press the enter key, in the Integration Console, when the node is fully started to continue the progress of the script.

Wait for the script to finish.

6. When the script has successfully completed (you will see "00CreateNodes script complete") in the console. Check using IIB Toolkit to see the following new Integration Nodes and servers:



## 2.3 Set up TCP/IP Monitor

This lab will use the TCP/IP Monitor that is provided as part of the standard eclipse installation for IIB. This tool is not shown by default, so perform the following steps to display this tool.

The monitor captures and records TCP/IP traffic by acting as a proxy for the required endpoint, as shown below.

In this scenario, the monitor will listen for traffic on port 7802, and will forward it on to port 7800. Responses are automatically captured and recorded. To allow this, the Client application has to be changed to send the request to the port of the TCP/IP Monitor.



1. In the IIB Toolkit, click Window, Show View, Other.

$\langle  $	Window Help			
	New Window New Editor Hide Toolbar		⇒ -   ≝	
	Open Perspective		Analiantian Development	
	Customize Perspective Save Perspective As Reset Perspective Close Perspective Close All Perspectives		Application Development     Source Explorer     Data Source Explorer     Deployment Log     Integration Nodes	
	Navigation Show Error View	► Alt+Shift+Q, L	Contraction Registries Contine Contin	Alt+Shift+Q, O
	Preferences		Problems Properties Tasks	Alt+Shift+Q, X
			Othor	Altushiftuo o

2. In the Filter field, type TCP, select TCP/IP Monitor, and click OK.



3. The TCP/IP Monitor will open.

Properties	Problems	E Outline	Tasks	🔠 Deployment Log	📱 TCP/IP Monitor 🔀		🔎 🖉 🛃	<b>1</b>	
							Time of request:		
							Response Time:		
							Type: HTTP		
Pequest:					Pernonse:				
Size:				Byte 💌	Size:			Byte	•
Header:					Header:				
Encoding: <no< td=""><td>one&gt;</td><td></td><td></td><td></td><td>Encoding: <pre></pre></td><td>•</td><td></td><td></td><td></td></no<>	one>				Encoding: <pre></pre>	•			
				<u></u>					<b>A</b>

4. Right-click in the open pane to create a new monitoring entry.

Properties	🖳 Problems	∎ ⊡⊡ Outline	🖉 Tasks	🔝 Deployment Log	🖳 TCP/IP Monitor 🔀	
			Properties	S		

5. Click Add to add a new entry for monitoring.

Preferences						
····· TCP/IP Monitor	TCP/IP Mo	onitor				
	Configure TC	P/IP monitors on local and i	remote por	ts.		
	Show the	TCP/IP Monitor view when	there is ac	tivity		
	TCP/IP Monit	ors:				
	Status	Host name	Туре	Local Port	Auto-stari	Add
						Ealt,
						Remove

- 6. Set the following values:
  - Local monitoring port: 7802 (this is the port that will be monitored by the Monitor).
  - Host name: localhost
  - Port: 7800 (this is the port that the monitor will forward TCPIP traffic to).

Click OK.

(	New Monitor	X
	Local monitoring port: 78	02
	Monitor	
	<u>H</u> ost name:	localhost
	Port:	7800
	<u>T</u> ype:	нттр
	Timeout (in milliseconds):	0 *
		Start monitor automatically
	<u></u>	
		OK Cancel
_		

7. Now click Start to start the TCP/IP Monitor.

references						
CP/IP Monitor	TCP/IP M	lonitor				
	Configure T	CP/IP monitors on local	and remote po	·ts.		
	Show th	e TCP/IP Monitor view v	when there is a	ctivity		
	TCP/IP Mon	itors:				
	Status	Host name	Туре	Local Port	Auto-start	Add
	Stoppe	d localhost:7800	нттр	7802	No	Edit
						Remove
					(	Start
						Stop
						Dtop

Click OK to close the TCPIP Monitor preferences.

## 3. Prepare the Consumer and Provider

- 1. Import the JSON Client application and EmployeeService provider into your workspace:
  - "EmployeeService\_JSONClient.getEmployee.10.0.0.3.zip"

```
(in "C:\student10\Integration_service_JSONClient\solution")
```

2. Ensure all projects are selected, and click Finish.

🌐 Import Project Ir	iterchange Contents	<u>- 🗆 ×</u>
Import Projects Import Projects from	a zip file.	ļ,
From zip file: Project location root:	getEmployee. 10.0.0.3.20151221.zip 💌 C: \Users\iibuser \IBM \IIBT 10 \withToolkitFi	Browse
Generation     G	vice vice_JSONClient vice_interface_and_maps	
Select All Deselect	t All Select Referenced	
(?) < Ba	ck Next > Finish	Cancel

## 3.1 Prepare the Provider

In IBM Integration Bus, use of a web services security related Policy Set and binding is configured using the IIB bar file editor. You will now create and store the bar files that you will use to configure the authentication and encryption scenarios later in this guide.

## 3.1.1 Create EmployeeService bar file

1. In Integration Toolkit, right click on the EmployeeService service and select "New > Bar File" from this list of options.

The "Create a new bar file" editor will open.

2. Call the bar file "ManualBuildEmployeeService" and specify the EmployeeService as the Container, click Finish:

🛞 New BAI	R file	<u> </u>
Create a n	ew BAR file ew BAR file resource	Ċ.
Container:	EmployeeService	New
Folder:	<default></default>	Browse
Name:	ManualBuildEmployeeService	$\supset$
?	Finish	Cancel

3. Tick the check box next to "EmployeeService" and "EmployeeService\_interface\_and\_maps", and click the "Build and Save" button to create the bar file:

Deployable Resources		Build and Save
Select an application to package all it applications.	ts contained resources. Resources within an application are isolated from ot	her
• Applications, shared libraries, ser	vices, and REST APIs O Message flows, static libraries and other message	e flow dependencies
Text filter: type filter text		
EngloyeeService		
EmployeeService_JS	SONClient	
⊡… 🗹 🌺 Shared Libraries ⊕… 🔽 💐 EmployeeService in	terface and maps	

4. Click OK on the window which says "Operation completed successfully".

5. Click the Manage tab to see the resources built to the bar file. You will see the subflow that implements the getEmployee operation under the Resources folder.

🛿 🙀 🙀 Filter by: <type filter="" text=""></type>		
ame	Туре	Modified
EmployeeService	Service	04-Jan-2016 09:24:24
Service Description	Service descriptor	04-Jan-2016 09:24:24
🖃 🗁 Resources		
EmployeeService_inputMessage.xml	XML file	04-Jan-2016 09:24:24
EmployeeService.msgflow	Message flow	04-Jan-2016 09:24:24
EmployeeServiceInputCatchHandler.subflow	Subflow	04-Jan-2016 09:24:24
EmployeeServiceInputFailureHandler.subflow	Subflow	04-Jan-2016 09:24:24
EmployeeServiceInputHTTPTimeoutHandler.subflow	Subflow	04-Jan-2016 09:24:24
getEmployee_Request_Response.subflow	Subflow	04-Jan-2016 09:24:24
EmployeeService_interface_and_maps	Shared Library	04-Jan-2016 09:24:24

6. Deploy the bar file ManualBuildEmployeeService to the **PROVIDER** Integration Server.

## 3.1.2 Note the Service Query URL for the provider

The EmployeeService is now running and can be called. However in order to call the service you need to know the URL and port that the service is running on. You can find this information from the properties tab associated with the running EmployeeService.

1. Using the Toolkit, note the "Service Query URL" port that the EmployeeService is running on: Click the EmployeeService in the Integration Nodes view and look at the properties for the running service:

🔲 Properties 🛛 🔝 Problems 🗄	🖻 Outline 🧟 Tasks 🔠 Deployment Log		
Property	Value		
🗖 Info			
Deployment Time	Sun May 17 10:03:56 BST 2015		
Full Name	EmployeeService.appzip		
Last Modified	Sun May 17 09:53:26 BST 2015		
Operations	getEmployee,updEmployee		
Service Query URL	http://192.168.59.215:7800/EmployeeService?wsdl		
Service URL	http://192.168.59.215:7800/EmployeeService		
<ul> <li>Keywords</li> </ul>			
BAR	C:/workspaces/WSSEC/EmployeeService/ManualBuildEmployeeService.bar		
VERSION			

This is required to ensure the requests made by EmployeeService\_JSONClient are made to the correct endpoint.

Note the IP address and port number details for your environment here:

You will use these details in the next section to configure the SOAP Request node used to call the EmployeeService from the JSON Client.

2. The IP address will be used in order to see communications between the Consumer and Provider application in the TCP/IP Monitor.

You will now set up the JSON Client application on the IB10NODEC.CONSUMER integration node.

## 3.2 Prepare the Consumer

- 1. In the Integration Toolkit, expand EmployeeService\_JSONClient > message flows
- 2. Open the message flow EmpServ\_JSON\_getEmployee.
- 3. Double click on the subflow called **getEmployee\_EmployeeService** to open it in the Integration Toolkit subflow editor:



4. Highlight the Soap Request Node called "Request" and click the properties tab.

On the **HTTP Transport** tab, set the port of the SOAP Request to 7802. This means that the request will be directed to the TCP/IP Monitor, which will in turn forward the request to 7800.

EmpServ_JSON_getEmployee	e.msgflow	eeService_EmpServ_JSON_getEmployee.subflow 🕱	-
👌 😳 Palette	Flow Exerciser: 🔟 🚰 况 📔 🍭		
	]		
Ravorites			
R WebSphere MQ			
🖓 MQTT	in Request		fault
🕞 JMS			<b>→</b> ⊳
💭 HTTP		Evtract	failure
🧟 Web Services		Exodet	
C. SCA			
🐻 WebSphere Adapters			getEmployeeResponse
🕞 Routing			2
Graph User Defined Properties			
🔲 Descritica 🕅 💽 Deschlarer	- 🖳 Outline 🗇 Tesles 📼 Desleveres	at Law ELTCD/ID Manitan	<b>.</b> / V 🗆
	s 📴 Outline 🚈 Tasks 🏢 Deployme	nt Log 🔹 I CP/IP Monitor	
SOAP Request Node P	roperties - Request		
Description	-	$\frown$	
Basic	Web service LIPL*	http://localhost:7802/EmployeeService	
HTTP Transport	Web service one		
IMS Transport	Bernet Knowl (seconds)	e.g. m.p.//server/paul/tu/service	
Message Delivery	Request timeout (in seconds)	120	
Transactions	HTTP(S) proxy location	<enter (if="" any)="" proxy="" server="" your=""></enter>	
Advanced	Protocol (if using SSL)	πs	
WS Extensions	Allowed SSL ciphers (if using SSL)	<enter any="" ciphers="" specific="" ssl="" to="" use="" wish="" you=""></enter>	
Response Message Parsing			
Parser Options	Use compression		
	· · · · · · · · · · · · · · · · · · ·		

- 5. Save the changes to the subflow (Ctrl-S) and close the subflow editor.
- 6. Close the message flow editor (there should be no changes to the message flow).

### 3.2.1 Create JSON Client bar file

You will now create and build a bar file to deploy the JSON Client application. This will be used to configure the authentication and encryption scenarios later in this guide.

1. In the Integration Toolkit, right click on the **EmployeeService\_JSONClient** application and select "New > Bar File" from this list of options.

The "Create a new bar file" editor will open.

Call the bar file "ManualBuildEmployeeService\_JSONClient".

Save the new bar file as part of the EmployeeService\_JSONClient application:

🛞 New BA	R file	
Create a r Create a n	new BAR file ew BAR file resource	
Container:	EmployeeService_JSONClient	New
Folder:	<default></default>	Browse
Name:	ManualBuildEmployeeService_JSONClient	>
?	Finish	Cancel

 In the Prepare tab, select the both the "EmployeeService\_JSONClient" and EmployeeService\_interface\_and\_maps. Click the "Build and Save" button:

elect deployable resources to i	nclude in the BAR	🗄 Build and Save
Select an application to package all its applications.	contained resources. Resources within an applicatio	on are isolated from other
Applications, shared libraries, servi Text filter: type filter text	ces, and REST APIs <b>O</b> Message flows, static librar	ries and other message flow dependencies
Prices     Price	NClient rface_and_maps	

3. Click the Manage tab to check the resources have been included in the bar file:

📳 ManualBuildEmployeeService.bar 📧 EmpServ\_JSON\_getEmployee.msgflow 🛛 📳 ManualBuildEmployeeService\_JS

#### Manage

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

ame	Туре	Modified
🛯 💒 EmployeeService_interface_and_maps	Shared Library	04-Jan-2016 09:53:06
EmployeeService_JSONClient	Application	04-Jan-2016 09:53:06
application.descriptor	04-Jan-2016 09	04-Jan-2016 09:53:00
EmpServ_JSON_getEmployee_inputMessage.xml	XML file	04-Jan-2016 09:53:00
EmpServ_JSON_getEmployee_JSON_to_SOAP.map	MAP file	04-Jan-2016 09:53:00
EmpServ_JSON_getEmployee_XML_to_JSON.map	MAP file	04-Jan-2016 09:53:00
EmpServ_JSON_getEmployee.msgflow	Message flow	04-Jan-2016 09:53:00
⊞ EmpServ_JSON_getEmployee		
Image:	Subflow	04-Jan-2016 09:53:00
Command for packaging the BAR contents		

- 4. Close the bar file editor.
- 5. Deploy the bar file ManualBuildEmployeeService\_JSONClient to the CONSUMER Integration Server.

# 4. Set up IIB to use a Key and Trust store

For both the Authentication and Encryption scenarios, X.509 certificates will be used in the configuration. You will now configure the Provider and Consumer IIB nodes to use a predefined key store and trust store.

Note: at the end of this section, the nodes will be configured to be capable of using a trust store and key store. However the data in the key store and trust store will not be used until a Policy Set and corresponding Policy Set Binding have been configured for the provider and consumer.

1. In an IIB V10 Integration console, change directory to

#### c:\student10\WSSecurity\commands

Run the command:

#### 01SetupPKI.cmd

- 2. Accept the defaults when prompted.
- 3. The script configures the PKI infrastructure used by the two nodes in this lab guide. For the changes to take effect the nodes are restarted by the script.

Wait until you see "Script 01SetupPKI Complete".

## 5. Scenario: Test the service (no active security)

The purpose of this section is to ensure that the JSON Client application can communicate with the EmployeeService provider without any security or encryption defined.

- 1. Open the EmpServ\_JSON\_getEmployee message flow and click the record icon (red button).
- 2. When prompted, select the **CONSUMER** Integration Server on **IB10NODEC**, and deploy the application.
- 3. Click the send message icon. Highlight the Input message "Get employee 000010" and click the send button:

🕀 Send Message	×
Send Message Create or select a message to send more information.	to the flow. Click the message category header (e.g. Input Messages) for
Input Messages Get employee 000010	Name       Get employee 0000 10         Main       Input Location:         Input Location:       HTTP Input         Message Details       Edit, type, or import a message.         Import from file       ["empNumber":"0000 10"]         ["empNumber":"0000 10"]       ["empNumber":"0000 10"]         Show in hexadecimal viewer (Read Only)       ["evert"]
?	Send Close

4. The EmployeeService will respond with the details for "Christine Haas":



- 5. Close the Progress Information window.
- 6. Switch to the TCP/IP Monitor in the IIB Toolkit.

You will see the two panels (send and receive) have been populated with the data that has been sent and received.

Both messages are in clear text, and you will be able to see the requested key (000010) that was sent (in the left pane), and the data that was returned (right pane). Note the data in the section above shows the data exchange between the EmployeeService\_JSONClient **request** and the **response** provider by EmployeeService. Both request and response messages appear in clear text and provide no authentication or privacy.

To view more data, expand the TCP/IP Monitor page (double-click the tab). You can also move the divider bar between the input and output messages.

🗖 Properties 🔝 Problems 🗄 Outline 🧔 Tasks 🏢 Deployment Lou 🖳 TCP/IP Monitor	n x ) 🛛 🖉 🖄 🖓 🖓
E. localhost: 7802	Time of request: 1:25.11.105 PM
/EmployeeService	Response Time: 1250 ms
	Type: HTTP
Request: localhost: 7802 Size: 210 (479) bytes	sponse: localhost: 7800 e: 984 (1126) bytes Byte
Header: POST /EmployeeService HTTP/1.1 Header:	ader: HTTP/1.1 200 OK
Encoding:   Encoding: Encoding:	oding: <none></none>
<ns1:envelope 1.0"="" ?="" encoding="utf-8" xmlns:ns1="http://schemas.xmlsoap.org/soap/envelope/ 🖉 &lt;&lt;/td&gt;&lt;td&gt;&lt;pre&gt;?xml version="><soapenv:envelope pre="" xmlns:soa<=""></soapenv:envelope></ns1:envelope>	
× .	

7. In the IIB Toolkit, click the red cross to stop the Flow Exercise on the JSON Client.

## 6. Policy sets and bindings in IIB

The remaining sections of this lab guide show how IIB support of web services authentication and encryption, can be used to secure the message exchange between the JSON client and the EmployeeService.

This will involve editing the bar files prefix "ManualBuild" that you created earlier. The current implementation of the Flow Exerciser does not permit being able to run a manually built bar file (with bar file overrides). As a result the rest of the lab guide will use SOAPUI to send a message into the JSON client.

In IIB, policy sets and bindings define and configure WS-Security requirements for SOAPInput, SOAPReply, SOAPRequest, SOAPAsyncRequest, and SOAPAsyncResponse nodes.

A policy set is a container for the WS-Security policy types.

A policy set binding is associated with a policy set and contains information that is environment specific, such as information about keys. In each of the testing scenarios throughout this lab guide, a policy set and corresponding binding file is provided to show the security feature.

In the case of the scenarios used by this lab guide, activity on the following nodes will be affected by the rules in the policy set and binding:

- 1. For the **EmployeeService\_JSONClient**, the SOAPRequest node
- 2. For the **EmployeeService**, the SOAPInput and SOAPReply nodes

# 7. Scenario: Authentication using WS-Security

In this scenario, web services authentication using X.509 certificates will be configured using a Policy Set and Policy Set bindings file provided for this lab guide.

The configuration has been automated using previously exported configuration files to save you time. However you will have the chance to optionally review the configuration required to implement Authentication using X.509 certificates.

## 7.1 Policy set and bindings for Authentication

1. In an Integration console, change directory to:

c:\student10\WSSecurity\commands

Run the command:

#### 02SetupAuthPSB.cmd

Accept the defaults when prompted.

2. When the script completes you will see "02SetupAuthPSB command complete."

The Policy set and Policy set bindings have now been defined as configurable services in the IIB nodes. In the next section you will configure both the EmployeeService (service) and EmployeeService\_JSONClient (application) to use the respective policy set and policy set bindings that you added in this section.

## 7.2 Activating a Policy Set and Policy Set Binding configuration

To configure an application or service to use a policy set and corresponding policy set binding file, an IIB administrator specifies the names of the policy sets and bindings in the Bar file.

You will now configure EmployeeService and EmployeeService\_JSONClient to use the policy set containing authentication configuration.

It is possible to specify a policy set and bindings rule either at the message flow level or the individual SOAP node level. In the following section you will specify the policy set and binding at the message flow level.

This is a one-time only configuration as the same policy set and policy set binding file is used for each scenario. Note if you decide to modify either the application or the service you may need to redo this configuration in the bar file.

### 7.2.1 Activate the Policy Set and Binding on the Consumer

1. Using Integration Toolkit, edit the bar file you created earlier called (double click to open).

#### ManualBuildEmployeeService\_JSONClient.bar

- 2. Click the Manage tab and expand the application until you see the message flow, click on the message flow name, EmpServ\_JSON\_getEmployee.
- 3. In the properties tab for the message flow, use the edit button to configure the Consumer Policy Set and Policy Set Bindings as follows:

📳 ManualBuildEmployeeSe	rvice.bar 🛛 🖽 EmpServ_JSON_getEmployee.ms	sgf [ 📳 *Manu	alBuildEmployeeService	 ເມຣ ເຊ		
Manage						
Rebuild remove edit	add resources to BAR and configure their (	properties				
,,,,,						
🔡 🔊 📓 🔮 🛛 Filter b	y: <pre><type filter="" text=""></type></pre>		•			
Name		Туре	Modified			
EmployeeService_interface_and_maps Shared Library 04-Jan-2016 09:53:06						
EmployeeService_JS	ONClient	Application	04-Jan-2016 09:53:06			
application.desc	riptor	04-Jan-2016 09	04-Jan-2016 09:53:06			
EmpServ_JSON	getEmployee_inputMessage.xml	XML file	04-Jan-2016 09:53:06			
EmpServ_JSON	getEmployee_JSON_to_SOAP.map	MAP file	04-Jan-2016 09:53:06			
EmpServ_JSON	getEmployee_XML_to_JSON.map	MAP file	04-Jan-2016 09:53:06			
E E EmpServ_JSON	_getEmployee.msgflow	Message flow	04-Jan-2016 09:53:06			
⊞ EmpServ_JS	ON_getEmployee					
E getEmployee_Er	nployeeService_EmpServ_JSON_getEmployee.subflow	Subflow	04-Jan-2016 09:53:06			
🛨 🗉 😑 gen\getEmp	loyee_EmployeeService_EmpServ_JSON_getEmployee					
Command for package	ing the BAR contents					
, command for packag						
Prepare Manage User Log S	ervice Loa					
🗖 Desceties 🕅 🔍 Deschi	ener 🖳 Outline 🛱 Teche 🥅 Declaument Lee	-		<b></b> \[\textstyle \]		
Properties 23 Trobi	ems 📴 Outline 🚈 Tasks 🌐 Deployment Log	9				
EmpServ_JSON_getEmp	oloyee.msgflow					
Configure	(i) Configure properties of selected bailt resource.					
Workload Management				1		
Detaile	Consumer Policy Set myConsumerPolicy	Set		Edit		
Details	Consumer Policy Set Bindings myConsumerPolicy	SetBinding		Edit		
	Coordinated Transaction					
	Monitoring Profile Name					
	Provider Policy Set			Edit		
	Provider Policy Set Bindings			Edit		
	Security Profile Name			•		

- 4. Save the bar file.
- 5. Redeploy the bar file to the CONSUMER Integration Server.

### 7.2.2 Activate the Policy Set and Binding on the Provider

- 1. Using Integration Toolkit, double click (to edit) the ManualBuildEmployeeService.bar file.
- 2. Click the Manage tab and expand the EmployeeService until you see the message flow **EmployeeService.msgflow**. Click on the message flow name.
- 3. In the properties tab for the message flow, configure the consumer policy and policy set bindings as follows (the edit button will provide the valid names):

[ \*ManualBuildEmployeeService.bar 🕱 💷 EmpServ\_JSON\_getEmployee.msg... 🛛 📳 ManualBuildEmployeeService\_JS...

#### Manage

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

Name		Type	Modified	Size
S EmployeeService		Service	04-Jan-2016 09:24:24	940
Service Descrip	tion	Service descriptor	04-Jan-2016 09:24:24	160
E 🗁 Resources				
X Employees	ervice_inputMessage.xml	XML file	04-Jan-2016 09:24:24	57
🛨 📲 EmployeeS	ervice.msgflow	Message flow	04-Jan-2016 09:24:24	118
🗉 🗉 EmployeeS	erviceInputCatchHandler.subflow	Subflow	04-Jan-2016 09:24:24	54
Image: Image	erviceInputFailureHandler.subflow	Subflow	04-Jan-2016 09:24:24	54
🗉 🗉 EmployeeS	erviceInputHTTPTimeoutHandler.subflow	Subflow	04-Jan-2016 09:24:24	55
🗉 🗉 getEmploye	ee_Request_Response.subflow	Subflow	04-Jan-2016 09:24:24	73
🛛 💒 EmployeeService_ir	nterface_and_maps	Shared Library	04-Jan-2016 09:24:24	1255
			1	
pare Manage User Log   Properties 없 🔐 Prob	Service Log	oyment Log		⊽
Properties 23 Prob	Service Log Iems 🗄 Outline 🖉 Tasks 🏢 Depl Flow	oyment Log		e* >
nare Manage User Log roperties & 🔐 Prob ployeeService.msg1	Service Log Iems 🗄 Outline 🖉 Tasks 🏢 Depl Flow	oyment Log It resource.		₫ ⊽
roperties 🛛 🔝 Prob ployeeService.msgf figure kload Management	Service Log Iems 📲 Outline 🕢 Tasks 🎹 Depl Flow I 🛈 Configure properties of selected bui Consumer Policy Set	oyment Log It resource.		Edit
roperties 🛛 🔝 Prob ployeeService.msgl nfigure kload Management ails	Service Log Iems 🗄 Outline 🕢 Tasks 🌐 Depl Flow 1 Configure properties of selected bui Consumer Policy Set Consumer Policy Set Bindings	oyment Log It resource,		Edit
Interface in the second	Service Log Iems 🗄 Outline 🕢 Tasks 🏛 Depl Flow Configure properties of selected bui Consumer Policy Set Consumer Policy Set Bindings Coordinated Transaction	oyment Log It resource.		Edit
Properties & R Prob Properties & Prob ployeeService.msgl nfigure rkload Management alls	Service Log Iems 🗄 Outline 🖉 Tasks 🖽 Depl Flow Consumer Policy Set Consumer Policy Set Bindings Coordinated Transaction Monitoring Profile Name	oyment Log It resource.		Edit
Interface in the second	Service Log Iems 🔮 Outline 🕢 Tasks 🎛 Depl Flow Consumer Policy Set Consumer Policy Set Bindings Coordinated Transaction Monitoring Profile Name Provider Policy Set Mypro	oyment Log It resource. viderPolicySet		Edit
Properties 23 🔐 Prob ployeeService.msg1 nfigure rkload Management ails	Service Log Iems 📴 Outline 🖉 Tasks 📰 Depl Flow Consumer Policy Set Consumer Policy Set Bindings Coordinated Transaction Monitoring Profile Name Provider Policy Set Bindings myPro Provider Policy Set Bindings myPro	oyment Log It resource. widerPolicySet		Edit Edit Edit

#### (NB take care not to specify the Provider policy sets on the Consumer fields)

- 4. Save the bar file.
- 5. Close the bar file editor.
- 6. Redeploy the bar file to the PROVIDER Integration Server.

## 7.3 Test the service: Authentication defined and active

The purpose of this section is to show that the JSON Client can communicate with the EmployeeService with authentication defined, with each environment knowing authentication details for the other.

### 7.3.1 Find the HTTPConnector port number on the CONSUMER server

In order to invoke the JSON Client that we are using as a Consumer, we need to identify the port number used by the HTTP Connector on the server CONSUMER (on the IIB node IB10NODEC).

1. In an Integration Console window, type the following command:

```
mqsireportproperties IB10NODEC
-e CONSUMER
-o HTTPConnector
-a
```

2. The response of the command will be similar to the following (the port number in your environment may be different – results are truncated)

```
HTTPConnector
uuid='HTTPConnector'
userTraceLevel='none'
traceLevel='none'
userTraceFilter='none'
traceFilter='none'
port='7801'
address=''
maxPostSize=''
acceptCount=''
compressableMimeTypes=''
connectionLinger=''
connectionTimeout=''
```

Note the value of the port number (probably 7801 in the workshop system

You will need this value when you use SOAPUI to start the JSON Client.

)

## 7.3.2 Reset the TCP/IP Monitor

1. In the TCP/IP Monitor window, click the Clear button (highlighted below).

Properties 🖳 Problems	E Outline	Tasks	📰 Deployment Log	📱 TCP/IP Monitor 🟻		
⊡ 着 localhost:7802				↓ Time of request: Response Time: Type: HTTP	2 2 13 147 PM 1:441 13 147 PM Clear 454 ms	⊳

This will clear the messages written during the current live capture.

## 7.3.3 Send a message using SOAPUI

- 1. Open SOAPUI from the Windows Start menu and expand :
  - EmployeeService\_JSONClient Sync
    - http://localhost:7801 (since the target endpoint is most likely to be 7801)
      - getEmployee
        - Employee 000010

Make sure the port on the "Endpoint" window matches the port in your environment (edit it if it doesn't) and click the green arrow to send the message:

<b>&gt;</b>	50apUI 5.0.0		
Eile	e <u>T</u> ools <u>D</u> esktop <u>H</u> elp		
٩	ଷ 🖻 🎄 🔷 💥 🐻		
Navigator	<ul> <li>EmployeeService_PrebuiltWorkspace</li> <li>EmployeeService_PrebuiltWorkspace</li> <li>EmployeeService_REST</li> <li>EmployeeService - MessageFlowSecurity (E</li> <li>EmployeeService - MessageFlowSecurity (3)</li> <li>EmployeeService_JSONClient - Async</li> <li>EmployeeService_JSONClient - Check Bonus</li> <li>Mttp://localhost:7801</li> <li>Mttp://localhost:7801</li> <li>EmployeeService_JSONMQClient - Sync</li> <li>\$</li></ul>	Raw Request	
			Media Type application/json  Post QueryS

2. After a few seconds the window will update with the response from the database:

Click the JSON tab to see the data in JSON format:



## 7.3.4 Verify the network traffic using TCP/IP Monitor

1. Switch to the TCP/IP Monitor view.

The data shown on both the request and response messages will be shown.

🚍 Properties 🕵 Problems 🗄 Outline 🧔 Tasks 🏢 Deployment Log 🖳 TCP/IP Monitor 😒	1
□ - B localhost:7802	Time of request: 1:51.28.631 PM Response Time: 3328 ms Type: HTTP
Request: localhost: 7802         Byte         Response: localhost: 7800           Size: 2004 (2274) bytes         Size: 2682 (2825) bytes           Header: POST /EmployeeService HTTP/1.1         Header: HTTP/1.1 200 OK           Encoding:	Byte
<pre></pre> <pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre></pre><pre><pre></pre></pre></pre>	?> <soapenv:envelope xmlns:s<br="">JjW3FXzZw/ARUYracp/htbe/Ntn sDeleted&gt;<sqlcode_errorcode< td=""></sqlcode_errorcode<></soapenv:envelope>

2. Ensure that the data is displayed in XML format (dropdown selection).

Note the Soap envelope now has a "wsse:security" section with a "BinarySecurityToken", on the request header.

The value of this token is derived from the X.509 certificate passed by the consumer and is used by the provider IIB to authenticate the request from the consumer. (The end tag of the Binary Security Token will be shown if you use the slide bar to display further parts of the message).

IIB trusts this request and allows it to progress as the public part of the consumer's X.509 certificate is stored in the trust store used by the PROVIDER Integration Server.

Note that the payload of the message is in clear text.

Request: localhost: 7802 Size: 2004 (2274) bytes	
header: POST/Employeeservice hTTP/1.1	
xml version="1.0" encoding="UTF-8" standalone="no"?	-
<ns1:envelope xmlns:ns1="http://schemas.xmlsoap.org/soap/envelope/"></ns1:envelope>	
<pre><soapeny:header docs.assis.onen.ong="" http:="" td="" w<="" xmlns:soapeny="http://schemas.xmlsoap.org/soap/envelope/&lt;/pre&gt;&lt;/td&gt;&lt;td&gt;/&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;pre&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;west Ringer Security Taken Encoding Type-"><td></td></soapeny:header></pre>	
lo K see Dinarysecul Lyroken Encounty (see an encounty and see	
I9aKoGA8J/gh/stQtD/GKsSm2aQc226gnJ88xaJ/q5XInee9MYEEsRbyVYPrIIgSmguIKA4EnY	
<ns1:body></ns1:body>	
<io:getemployee xmlns:io="http://EmployeeService"></io:getemployee>	
<pre><employeenumber>000010</employeenumber></pre>	
(io;getEmployee)	
() 1. Becking toyee	

3. Now look at the response message in the right-hand pane. Ensure the format is XML.

This message is the response back to the consumer from the provider.

Note the soapenv:Header also contains a (different) BinarySecurityToken derived from the provider's X.509 certificate. On the provider machine this too has been used to authenticate the response from the provider by verifying the token with the contents of the provider's trust store:



The consumer JSON Client application and the provider EmployeeService now authenticate when communicating. The remainder of this lab guide will guide you through two encryption scenarios.

# 8. Message Level Encryption

The previous scenario showed you how to authenticate web services requests (and responses) using X.509 certificates.

As you can see from the captured data in the TCP/IP Monitor, the request message payload is "in the clear". The data on the network link can be viewed, for both the request and response messages.

An option to secure the exchange of data between two web services is to encrypt the data. With IBM Integration Bus this can be done using whole or partial message encryption. The following sections show how each of these scenarios can be used to secure the data being exchanged between two web services.

The configuration required to use encryption will require changes to the policy sets and bindings configuration used on both the CONSUMER and PROVIDER Integration Servers. The bar file configuration performed in the previous section will continue to be used so no changes will be required to the bar file in both scenarios.

## 8.1 Prepare the PKI for encryption

For *authentication*, IIB requires the public/private key that is used for authentication purposes to exist in the keystore used by the Integration Server.

For example, in the case where the provider sends a response back to the consumer, the key used by the provider to authenticate with the consumer must exist in the provider **keystore**. When the key is passed over to the consumer machine, the consumer integration server checks for the existence of the public portion of the key in the **truststore** used by the consumer integration server. If the key is trusted then authentication succeeds and the request flows to the consumer.

For **Encryption**, the requirement for the placement of keys is slightly different. When encrypting data, the public key of the party which will be decrypting the data is required in the **keystore** of the party encrypting the data.

For example, in the case where the provider sends an encrypted response back to the consumer, the consumer's public key needs to exist in the keystore used by the provider.

The following scripts will prepare the PKI environment you will use for both encryption scenarios in this lab guide.

1. In an Integration console, change directory to:

#### c:\student10\WSSecurity\commands

Run the command:

#### 03SetupPKIForEncryption.cmd

Accept the defaults when prompted.

2. When the script completes you will see "03SetupPKIForEncryption command complete."

You now have the required X.509 certificates in place to perform the encryption scenarios.

## 8.2 Whole Message Encryption

Full message encryption is possible on the request and response messages between a client and a web services provider. You will now configure your environment to **encrypt** both the request from the JSON Client and the response from the EmployeeService.

**Note:** encrypting the whole message exchange between the consumer and provider, although secure, is the more costly in terms of the volume of data being exchanged and the processing power/time required to encrypt the data.

### 8.2.1 Policy Set and Bindings for Whole Message Encryption

1. In an Integration console, navigate to:

c:\student10\WSSecurity\commands.

Run the command:

#### 04PSBSetupEncryptWholeMsg.cmd

Accept the defaults when prompted.

- 2. When the script completes you will see "04PSBSetupEncryptWholeMsg command complete."
- 3. Note a restart of the node is required in order for the changes to take effect on IIB nodes. This is done automatically by the script.

The Policy set and Policy set bindings have now been defined as configurable services in the IIB nodes. In the next section you will configure both the EmployeeService (service) and EmployeeService\_JSONClient (application) to use the respective policy set and policy set bindings that you added in this section.

### 8.2.2 Reset the TCP/IP Monitor

1. In the TCP/IP Monitor view, click the Clear button.

Properties	Problems	Dutline	Tasks	🔝 Deployment Log	📱 TCP/IP Monitor 🛛	JO 🖉 🛃	l
						Time of request: Response Time: Type: HTTP	
Request:				XML	Response:		

This will clear the messages written during the current live capture.

2. If not already started, start the IIB Event Log monitor from the start menu.

### 8.2.3 Send a message using SOAPUI

1. In the SOAPUI window click the green arrow to send the message:

RE ST	Employee 0000	)10					
	Method	Endpoint				Resource	Ś
C	■ '≡ POST	http://loca	lhost:7801			<ul> <li>/empServClient_getEmplo</li> </ul>	yee
lest	t_ ×_ unit (					<pre>&lt;data contentType="te&lt;br&gt;&lt;</pre>	ext/xm
Reg	Name	Value	Style	Level		 	}
M					N N	Ŕ	Į
Å					E		Ċ
					Raw	ND Y	ļ
h	Lann	w	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	m		the man	mm

The request will fail.

 In the IIB Event Log monitor (and in the SOAPUI output) you will see a number of Error messages. The crux of the Failure is an Illegal Key size error. Look for a Java security error message similar to the following:

> org.apache.axis2.AxisFault: CWWSS5612E: Encrypting the data produced the following exception: Illegal key size or default parameters: java.security.InvalidKeyException: Illegal key size or default parameters

When creating the "self-signed" certificates provided with this lab guide, a (default) key size of 1024 was used for both consumercert and providercert.

Due to export restrictions the IBM JDK's ship with a set of restricted policy files that limit the size of the cryptographic keys that are supported.

It is possible to remove this restriction by downloading the unrestricted policy files from the web. The policy files have been downloaded for your convenience and a scripts are provided that will replace the restricted policy files with "Unrestricted" policy files in the consumer and provider environments.

If you are using the lab in your own environment, you can obtain the unrestricted policy files from the URL: https://www-01.ibm.com/marketing/iwm/iwm/web/preLogin.do?source=jcesdk

### 8.2.4 Replace IBM JDK restricted policy files

1. In an IIB V10 Integration console, navigate to c:\student10\WSSecurity\commands, run the command:

#### 05SetJCEForEncryption.cmd

2. Accept the defaults when prompted.

For the Fix Pack number, provide the appropriate fixpack (eg. 3).

3. The script replaces the default IBM JDK **restricted** policy files with **unrestricted** policy files on the provider and consumer IIB nodes.

This affects the whole of this installation not just a specific IIB node.

The IB10NODEP and IB10NODEC nodes are stopped and restarted to bring the changes into effect.

4. When the script completes you will see "05SetJCEForEncryption Complete, ready to run scenario again."

### 8.2.5 Reset the TCP/IP Monitor

1. In the TCP/IP Monitor view, click the Clear button.

Properties	📳 Problems	🗄 Outline	🧟 Tasks	🔲 Deployment Log	📱 TCP/IP Monitor 🔀		- 📭 🖉 t	-
						Ti Re Ty	me of request: esponse mme: rpe: HTTP	
Request:				XML	Response:			

This will clear the messages written during the current live capture.

## 8.2.6 Send a message using SOAPUI

1. In SOAPUI, click the green arrow to send the message again. The port should be the same port as used earlier.

ST Employee 000010			
A Method Endpoint			Resource
POST http://loca	ilhost:7801		<ul> <li>/empServClient_getEmployee</li> </ul>
to ta 🛬 📲 🖸 🗸 🔺			data contentType="text/xm
Name Value	Style L	Level	
(aw)		St.	Į
		HTM	
		Mar Mar	í

2. The request should now respond successfully (click the "JSON" tab to see the response data).



## 8.2.7 Verify the Encryption in TCP/IP Monitor

1. Switch to the TCP/IP Monitor.

Make sure the format is XML. You will see the SOAP body with the encrypted request (wrapped in "CipherValue"):

Request: localhost:7802
SIZE: 30/5 (SAYS) DVTES
Header: PUST /EmployeeService HTTP/1.1
xml version="1.0" encoding="UTF-8" standalone="no"?
<ns1:envelope xmlns:ns1="http://schemas.xmlsoap.org/soap/envelope/"></ns1:envelope>
<soapenv:header xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"></soapenv:header>
<pre><wsse:security 01="" 2004="" docs.oasis-open.org="" http:="" oasis-200401-wss-soap-me<="" pre="" soapenv:mustunderstand="1" wss="" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401&lt;/pre&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;pre&gt;&lt;wsse:BinarySecurityToken EncodingType="></wsse:security></pre>
l9aKoGA8J/gn7stQfD/GKsSm2aQcz26gnJ88xaj/q5Xlhee9MYEEsRbyVYPrlIgSmgulKA4EhYM4KfA7SXctIHKUwIDAQABMA0GCSqGSIb3DQ
<pre><enc:encryptedkey xmlns:enc="http://www.w3.org/2001/04/xmlenc#"></enc:encryptedkey></pre>
<pre><enc:encryptionmethod algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"></enc:encryptionmethod></pre>
<ds:digestmethod 09="" 2000="" algorithm="http://www.w3.org/2000/09/xmldsig#sha1" http:="" www.w3.org="" xmldsig#"="" xmlns:ds="http://www.w3.org/200&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/enc:EncryptionMethod&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;ds:KeyInfo xmlns:ds="></ds:digestmethod>
<wsse:securitytokenreference></wsse:securitytokenreference>
<wsse:keyidentifier #wssecurity_encryption_id_21"="" encodingtype="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-me&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/wsse:SecurityTokenReference&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/ds:KeyInfo&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;enc:CipherData&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;enc:CipherValue&gt;PIJ1V821qJAp82PAemrXj00wAIhNF9piEMZD3iDikr9cYW1MKQPgWJJvf4ePKZuNNcTUhXrYqt7+/18yBE&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/enc:CipherData&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;pre&gt;&lt;enc:ReferenceList&gt;&lt;/pre&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;pre&gt;&lt;enc:DataReference URI="></wsse:keyidentifier>
<pre><enc:encrypteduata la="wssecurity_encryption_la_21" lype="nttp://www.ws.org/2001/04/xmienc#content" pre="" xmins<=""></enc:encrypteduata></pre>
Cency of the Party in the Party is a region of the arth 17 Mark with org/2000 / Wal Vinter Hars/2000 / Dr. / >
<pre><pre>concidentering</pre></pre>
<pre></pre>
//encites/encided/
<t< td=""></t<>
(mailenvelope)

This is the encrypted request sent from the Consumer machine.

2. In the right pane, you will see the response message which will show "HTTP/1.1 200 OK".

As with the request message, the SOAP message will show the encrypted response (also wrapped in "CipherValue"):

Response	e: localhost: 7800
Size: 459	7 (4740) bytes
Header: H	нттр/1.1 200 ОК
xml</td <td>version="1encoding="UTF-8" standalone="no"?&gt;</td>	version="1encoding="UTF-8" standalone="no"?>
<soape< td=""><td>nv.Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"&gt;</td></soape<>	nv.Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
< 508	penv:Header>
<v fw/onz</v 	, sse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasi ≺wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss m1JaWUD0rdMmByF6R97C/OWo4eL700LOu2Rxv68E/8y7gbTrr170r/80sIad3vgX3+EPbzVcF8N+o01PhP0IDA0ABMA0GCS
	<enc:encryptedkey_xmlns:enc="http: 04="" 2001="" www.w3.org="" xmlenc#"=""></enc:encryptedkey_xmlns:enc="http:>
	<enc:encryptionmethod algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"></enc:encryptionmethod>
	<pre><ds:digestmethod 09="" 2000="" algorithm="http://www.w3.org/2000/09/xmldsig#shal" http:="" www.w3.org="" xmldsig#"="" xmlns:ds="http://www.wi&lt;/pre&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/enc:EncryptionMethod&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;pre&gt;&lt;ds:KeyInfo xmlns:ds="></ds:digestmethod></pre>
	<pre><wsse:securitytokenreference></wsse:securitytokenreference></pre>
	<pre><wsse:keyidentifier #wssecurity_encryption_id_21"="" encodingtype="http://docs.oasis-open.org/wss/2004/01/oasis-200401-ws &lt;/wsse:SecurityTokenReference&gt;&lt;/pre&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/ds:KeyInfo&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;enc:CipherData&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;pre&gt;&lt;enc:CipherValue&gt;of/EPlLq/ByyJstIeU+DJtzdxovRXjes+1ZMFyLRA2NgjRcuaXUZmIWPxk5VCjEr3m6FzdX0Y;&lt;/pre&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/enc:CipherData&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;enc:ReferenceList&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;pre&gt;&lt;enc:DataReference URI="></wsse:keyidentifier></pre>
</td <td>wsse:Security&gt;</td>	wsse:Security>
<td>apenv:Header&gt;</td>	apenv:Header>
<soa< td=""><td>penv:Body&gt;</td></soa<>	penv:Body>
<c><c><c><c><c><c><c><c><c><c><c><c><c>&lt;</c></c></c></c></c></c></c></c></c></c></c></c></c>	nc:EncryptedData Id="wssecurity_encryption_id_21" Type="http://www.w3.org/2001/04/xmlenc#Conte
	<pre><enc:encryptionmethod algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"></enc:encryptionmethod></pre>
	<enc:cipherdata></enc:cipherdata>
	<pre><enc:ciphervalue>hlkSbTf4K6puXkkIr7w+GyujClpszlmt9TRbb+KbJJCDvIBkzWq5+7A0LgOuWcKevrGasFZaKcB</enc:ciphervalue></pre>
AVzLL2	!Ira&t4mkwPtcV&LIsMf3QjREVhBCGv0Z66xKPZkurhgJ&LTAKm9vs&4k3E5KaP6Ra4A2PKaVlX/uw7LWS77NYN41jJwU5N
</td <td><pre>'enc:EncryptedData&gt;</pre></td>	<pre>'enc:EncryptedData&gt;</pre>
<td>apenv:Body&gt;</td>	apenv:Body>
<td>env:Envelope&gt;</td>	env:Envelope>

## 8.3 Partial Message encryption

An alternative to encrypting the whole request and response message is to encrypt only a part of the request or response message. In the next section you will see how to encrypt a sensitive part of the response sent back to the consumer.

Partial message encryption is possible on the request and response conversation between two web services. The EmployeeService provides sensitive information back to the requestor. In this next section you will see how the "SALARY" portion of the message can be encrypted leaving the rest of the message untouched.

**Note:** encrypting only sensitive parts of the message exchange between the consumer and provider can provide benefits in terms of less volume of data being exchanged and less processing power/time required to encrypt the data.

Partial message encryption is configured using a policy set and policy set binding. The configuration of these has been automated for your convenience using command line scripts. You will have a chance to review the policy set and policy set binding to see what is required to encrypt part of the response data.

### 8.3.1 Policy Set and Bindings for Whole Message Encryption

1. In an Integration console, navigate to:

```
c:\student10\WSSecurity\commands.
```

Run the command:

#### 06SetupPSBEncryptPartialMsg.cmd

Accept the defaults when prompted.

2. When the script completes you will see "O6SetupPSBEncryptPartialMsg command complete."

The PKI, policy set and policy set binding configuration is now complete to allow for the SALARY field in the response message to be encrypted leaving the rest of the message in clear text. You will now test the partial message encryption scenario.

### 8.3.2 Reset the TCP/IP Monitor

1. In the TCP/IP Monitor window, click the Clear button.

Properties	Problems	🗄 Outline	🖉 Tasks	🔝 Deployment Log	📱 TCP/IP Monitor 🔀	💷 🔎 🖉 🛃
						Time of request: Response Time: Type: HTTP
Request:				XML 💌	Response:	_

This will clear the messages written during the current live capture.

### 8.3.3 Send a message using SOAPUI

1. In SOAPUI, click the green arrow to send the message again.

ST Employee 000010	
Method Endpoint	Resource
POST http://localhost:7801	<ul> <li>/empServClient_getEmployee</li> </ul>
	<pre> data contentType="text/xm2</pre>
Name Value Style Level	
me	

2. As before, the request should respond successfully (click the "JSON" tab to see the response data).

2			
	M	1日	{"EmployeeResponse": {
	Ê	2日	"DBResp": {
	S.	3	"UserReturnCode": 0,
	Ľ,	4	"RowsRetrieved": 1,
	₹	5	"RowsAdded": 0,
	Ē	6	"RowsUpdated": 0,
	2	7	"RowsDeleted": 0,
	Ra	8	"SQLCode_ErrorCode": 0,
	-	9	"SQLState_SQLState": "",
		10	"SQL_Error_Message": ""
		11	},
		12日	"EMPLOYEE": {
		13	"EMPNO": "000010",
		14	"FIRSTNME": "CHRISTINE",
		15	"MIDINIT": "I",
		16	"LASTNAME": "HAAS",

## 8.3.4 Verify the Encryption in the TCP/IP Monitor

1. Switch to the TCP/IP Monitor. Look at the data in the Request pane. Select the XML format.

The header still has the security authentication, however the request is now back to being "in the clear text. In particular, you will see the employeeNumber element, and the value of 000010.

Request: localhost:7802	YMI	
Size: 2004 (2274) bytes	INNE.	
Header: POST /EmployeeService HTTP/1.1		
<pre><?xml version="1.0" encoding="UTF-8" standalone="no"?></pre>		
<pre><ns1:envelope xmlns:ns1="http://schemas.xmlsoap.org/soap/envelope/"></ns1:envelope></pre>		
<pre><soapenv:header xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"></soapenv:header></pre>		
<pre><wsse:security 20<="" docs.oasis-open.org="" http:="" pre="" soapenv:mustunderstand="1" wss="" xmlns:wsse="http://docs.oasis-open&lt;br&gt;&lt;/pre&gt;&lt;/td&gt;&lt;td&gt;en.org/wss/:&lt;/td&gt;&lt;td&gt;í.&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;pre&gt;&lt;wsse:BinarySecurityToken EncodingType="></wsse:security></pre>	004/01/oasi	:
19aKoGA8J/gn7stQfD/GKsSm2aQcz26gnJ88xaj/q5X1hee9MYEEsRbyVYPr1IgSmgu1KA4EhYM4Kf/	A7SXctIHKUw	
<pre><ns1:body></ns1:body></pre>		
<io:getemployee xmlns:io="http://EmployeeService"></io:getemployee>		
<pre><employeenumber>000010</employeenumber></pre>		
<pre>//NS1:Envelope&gt;</pre>		

2. Now look at the Response pane. First, you will see that the Provider responded with an HTTP 200 code.

You will see the authentication information at the top of the response message.

Response: localhost: 7800
Size: 4328 (4471) bytes
Header: HTTP/1.1 200 OK
<pre><?xml version="1.0" encoding="UTF-8" standalone="no"?></pre>
<soapenv:envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"> <soapenv:header></soapenv:header></soapenv:envelope>
<pre><wsse:security 01="" 2004="" docs.oasis-open.org="" http:="" oasi:<br="" soapenv:mustunderstand="1" wss="" xmlns:wsse="http://docs.oasis-open.org/wss//&lt;br&gt;&lt;wsse:BinarySecurityToken EncodingType=">fW/ONZm1JaWUD0rdMmByF6R97C/OWo4eL70QLOu2Rxv68E/8y7gbTrr17Qr/80sIad3vgX3+EPbzVcF8N+o01PhPQ1 <enc:encryptedkey xmlns:enc="http://www.w3.org/2001/04/xmlenc#"></enc:encryptedkey></wsse:security></pre>
<pre><enc:encryptionmethod ;<="" algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p" td=""></enc:encryptionmethod></pre>
<ds:keyinfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#"> <wsse:securitytokenreference></wsse:securitytokenreference></ds:keyinfo>
<pre><wsse:keyidentifier #wssecurity_encryption_id_21"="" encodingtype="http://docs.oasis-open.org/wss/2004/01/oasis&lt;br&gt;&lt;/wsse:SecurityTokenReference&gt;&lt;/pre&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/ds:KeyInfo&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;pre&gt;&lt;enc:CipherValue&gt;jGCOBlFMkXFzfQX3adCAsitJUlo/KzOboJSWgzJ5VnTf8YX9X9fK7YsK0rMz5/) &lt;/pre&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/enc:CipherData&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;pre&gt;&lt;enc:ReferenceList&gt;&lt;/pre&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;pre&gt;&lt;enc:DataReference URI="></wsse:keyidentifier></pre>

Further down the response message, you will see the SOAP Body. You will see that most of the message is in clear text. However, the SALARY element (positioned between BIRTHDATE and BONUS) no longer appears as a readable field. The configuration of the XPATH statement in the Policy Set used by the message flow has encrypted (just) the SALARY field:



# 9. Transport Level Security (TLS)

Encryption support in WS-Security concentrates on providing encryption for the content of the SOAP message exchange between a consumer and provider. The http request from the JSON client (consumer) to the EmployeeService (provider) is a standard non secure http request. You will now secure the provider by configuring the EmployeeService to use https and change the consumer request configuration to use an authenticated TLS request.

## 9.1.1 Configure the provider to use https

Perform the following tasks on the PROVIDER machine:

1. In an IIB V10 Integration Console, navigate to c:\student10\WSSecurity\commands, run the command

#### 07SetupHTTPS.cmd.

- 2. Accept the defaults when prompted.
- 3. The script:
  - a) configures the **HTTPSConnector** defined at the Integration Server level on a specific port (7848)
  - b) sets the "**clientAuth**" property to "true" enabling client authentication based on X.509 certificates.
  - c) stops and starts IB10NODEC for the changes to take effect.

Make sure IB10NODEP starts ok and that the EmployeeService is running.

4. In the IIB Toolkit, expand EmployeeService and double click on the "Integration Service Description":



5. This will open the Service at the "Service" tab:

EmployeeService 🛛	
S EmployeeService >	
S EmployeeService	EmployeeService     getEmployee     updEmployee
	🔀 Error Handlers
	🗏 🗉 Failure
	<sup></sup> <u>■</u> <u>Catch</u>
	<sup>□</sup> <sub>■</sub> <u>Timeout</u>
Service 主 Interface	

Click the service tab to show the SOAP HTTP Binding properties

6. Tick the "Use HTTPS" check box:

Properties 🛛 🛛	🚼 Problems 🗄 Outline 🖉 Tasks 🖽 Deployment Log			
SOAP/HTTP Binding				
Basic				
HTTP Transport				
Advanced	Maximum client wait time (sec)* 180			
WS Extensions	Enable support for ?wsdl			
Validation				
Instances				
Monitoring	{			
	L			

- 7. Save the change (Ctrl –s).
- 8. Close the window.
- 9. Open ManualBuildEmployeeService.bar file in the bar file editor.

10. In the bar file editor, on the Prepare tab, Click the Build and Save button to rebuild the bar file.

I ManualBuildEmployeeService.bar ⊠	
Prepare	
Select deployable resources to include in the BAR	
Deployable Resources	Build and Save
Select an application to package all its contained resources. Resources within an application are isolated from other applications.	
• Applications, shared libraries, services, and REST APIs • O Message flows, static libraries and other message flow d	dependencies
Text filter: type filter text	
Image: Services       Image: Service       Image:	<u> </u>
Prepare Manage User Log Service Log	

11. When the bar file has been rebuilt with the changes applied, click the Manage tab.

In the Name column of the table, expand EmployeeService > Resources and click EmployeeService.msgflow:

🕼 ManualBuildEmployeeService.bar 🛛				
Manage				
Rebuild, remove, edit, add resources to BAR and configure their properties				
🔡 📲 😪 😼 Filter by: <a>Type filter text&gt;</a>				
Name	Туре	Modified		
S EmployeeService	Service	20-May-2015 14:53:41		
🗄 📄 Service Description	Service descriptor	20-May-2015 14:53:40		
🗆 🗁 Resources		4		
X EmployeeService_inputMessage.xml	XML file	20-May-2015 14:53:40		
	Message flow	20-May-2015 14:53:40		
EmployeeServiceInputCatchlandler.sub	Subflow	20-May-2015 14:53:40		
EmployeeServiceInputFailureHandler.su	Subflow	20-May-2015 14:53:40		
EmployeeServiceInputHTTPTimeoutHand	Subflow	20-May-2015 14:53:40		
<b>   </b>				
Command for packaging the BAR contents				
Prepare Manage User Log Service Log		<u>ا</u>		

12. In the "Configure" tab, use the Edit button to set the Provider Policy Set and Provider Policy Set Bindings values to myProviderPolicySet and myProviderPolicySetBinding respectively:

🔲 Properties 🔀 🔝 Proble	ms 📴 Outline 🖉 Tasks 🎞 D	eployment Log	1	
EmployeeService.msgflow				
Configure	Configure properties of sel	ected built resource.		1
Workload Management	Consumer Policy Set		Edit	
Details	Consumer Policy Set Bindings		Edit	
	Coordinated Transaction			i
	Monitoring Profile Name			
(	Provider Policy Set	myProviderPolicySet	Edit	
	Provider Policy Set Bindings	myProviderPolicySetBinding	Edit	
	Security Profile Name		•	
				~~~~~

(note: do not specify anything in the Consumer Policy set and Bindings fields)

- 13. Save the bar file (ctrl –s)
- 14. Close the bar file editor.
- 15. Deploy the bar file to IB10NODEP.PROVIDER.
- 16. In the Integration Nodes window, navigate down to the EmployeeService and highlight the service by clicking on it. In the properties tab, make sure the "Service URL" is now using "https" and the port the service uses is "7848":

🖁 🖾 👷 🖓 🖾 🖉 🖉 🖓 🖓	Properties 🛛 🔝 Problems 🗄	🗄 Outline 🖉 Tasks 🔠 Deployment Log		
🚔 🗄	Property	Value		
Integration Nodes	🖃 Info			
IB10NODE	Deployment Time	Wed May 20 15:01:18 BST 2015		
TB10NODE CMO	Full Name	EmployeeService.appzip		
TB10NODE PMO	Last Modified	Wed May 20 14:53:40 BST 2015		
	Operations	getEmployee,updEmployee		
	Service Query URL	https://192.168.59.215:7848/EmployeeService?wsdl		
	Service URL	https://192.168.59.215:7848/EmployeeService		
	<ul> <li>Keywords</li> </ul>			
PROVIDER	BAR	C:/workspaces/WSSEC/EmployeeService/ManualBuildEmployeeService.bar		
EmployeeService	VERSION	{		
	Louis and the second second	Lange and the second		

(EmployeeService is now running on an https connection). Write down (or copy using ctrl c) the Service URL, you will use this in the next section:

Service URL :

## 9.2 Call the Provider using HTTPS and Client authentication

There is no requirement to configure the HTTPSConnector on the consumer side. However you will need to modify the following details on the SOAP Request node in the JSON client:

- a) the Web service URL
- b) the SSL client authentication key alias

Both are on the "HTTP transport" tab on the SOAP Request Node Properties.

- 1. Open the **EmpServ\_JSON\_getEmployee** message flow.
- 2. In the message flow double click on the **getEmployee\_EmployeeService** sub flow.
- 3. In the sub flow, click on the Soap Request node called "Request":

EmpServ_JSON_getEmploye	e.msgflow	x j
Flow Exerciser: 💿 💈		
in	Request Fault	
	geanpoyee	Laponac
Graph User Defined Properties		Ì

4. In the properties tab, click on "HTTP Transport" to show the Web service URL settings.

- 5. Make the following changes:
  - a) Set Web service URL to : https://localhost:7802/EmployeeService
  - b) Set SSL client authentication key alias to : consumercert

Note that the target port, 7802, is owned by the TCP/IP Monitor. This will forward the request on to the ultimate destination, on port 7848.

🔲 Properties 🔀 💽 Problems	; 🗄 Outline 🖉 Tasks 🏢 Deployme	nt Log 📮 TCP/IP Monitor	<b>B</b>	
👏 SOAP Request Node Properties - Request				
Description				
Basic	Web service URL*	https://localhost:7802/EmployeeService		
HTTP Transport		e.g. http://server/path/to/service		
JMS Transport	Request timeout (in seconds)	120		
Message Delivery	HTTP(S) proxy location	<enter (if="" any)="" proxy="" server="" your=""></enter>		
Transactions				
Advanced	Protocol (if using SSL)	πs		
WS Extensions	Allowed SSL ciphers (if using SSL)	<enter any="" ciphers="" specific="" ssl="" to="" use="" wish="" you=""></enter>		
Response Message Parsing	Use compression	none		
Parser Options	· · · · · · · · · · · · · · · · · · ·	Г		
Validation	Accept compressed responses by default			
Monitoring	Enable SSL certificate hostname checking			
	SSL client authentication key alia	consumercert		
	Enable certificate revocation list checking			

- 6. Save the sub flow (ctrl S).
- 7. Close the subflow and message flow editors.
- 8. Open the **"ManualBuild\_EmployeeService-JSONClient**" bar file and click the prepare tab.
- 9. Click the Build and Save button to rebuild the bar file.

Select deployable resources to include i	n the BAR
Deployable Resources	Build and Save.
select an application to package all its contained applications.	resources. Resources within an application are isolated from other
• Applications, shared libraries, services, and R	EST APIs O Message flows, static libraries and other message flow dependencies
Applications, shared libraries, services, and F Text filter: type filter text	REST APIs $$ O Message flows, static libraries and other message flow dependencies
Applications, shared libraries, services, and F Text filter: type filter text	REST APIs O Message flows, static libraries and other message flow dependencies
Applications, shared libraries, services, and F Text filter: type filter text      Text filter: generate service      Definitions      Definity      Definity      Definitions      Definitions      Definiti	LEST APIs O Message flows, static libraries and other message flow dependencies

10. Click the manage tab, navigate down to the message flow called

#### EmpServ\_JSON\_getEmployee.msgflow

Click the message flow name, then click the properties tab.

 In the "Configure" (horizontal) tab, use the Edit button to reset the Consumer Policy Set and Consumer Policy Set Bindings values to myConsumerPolicySet and myConsumerPolicySetBinding respectively:

🔲 Properties 🔀 🔝 Problem	ms 📴 Outline 🖉 Tasks 🥅 De	eployment Log	₹ ~		
EmpServ_JSON_getEmp	EmpServ_JSON_getEmployee.msgflow				
Configure	() Configure properties of sele	cted built resource.			
Workload Management	Consumer Policy Set	myConsumerPolicySet	Edit		
	Consumer Policy Set Bindings	myConsumerPolicySetBinding	Edit		
	Coordinated Transaction				
	Notitoring Profile Name				
	Provider Policy Set		Edit		
	Provider Policy Set Bindings		Edit		
	Security Frome Name				

(do not specifying anything in the Provider Policy settings).

- 12. Save (ctrl-s) and close the bar file.
- 13. Deploy the bar file to IB10NODEC.CONSUMER.

The JSON client is now configured to request the EmployeeService on an https connection and pass "consumercert" as credentials to authenticate with the service. The Policy Set configuration is (still) set to partially secure the message response from the EmployeeService, however the remaining part of the message content response and the request from the JSON client would otherwise be in the clear.

The TLS configuration will ensure the data exchanged between the consumer and provider is not in the clear.

## 9.3 Test TLS scenario

The JSON client is now configured to request the EmployeeService on an https connection and pass "consumercert" as credentials to authenticate with the service. The Policy Set configuration is (still) set to partially secure the message response from the EmployeeService, however the remaining part of the message content response and the request from the JSON client would otherwise be in the clear.

The TLS configuration will ensure the data exchanged between the consumer and provider is not in the clear.

## 9.3.1 Reconfigure the TCP/IP Monitor properties

The connection between the Consumer and Provider is now secured using TLS. This means that the TCP/IP Monitor will not be able to detect that the transmission is using http. To accommodate this change, you will need to reconfigure the TCP/IP Monitor.

1. In the TCP/IP Monitor window, right-click and select Properties.



2. Stop the current Monitor:

Preferences							
····· TCP/IP Monitor	TCP/IP Mor	TCP/IP Monitor					
	Configure TCP	Configure TCP/IP monitors on local and remote ports.					
Show the TCP/IP Monitor view when there is a				tivity			
TCP/IP Monitors:							
	Status	Host name	Туре	Local Port	Auto-start	Add	
	💻 Stopped	localhost:7848	TCP/IP	7802	No	Edit	
						Remove	
					(	Start	
						Stop	

3. Click Add to add a new monitor.

Set the following values:

- Local monitoring port: 7802
- Host name : localhost
- Port : 7848
- Type : TCP/IP

Click OK, then Start the Monitor.

Click OK to close the Properties window.

6	New Monitor	×
ļ	Local monitoring port: 780	)2
	Monitor	
	<u>H</u> ost name:	localhost
	Port:	7848
	<u>T</u> ype:	TCP/IP
	Timeout (in milliseconds):	0
		Start monitor automatically
		OK Cancel

9.3.2 Send a message using SOAPUI

1. In the SOAPUI window click the green arrow to send the message again.

RE ST Employee 000010		
Method E	Endpoint	Resource
	http://localhost:7801	<ul> <li>/empServClient_getEmployee</li> </ul>
to t_ = = "" 🖸 🔹	~ ~	<pre>data contentType="text/xm</pre>
Name \	/alue Style Level	
MP		ST

2. The request should respond successfully (click the "JSON" tab to see the response data).

XML 1 [ {"EmployeeResponse": { 2 🖂 "DBResp": { JSON 3 "UserReturnCode": 0, 4 "RowsRetrieved": 1, HTML 5 "RowsAdded": 0, 6 "RowsUpdated": 0, 7 "RowsDeleted": 0, Raw 8 "SQLCode\_ErrorCode": 0, 9 "SQLState\_SQLState": "", 10 "SQL Error Message": "" 11 }, 12 🖂 "EMPLOYEE": - { "EMPNO": "000010", 13 14 "FIRSTNME": "CHRISTINE", "MIDINIT": "I", 15 16 "LASTNAME": "HAAS", 17 "WORKDEPT": "A00", "PHONENO": "3978", 18 "HIREDATE": "1995-01-01", 19 20 "JOB": "PRES ۰. 21 "EDLEVEL": 18, "SEX": "F", 22 23 "BIRTHDATE": "1963-08-24", 24 "SALARY": 152750, 25 "BONUS": 1000, 26 "COMM": 4220 27 } 28 }}

### 9.3.3 Verify the test in the TCP/IP Monitor

1. Switch to the Monitor.

This time, with TLS, there will be no evidence of the data or protocols that the Consumer and Provider have used to talk to each other. Nothing will be "in the clear".

However, you will see that several thousand bytes have been sent and received.

🔲 Properties  🖳 Problems	E Outline	Tasks	📰 Deployment Log	📱 TCP/IP Monitor 🛛	JO 🖉	≝ ≦ ∼ □
Iocalhost: 7802       Iocalhost: 7848       Iocalhost: 7848					Time of reque Response Tim Type: TCP/IP	st: 2:55.43.454 PM e: 78 ms
, Request: localhost:7802 Size: 4192 (4192) bytes Header: T <sup>L</sup> Encoding: UTF-8		ſ	Byte Resp Size: Head Enco	oonse: localhost: 7848 : 4924 (4924) bytes der: ⊤ <sup>L</sup> ding:	×	Byte 💌
			T T			X



Page 54 of 65

# 10. Appendix

## 10.1 Defining the self signed certificates used in this lab guide

The self-signed certificates used in this lab guide were created using the Open SSL package.

A key and self-signed certificate were first created for a Root Authority called rootCA.

Private keys and certificates (signed by the rootCA) where then created for use by the consumer and provider.

These were then saved as in PKCS12 format before being imported into the JKS key stores used by the IBM Integration Servers.

The following section outlines the commands that were used to create the rootCA and consumer certificates.

The provider certificates were created in the same way as the consumer certificates.

### 10.1.1 Preparing your openssl environment:

- 1. Install OpenSSL. (the details in this section were done on SSL Light downloaded from the openssl.org web site)
- 2. If you are using windows, and you see an error stating that the openssl.cfg file cannot be located, you will need to set an environment variable OPENSSL\_CONF, the variable needs setting to (by default C:\OpenSSL-Win32\bin\openssl.cfg")
- 3. Ensure the OpenSSL application can write to the directory you were using the openssl.exe file in. In windows open a command prompt with the "Run As administrator" option.

### 10.1.2 Create the RootCA:

1. Enter the command to create an RSA 2048 bit key:

This will create a private key called "rootCA.key" in C:\sample which we can now use to base the certificate requests on.

2. Now create the certificate request:

# openssl req -x509 -new -nodes -key c:\sample\rootCA.key -days 2048 -out c:\sample\rootCA.crt

The response to the command does require input, the input in the following is in the larger text:

Loading 'screen' into random state - done You are about to be asked to enter information that will be incorporated into your certificate request. What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few fields but you can leave some blank For some fields there will be a default value, If you enter '.', the field will be left blank. -----Country Name (2 letter code) [AU]: **GB** State or Province Name (full name) [Some-State]: **warwickshire** Locality Name (eg, city) []: **warwick** Organization Name (eg, company) [Internet Widgits Pty Ltd]: **ibm** Organizational Unit Name (eg, section) []:**betaworks** Common Name (e.g. server FQDN or YOUR name) []:**rootCA** Email Address []:

At the end of this command you have a certificate request for the rootCA

### 10.1.3 Create the signed CONSUMER certificate:

1. Enter the command to create an RSA 2048 bit key:

This will create a private key called "consumer.key" in C:\sample which we can now use to base the certificate requests on.

2. Create the certificate request for the consumer

#### openssl req -new -key c:\sample\consumer.key -out c:\sample\consumer.csr

The response to the command does require input, the input in the following is in the larger text

Loading 'screen' into random state - done You are about to be asked to enter information that will be incorporated into your certificate request. What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few fields but you can leave some blank For some fields there will be a default value, If you enter '.', the field will be left blank. \_\_\_\_ Country Name (2 letter code) [AU]: GB State or Province Name (full name) [Some-State]: warwickshire Locality Name (eg, city) []: warwick Organization Name (eg, company) [Internet Widgits Pty Ltd]: **ibm** Organizational Unit Name (eg, section) []: betaworks Common Name (e.g. server FQDN or YOUR name) []: Consumer Email Address []: Please enter the following 'extra' attributes to be sent with your certificate request

An optional company name []:

A challenge password []:

3. Now generate the consumer signed certificate using the rootCA certificate and key:

openssl x509 -req -in c:\sample\consumer.csr -CA c:\sample\rootCA.crt -CAkey c:\sample\rootCA.key -CAcreateserial -days 2048 -out c:\sample\consumer.crt

#### Response:

Loading 'screen' into random state - done Signature ok subject=/C=GB/ST=warwickshire/L=warwick/O=ibm/OU=betaworks/CN=consumer Getting CA Private Key

 Store the key and certificate in PKCS12 format to enable import into JKS key stores used by IBM Integration Bus.

openssl pkcs12 -export -in c:\sample\consumer.crt -inkey c:\sample\consumer.key -out c:\sample\consumerkeystore.p12 -name consumercert

Response:

Loading 'screen' into random state - done

Enter Export Password:

Verifying - Enter Export Password:

Give the PKCS12 format file a password as part of the response otherwise the import will fail when you come to import the key.

#### 10.1.4 Import the keys and certificates:

The following section show how you to import a key saved in PKCS12 format into a JKS key store used by IIB.

1. Using IKEYMAN, open the Consumer key store:

🪺 IBM Key Mana	gement	_ 0
Key Database <u>F</u> il	e <u>C</u> reate <u>V</u> iew <u>H</u> elp	
	Key database information	
DB-Type:		
File Name:		
Token Label:		
	Key database content	
Personal Certif	icates	▼ Receive
		Delete
	Key database type JKS	Vie <u>w</u> /Edit
	Eile Name:         ConsumerKeyStore.jks         Browse	Import
	Location: C:\student10\WSSecurity\Keystores	Recre <u>a</u> te Request
	<u>O</u> K <u>C</u> ancel	Rename

2. When prompted, enter the password specified for the key store:

Password Pro	mpt		×
<u>P</u> assword:	•••••		
<u>0</u>	<u>R</u> es	et <u>C</u> ancel	

3. Once the keystore is open click the Import button (if there are keys already stored in the key store this will also appear as an Import/Export button):

5	
	Import
Ł	Recreate Request.

4. Browse for the consumerkeystore.p12 file that you created in the previous section:

Open		×
Choose Action Typ	0e	
🔘 Export Key 🛛 🧕	Import Key	
<u>K</u> ey file type	PKCS12	
<u>F</u> ile Name:	consumerkeystore.p12	Browse
Location:	cisample	
	<u>O</u> K <u>C</u> ancel	

5. Key in the password you defined for the PKCS12 file:



6. Click "ok" on the "Change Labels" screen:

Change Labels	×
Would you like to change any of these labels before completing the import process?           Select a label to change:           consumercert	<u>QK</u> <u>C</u> ancel
Enter a new label:	

7. The consumercert certificate and key will be imported into the key store:

IBM Key Mana	gement - [Z:\Z\sample\ConsumerKeyStore.jks]	_0,
(ey Database <u>F</u> i	le <u>C</u> reate <u>V</u> iew <u>H</u> elp	
D 🚄		
	Key database information	
DB-Type:	JKS	
File Name:	Z:\Z\sample\ConsumerKeyStore.jks	
Token Label:		
	Key database content	
Personal Certi	ficates 🗸 🗸	Rece <u>i</u> ve
consumercert		Delete
		Vie <u>w</u> /Edit
		Export/Import
		Recreate Request
		Rena <u>m</u> e
		New Self-Signed
		Extract Certificate
he requested a	ction has successfully completed!	

8. Take a look at the certificate, click the View/Edit button when the consumercert is highlighted:

Key information for [con	sumercert]	X
	consumercert	
Key Size: Certificate Properties:	2048	
Version:	X509 V1	
Serial Number:	00 FD 60 9E DD B2 DA E2 19	
Issued to:		
CN=consumer, OU=	betaworks, O=ibm, L=warwick, ST=warwickshire, C=GB	
Issued by: CN=rootCA, OU=bet	aworks, O=ibm, L=warwick, ST=warwickshire, C=GB	
Validity: Fingerprint (SHA1 Dig 5C:EC:73:18:21:7B:F Signature Algorithm:	Valid from 16 January 2015 to 25 August 2020 est): F:14:E4:0D:86:F5:9A:55:C2:B2:0F:5D:51:5D SHA1withRSA (1.2.840.113549.1.1.5)	
Subject Alternative N	ames:	
Email Address:		
IP Address:		
DNC Hamai		

Notice:

- 1) the Distinguished Name defined in the "**Issued to**" field reflect the details you entered when creating the *consumer* certificate request in OpenSSL
- the Distinguished Name defined in the "Issued by" field reflect the details of the certificate Authority (rootCA) which signed the *consumer* certificate request in OpenSSL.
- 9. You can now Extract the Certificate as an ".arm" file enabling you to then "add" it (for example) to trust store used by the provider (enabling an integration server used as a "provider" to trust certificates sent from an application running on the integration server used as a "consumer").

Similarly it is likely that you will need to configure a "consumer" to trust responses back from the provider (in this case you would "Add" the extracted provider certificate *as a signer* to the "consumer" trust store).

The key stores and trust stores used throughout this lab guide were set up and configured using the above techniques.

10. In order to avoid certificate verification chain errors you will also need the rootca certificate defined in the key store. Add (as a "Signer certificate") the rootCA.crt file created in Open SSL to the key store:

Key database information				
DB-Type:	JKS			
File Name:	Z:\Z\sam	ple\ConsumerKeyStore.jks		
Token Label:				
Key database content				
Signer Certifica	ites	•	<u>A</u> dd	
rootca			Delete	
		~	Vie <u>w</u> /Edit	

11. If you look at this certificate using the View/Edit button you will see the details of the Certificate request for the "mock" root certificate authority you specified in "open SSL" above:

Key information for [roo	tca]	×
	rootca	
Key Size:	2048	_
Certificate Properties:		
Version:	X509 V3	
Serial Number:	00 C9 64 7F D0 F5 E8 BE 25	
Issued to:		_
CN=rootCA, OU=be	taworks, O=ibm, L=warwick, ST=warwickshire, C=GB	
Issued by:		
CN=rootCA, OU=be	aworks, O=ibm, L=warwick, ST=warwickshire, C=GB	
Validity:	Valid from 16 January 2015 to 25 August 2020	
Fingerprint (SHA1 Dig	est); 2.47:24:44:25:50:55:52:24:50:20:02:04:50	
F1:5B:0E:2C:T1:5F:0	SHA1:24:14:2F:F9:SF:E2:24:ED:20:03:01:FC SHA1withDSA /1 2 940 113540 1 1 5)	
Subject Alternative N	amoe	
Email Addroses	ames.	
Ellian Audress.		
IP Address:		
DNS Name:		
	ma	

End of Lab Guide