Steve's Exercise Progress Notes - WE751 (ERC1.1) 2018-11-02
Issues:
- problems with Blueprint console (esp. Chrome)

Backend Simulation in *FlyServices* Domain

The FLY airline services are self-contained MPGWs that mimic a web services backend that might be on WAS.
Booking Service web service and a Baggage Service web service.
BookingServiceBackend MPGW and a *BaggageStatusMockService* MPGW, both running within the *FLYServices* domain.
The **Booking Service** has one operation:
- #BookTravel (BookingRequest, BookingResponse),
endpoint: http://<dp_internal_ip>:9080/BookingService/
The **Baggage Service** (BaggageStatusMockService/MPGW, 0.0.0.0:2068) has two operations, both skip-backside:
- #BaggageStatus (BaggageStatusRequest, BaggageStatusResponse)
  XPath = /*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='BaggageStatusRequest']/*[local-name()='refNumber']
- #BagInfo (BagInfoRequest, BagInfoResponse)
  XPath = /*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='BagInfoRequest']/*[local-name()='id']
endpoint: http://<dp_internal_ip>:2068/BaggageService/

Exercise 1 - First exposure to the DataPower developer environment
After completing this exercise, you should be able to:
• Log in to the WebGUI
• Use the navigation bar
• Use an object catalog
• Connect to the Blueprint Console
• Import a service
• Edit a multi-protocol gateway
• Review the actions in a policy editor
• Test a service from a browser and a cURL command
• Export a service

1.1. Initialize the lab environment
  check against Appendix B (correct values not needed until Exercise 2.)
1.2. Work with the WebGUI home page
1.3. Work in the Blueprint Console
  WebGUI / Blueprint explore domain: *FlyServices*.
  From this point forward, you can work in either web interface,
  although you should work primarily in one or the other.
  Import `dp/intro/HelloWorldMPGW.zip`
1.4. Examine and edit a service: HTTPFSH, Policy, Rules
  change <mpgw_helloworld_port>: 12nn7
  MPGW Contains two rules:
    /xsl -> helloxslworld.xsl
    /javascript -> hellojsworld.js
1.5. Test a service
  Test browser:
    `http://192.168.0.101:12007/xsl`
    `http://192.168.0.101:12007/javascript`

View log entries made by:

xsl: `<xsl:message >In the XSL style sheet for HelloWorld`

js: `console.error('In GatewayScript code in HelloWorld')`

Test cURL:

```
curl -G http://192.168.0.101:12007/xsl
curl -G http://192.168.0.101:12007/javascript
```

1.6. Export a configuration

MyUpdatedMPGW.zip

explore content: XSL & JS files, and `<LocalPort>` in export.xml

Exercise 2. Creating a BookingService gateway

*** need SOAPUI set up as Appendix B ***

After completing this exercise, you should be able to:

• Create a multi-protocol gateway

• Test the message flow by using the SoapUI graphical test tool

The service validates the client SOAP message before forwarding to the backend.

2.1. Initialize the lab environment (already done in Exercise 1.)

2.2. Create a basic MPGW to validate SOAP messages

(Chrome does NOT show settings below *User Agent settings*, so unable to set *Response/Request Type*.)

MPGW: *BookingServiceProxy*

Default Backend URL

`http://dp_internal_ip:9080/BookingService` (*** actual host alias - OK ***)

FSH: HTTP_12001 : <mpgw_booking_port> : 12nn1

Section 3: *BookingServiceProxy* MPGW testing

Additional:

The above exercise, using the call **`01- Initial Test`**, calls your MPGW, but only with a message that conform with the SOAP specification. So you do not see the rejection of nonconforming messages.

Rerun the same call, but modifying the message to change the element named Body to Bodi.

What is the result?

Set the Probe on, rerun again, and in the Probe window, inspect what is happening.

Also, the call **`00 - Web Service Test - Booking`**, calls the backend directly, i.e. not through your MPGW.

Run this call, without and with the same modification, and inspect the results.

Exercise 3. Enhancing the *BookingService* gateway

After completing this exercise, you should be able to:

• Perform advanced configuration of an MPGW

• Configure a document processing policy with more actions validation, filtering, and transformation

• Test the MPGW policy by using the graphical SoapUI tool

• Perform basic debugging by using the system log

3.1. Initialize the lab environment (already done in Exercise 1.)

3.2. Add more capability to the *BookingServiceProxy* MPGW

Section 1: Schema Validation

`<BookingType>I/E</BookingType>` **`BookingService.wsdl`**

"I" or "E" allowed enumeration values

Section 2: Schema Validation Test

`<BookingType>`**EXTERNAL**`</BookingType>` **02 – Invalid Booking Type**

Section 3: SOAP Envelope Schema Validation

03 – Missing SOAP Envelope

Section 4: Content-based Filtering

`<ReservervationCode>` **ReservationCode_Filter.xsl**
should start with "JK"

`<ReservationCode>`**ERROOL0D**`<ReservationCode>` **04 - ReservationCode Invalid Test**

The XPath expression used in the given XSL filter is not very safe:
- not specific on element location
- not specific on actual namespace (prefixes in XML might change)

The following would be safer:
starts-with(/*/*/*/*/*[local-name()='ReservationCode'][namespace-uri()='http://www.ibm.com/datapower/FLY/BookingService/'], 'JK')

Section 5: SQL Injection Threat Filtering

Section 6: Transforming with XSL **BookingResponse_Transform.xsl**
(acts on response)
01 – Initial Request

"*Therefore, because the transform action modifies the overall structure of the message, it does not match the schema that the backend service is expecting, the request fails.*"

The above statement in the exercise does not make sense. All that is intended is that the client receives a response that is modified by your MPGW service from what the backend sends as a response.

`<book:ConfirmationCode>`	becomes	`<book:ConfirmationText>`base64 decoded
`<book:Expiry>`, `<book:CVV>`	removed	
`<book:Number>`	text all * except last four digits	

Exercise 4. Adding error handling to a service policy

After completing this exercise, you should be able to:
- Configure an error policy at the MPGW service level
- Configure a service policy with an On Error action
- Configure a service policy with an Error rule

4.1. Initialize the lab environment (already done in Exercise 1.)
4.2. Add error processing
Section 1: Add an Error Policy
- BookingServiceProxy_ErrorPolicy
matching on **GenericErrorcode 0x0***
action **static (local)** **default-error.html** (has html seen below)

Section 2: Test the default error policy **08 – Request Error**
`<`**PaymentCardDetailsBad**`>` - fails WSDL validation

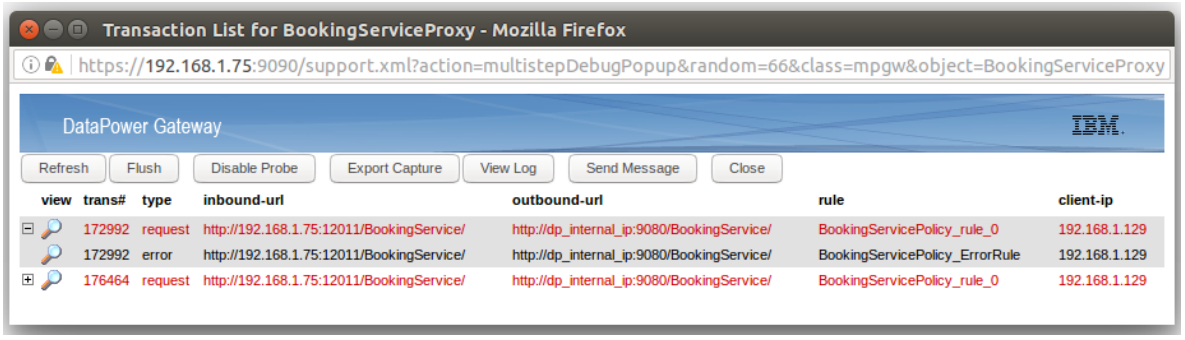
```
<html>
    <head>
        <title>FLY Airlines services</title>
    </head>
    <body>
        <h2>== Error in BookingServiceProxy == </h2>
        <h2> Default error processing has occurred </h2>
```

```
    </body>
</html>
```

Section 3: Create the error rule and add it to the service policy
- same match as used above: **`GenericErrorcode`**
- rule contains XSL, outputs service vars **`custom-error.xsl`**

Section 4: Test the error rule **`08 – Request Error`**
`<PaymentCardDetailsBad>` - fails WSDL validation

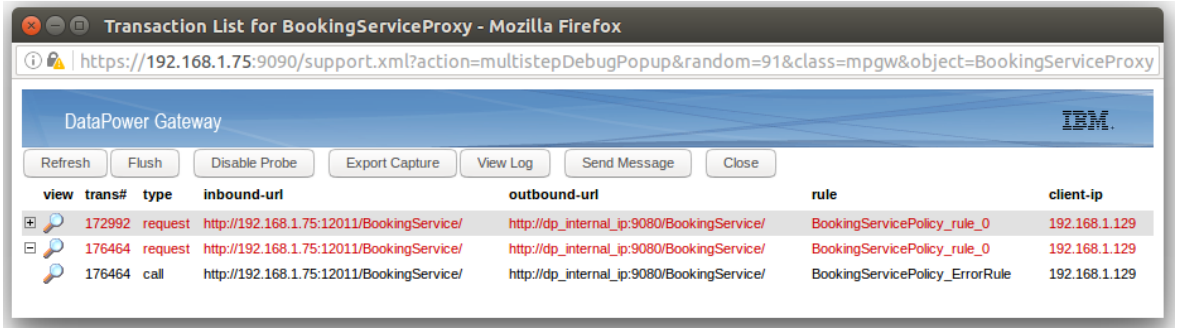
Exercise line: `__28. Click the Service Variables tab.`
- closely inspect the full list of service variables
Probe has type **`error`**



Section 5: Add an On Error action to the policy
- goes before the initial validate action
- calls the same *BookingServicePolicy_ErrorRule* (custom-error.xsl)

Section 6: Test the On Error action **`08 – Request Error`**
`<PaymentCardDetailsBad>` - fails WSDL validation

Probe has type **`call`**



Section 7: Add another Error rule and `On Error` action
- *BookingServicePolicy_filter_ErrorRule*
  has transform with **`filter-custom-error.xsl`**
- new `On Error` action **before SQL Injection filter**

Section 8: Send a message to test the new error-handling **`05 - SQL Injection`**
This error will be reported to Application Security

Exercise 5. Creating cryptographic objects and configuring SSL

After completing this exercise, you should be able to:

- Generate crypto keys by using the DataPower cryptographic tools
- Create a crypto identification credential by using a crypto key object and a crypto certificate object
- Validate certificates by using a validation credential object
- Create an SSL client profile that initiates an SSL connection request from a DataPower service
- Create an SSL server profile that accepts an SSL connection request from a client
- Create an SNI SSL server profile that accepts an SSL connection request with an SNI extension from a client

5.1. Initialize the lab environment (already done in Exercise 1.)

5.2. Generate a certificate-key pair on the DataPower gateway
ServerA, ServerB, ServerC, StudentClient

Click *`Save changes`* after the above steps.

5.3. Create cryptographic objects, Crypto Validation Credentials
Crypto Identification Credentials,

5.4. Create SSL/TLS objects

All default cipher settings

ServerA: default TLS

ServerB: no TLS 1.0

ServerC: no TLS 1.0, no TLS 1.1

Request client authentication, disable Send client authentication CA list

SSL Server Profile:

```
ssl-sni-mapping "AllServersMap"
  sni-mapping "*serverA" "ServerA"
  sni-mapping "*serverB" "ServerB"
  sni-mapping "*serverC" "ServerC"
  sni-mapping "*" "ServerC"
```

SSL SNI Server Profile

AllServersProfile (*AllServersMap*)

SSL Client Profile

StudentClientProfile

5.5. Verify web service behavior

Actually goes through student built MPGW **`01 - Initial Request`**

5.6. Add an HTTPS handler to the *BookingServiceProxy* service

HTTPS- FSH: *HTTPS_12nn2* **`<mpgw_booking_ssl_port>`**

SSL SNI server profile ***`AllServersProfile`***

5.7. Test the HTTPS handler - *** CORRECTION to name: <mpgw_booking_ssl_port>

cd $HOME/Documents

sudo cp /usr/labfiles/dp/BookingService/BookingRequest.xml .

sudo chown localuser BookingRequest.xml

_5. Run with SNI call:

curl --resolve serverA:<mpgw_booking_ssl_port>:<dp_public_ip> --data-binary @BookingRequest.xml https://serverA:<mpgw_booking_ssl_port>/BookingService -k -v

e.g.:

curl --resolve serverA:12002:192.168.0.101 --data-binary @BookingRequest.xml https://serverA:12002/BookingService -k -v

_7. Run without SNI name:

`<dp_external_ip>` is typo, it should be `<dp_public_ip>`

curl --data-binary @BookingRequest.xml https://192.168.0.101:12002/BookingService -k -v

Above fails.

_8. SSL SNI Server Profile *AllServersProfile*: set Default server profile to *ServerA* (was blank)

Same above curl now succeeds, MPGW serves up certificate for *ServerA*.

5.8. Configure an SSL Proxy Booking MPGW: *BookingServiceSSLProxy*
adding another MPGW that become client to above
HTTP Handler HTTP2_12nn3 <mpgw_booking_client>: 12nn3
*** CHECK SOAPUI Preferences / Global Properties has entry for **`mpgw_booking_client`**
***** Sometimes the Blueprint Console does not show the section for the below!**
Client Profile *StudentClientProfile*

5.9. Test the SSL connection between the BookingServiceSSLProxy and the BookingServiceProxy

ERROR:
<mpgw_ssl_booking_port> NOT USED

Support links for this exercise:
[SNI - Success with Curl]
[Demonstration of SNI Used By Browser]

Exercise 6. Implementing a service level monitor in a multi-protocol gateway

After completing this exercise, you should be able to:
• Specify service level monitoring criteria for a multi-protocol gateway
• Inspect and edit an SLM policy object
• Create an SLM Resource Class object
• Create a custom log target for SLM events

6.1. Initialize the lab environment (already done in Exercise 1.)

6.2. Test the existing MPGW with SoapUI **`01 - Initial Request`**

6.3. Test the existing *BookingServiceProxy* by using the load test
Check no SLM in *BookingServicePolicy*
10 messages in 10 seconds **`TestSuite / SLM LoadTest 10`**

The Limit of 10 in seconds defines how long the load test runs. i.e 10 seconds.
Threads is the number of concurrent users. Only one for this test.
Test 1000 delay milliseconds between message transmissions, one message every second.
Summary 10 messages over 10 seconds.
The Test Step of **01 - Initial Request**

6.4. Create a log target for SLM log messages
If a custom log target is defined with a Log Format of XML, the log is included in the Target list, and can be viewed.

6.5. Add SLM criteria to the MPGW
When travel booking requests **exceed five requests per minute**, you want to **reject** (*throttle*) the new requests.
SLM Resource Class: *BookingRequestResource* /*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='BookingRequest']
SLM Policy page: *LimitBookingRequests*

Execute All Statements

Statement

Identifier: 1

User Annotation: Terminate at 5
Resource Class: *BookingRequestResource*
SLM Action: throttle
Threshold Interval Length: 30 seconds
Threshold Level: 5
Credential Class empty, because not limiting the control from a specific client

Reporting Aggregation Level
Specifies the base aggregation level in minutes for reporting of statistics. It is the statistic interval. This value does not affect the thresholding intervals.

Maximum Records Across Intervals
A single reporting aggregation interval may contain multiple records, one record per resource or credential, for example. The total records to be saved across the maximum saved intervals is configurable via this parameter. This allows a maximum memory-consumption threshold to be set.

6.6. Test the SLM action (throttle)
***** Check with the WebGUI that the Blueprint Console has set the SLM Policy page: *LimitBookingRequests* settings correctly!**

6.7. Change the SLM statement to “shape”
6.8. Test the SLM action with “shape”
- may need to run this more than once to see expected log output

Try running this shell script in the Linux terminal to get a better idea of what is happening for both 6.6, 6.8:
https://github.com/steve-a-edwards/we751/blob/master/driveSLM-WE751.sh

---

Exercise 7. Using a DataPower pattern to deploy a service

After completing this exercise, you should be able to:
• Import a pattern
• Specify the values for the points of variability in the pattern
• Deploy the pattern into a generated service

7.1. Initialize the lab environment (already done in Exercise 1.)

7.2. Import a pattern into your application domain
*** need to refresh browser after import

7.3. Deploy a pattern
<mpgw_patterns_port>
12008
*** For the ***LDAP bind DN*** setting, do not put any spaces between the components

7.4. Test the generated service
This document shows how you can test if LDAP is working: `/usr/labfiles/LDAP-Checking.txt`
(Working on SE MA image)

In addition to looking at the log entries, it is worthwhile also looking at the probe windows (the probe is already on, presumably because of this being on in the imported blueprint.)

Transaction 755024 Context 1

Not Secure | https://192.168.1.75:9090/support.xml?action=multistepDebugStepPopup&class=mpgw&o...

Previous | Processing Step 1 | Next

Step 1: AAA Action:Input=INPUT, TransformLanguage=none, ActionDebug=off, Output=dpvar_1, NamedInOutLocationType=default, AAA=BookingServiceProxyFromPattern_BookingServiceLDAP, SSLClientConfigType=proxy, Transactional=off, SOAPValidation=body, SQLSourceType=static, JWSVerifyStripSignature=on, Asynchronous=off, ResultsMode=first-available, RetryCount=0, RetryInterval=1000, MultipleOutputs=off, IteratorType=XPATH, Timeout=0, MethodRewriteType=GET, MethodType=POST, MethodType2=POST

Extension Trace | Execution Trace | Stylesheet Parameters

{0}:

seq	type	url	request	resp mode	response	error
1	ldap-authen	192.168.1.122	*(show nodeset)*	xml-parse	*(show nodeset)*	
2	ldap-search	192.168.1.122	*(show nodeset)*	xml-parse	*(show nodeset)*	

View Nodeset

Not Secure | https://192.168.1.75:9090/support.xml?action=multistepDebug...

Content of Request:

```
<parameter
    xmlns:dp="http://www.datapower.com/schemas/management"
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
>
  <bindDN>cn=student, dc=ibm, dc=com</bindDN>
  <bindPassword>passw0rd</bindPassword>
  <lookupDN />
  <lookupAttribute />
  <filter />
</parameter>
```

Show unformatted

View Nodeset

Not Secure | https://192.168.1.75:9090/support.xml?action=multis...

Content of Response:

```
<entry type="ldap"
    xmlns:dp="http://www.datapower.com/schemas/management"
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
>cn=student,dc=ibm,dc=com</entry>
```

Show unformatted

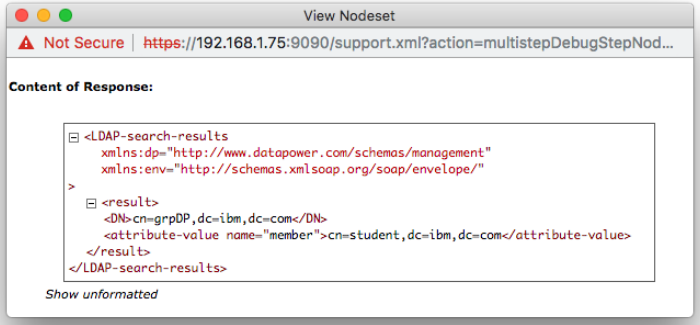
View Nodeset

Not Secure | https://192.168.1.75:9090/support.xml?action=multist...

Content of Request:

```
<parameter
    xmlns:dp="http://www.datapower.com/schemas/management"
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
>
  <bindDN>cn=admin, dc=ibm, dc=com</bindDN>
  <bindPassword>passw0rd</bindPassword>
  <lookupDN>cn=grpDP, dc=ibm, dc=com</lookupDN>
  <lookupAttribute>member</lookupAttribute>
  <filter>(objectClass=*)</filter>
</parameter>
```

Show unformatted



Extra Notes

These are the only ports needed for all of the above exercises:
<mpgw_booking_port>
12nn1
<mpgw_booking_ssl_port>
12nn2
<mpgw_ssl_booking_port> should be <mpgw_booking_client>
12nn3
<mpgw_helloworld_port>
12nn7
<mpgw_patterns_port>
12nn8

- where nn is the student number, if IRLP nn = 01