Table of Contents

HOW TO USE THIS LAB WORKBOOK

This workbook contains **30+ labs** that will show you how to use the Policy Manager and Gateway to protect your APIs and Web Services. We introduce each lab within the context of an airline, Voonair. You can use this lab workbook at the prompt of your instructor once you cover the basics and configure the API Gateway.

This workbook assumes you have a Gateway in RUNNING status as well as the Policy Manager application running and connected to that Gateway. It also assumes you have access to the various supporting files (ie. .wadl and .wsdl files).

PRE-REQUISITES FOR THE LABS

- You need to have your API Gateway .ova up and running. For configuration instructions please refer to the **Gateway Configuration Guide** as part of your training handouts. Start VMPlayer and click the virtual instance of the Gateway that you configured.
- You need the Policy Manager installed and licensed, as per the configuration guide. And running.
- SDE Environment – this is the .zip file that should be made available in your training folder and is installed in stage 5 of the configuration guide. It is expressed as a .tgz file.
- Access to provided training assets to a local or shared folder. This folder contains docs, licenses, etc.

THE MODULES AND THEIR CORRESPONDING LABS

Remember **Voonair**?

The map to the right outlines the **9 distinct Modules** that this workbook will take you through.

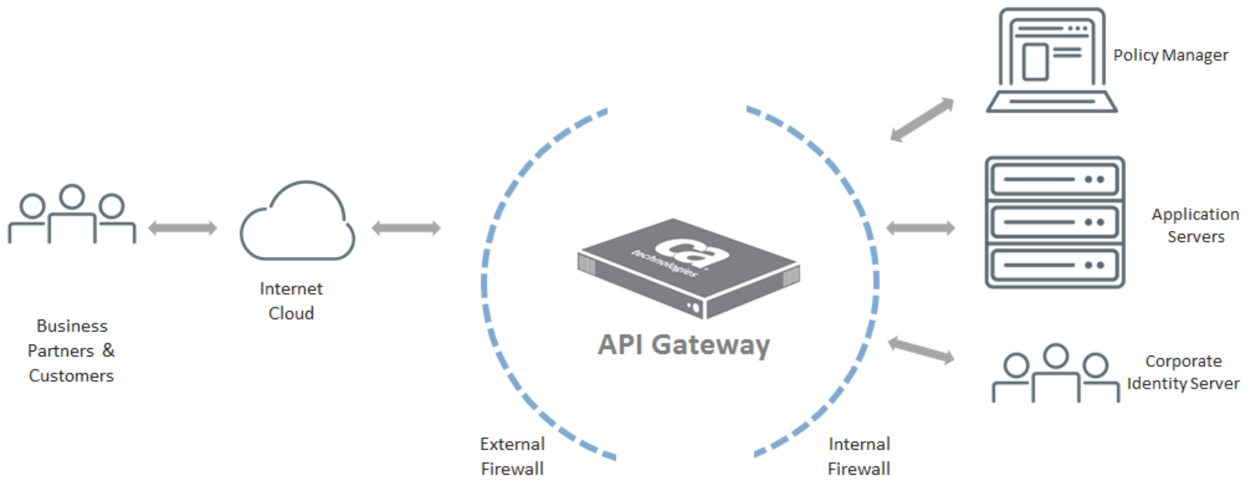
These are designed to help you learn about the features in the Policy Manager. Follow this airline and the characters involved to see how Voonair wins by using the CA API Gateway toolset.

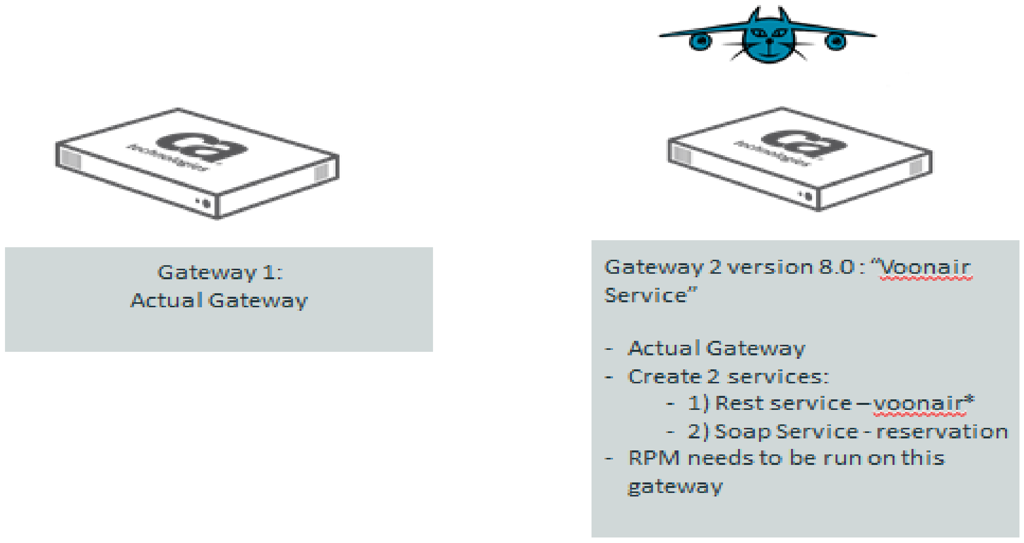
THE GATEWAY TRAINING ENVIRONMENT

To be able to complete the labs in this workbook, you will need to set up the training environment.

We start by setting up the Virtual Machine and the training environment.

The training environment (simulated backend services, ex. LDAP Server, a Soap and Rest service exposed by the 2nd gateway) allows you to simulate an end-to-end message flow. In the labs that follow, you will be sending messages from the client to the backend services and writing policies on the Gateway that affect that message.





Gateway 1:
Actual Gateway

Gateway 2 version 8.0 : "Voonair Service"

- Actual Gateway
- Create 2 services:
  - 1) Rest service – voonair*
  - 2) Soap Service - reservation
- RPM needs to be run on this gateway

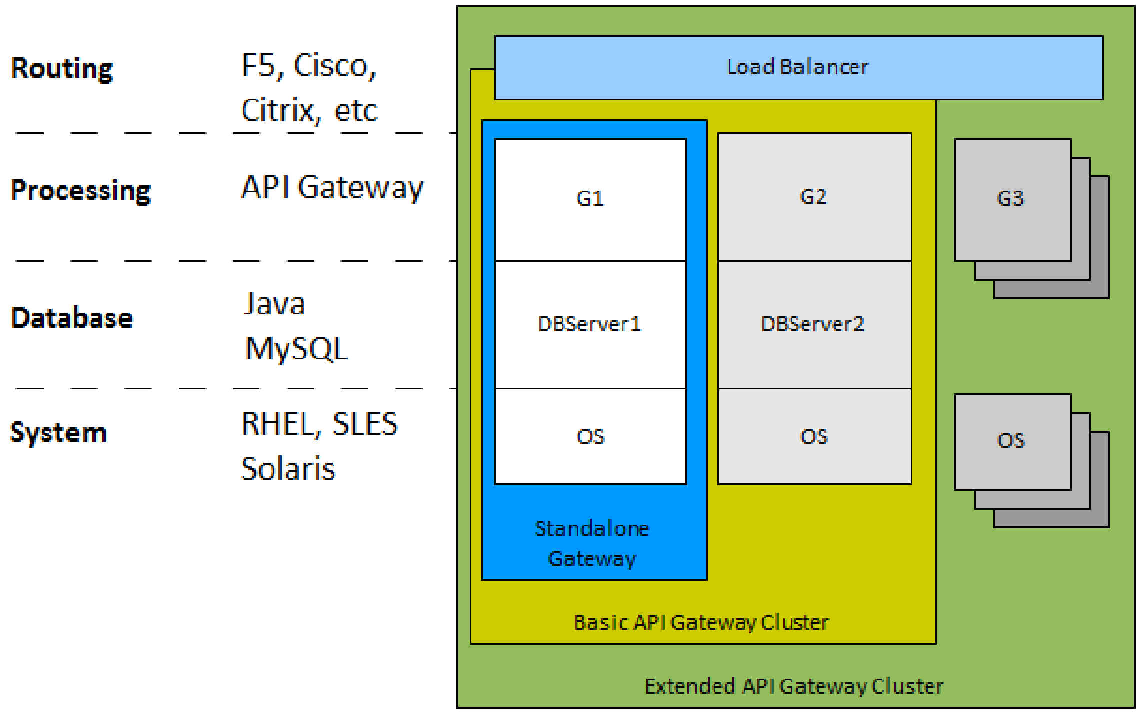
Configuring the API Gateway v8.3

Description	Stand-Up the Gateway Virtual Machine and connect it to the Policy Manager. This is an ideal guide to use if you wish to locally configure a Gateway and Policy Manager on your machine. It is also used when receiving any instructor led training. There are 5 unique stages to this process, detailed below. There is also a **cheat sheet** at the end of this guide should you wish to print for reference.
Time	**60 minutes**

Stage	Description
1	**Setup your Virtual Machine.** This allows you to access an instance of the Gateway. This step assumes you have access to the Gateway .ova file and you have downloaded various tools to support your training, including VMWare Player, SoapUI, WinSCP, and Putty.
2	**Configure your Gateway Settings.** This allows you to configure network settings and other preferences like time zones.
3	**Edit the Local Host and Create your Gateway Database.** This allows you to create a database.
4	**Install the Policy Manager and the Gateway License.** The Policy Manager is the GUI where you use services and assertions to write policies. The license key ‘unlocks’ various features of the Gateway.

THE GATEWAY ARCHITECTURE

The diagram below outlines the different layers involved in calls between clients and backend services. The Gateway sits in the processing/runtime layer. This chart shows how you can have various configurations including a standalone gateway or a cluster.



MODULE 1
PUBLISHING A SOAP WEB SERVICE

LAB 1A: PUBLISHING A SERVICE

THE SITUATION

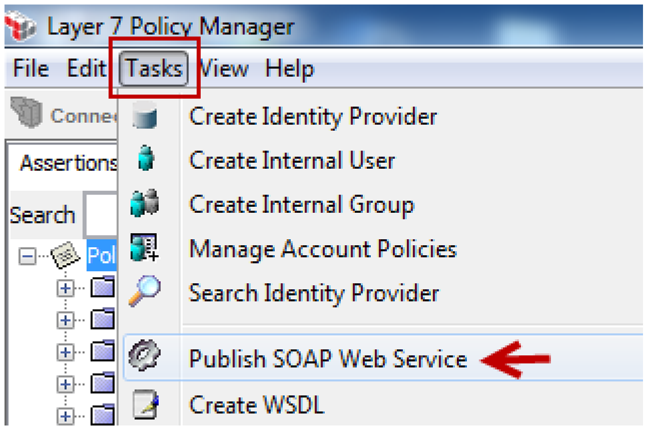
Voonair has never exposed their APIs to external partners and want to make sure their systems are secure when they do this. They already have their application running and have provided us a description via a .wsdl file for publishing on the Gateway.

THE SOLUTION

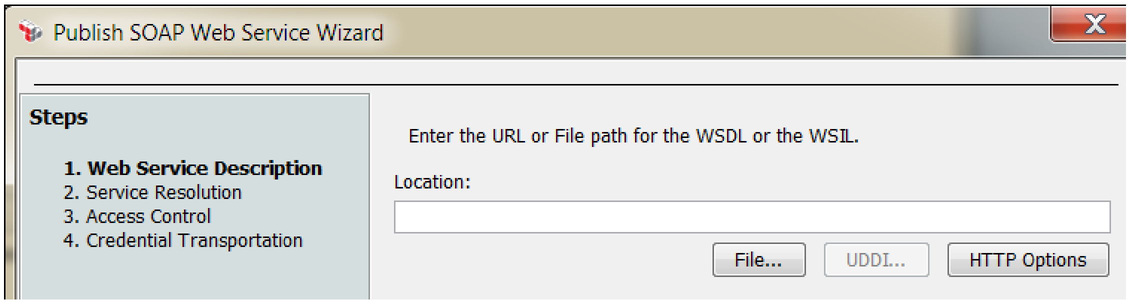
In order for Voonair to accomplish all of their business goals, they need to **start by publishing web services (or, APIs)** which will expose their data to partners. It is currently sitting as an unused .wsdl file.

This process can be started in the Policy Manager tool by writing and publishing web services. Let's go!

Step 1: Make sure your Gateway is up and running. Click the Policy Manager icon on your desktop, or from your Start Menu. In the **Tasks** menu, select the option titled **Publish SOAP Web Service.**



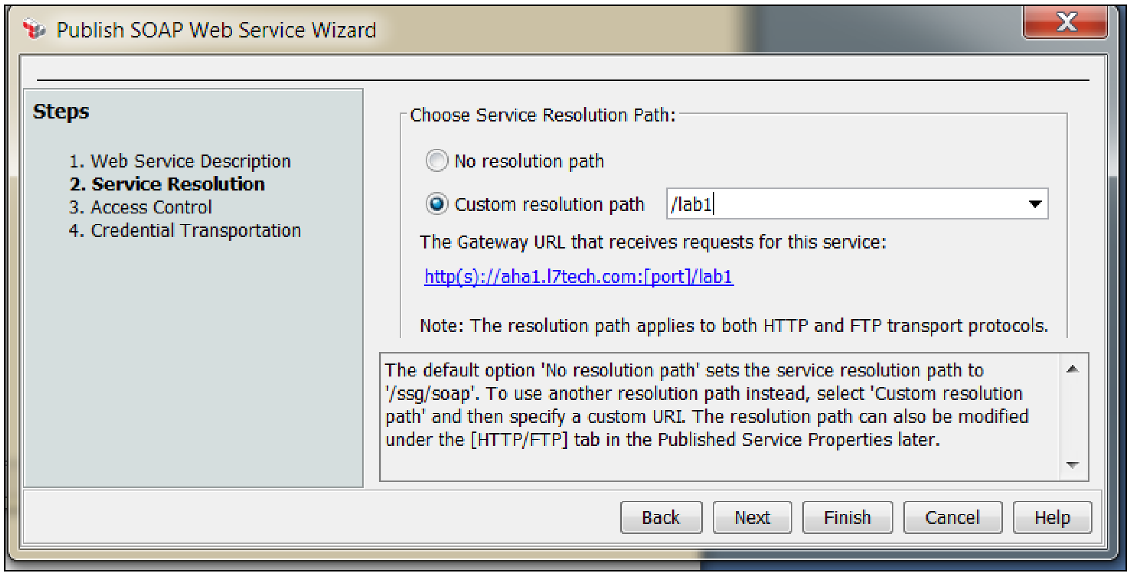
Step 2: In the wizard, click **file** and import the ***voonair.wsdl*** from the training folder. Once you have located the WSDL, click **Next** to continue.


Publish SOAP Web Service Wizard
Steps
1. Web Service Description
2. Service Resolution
3. Access Control
4. Credential Transportation
File...
UDDI...
HTTP Options

Enter the URL or File path for the WSDL or the WSIL.

Location:

Step 3: Define a **custom resolution path** for the service. We recommend **/lab1**. Then, click **Next.**


Publish SOAP Web Service Wizard
Steps
1. Web Service Description
2. Service Resolution
3. Access Control
4. Credential Transportation
Back
Next
Finish
Cancel
Help

Choose Service Resolution Path:

No resolution path [✓ Custom resolution path] /lab1

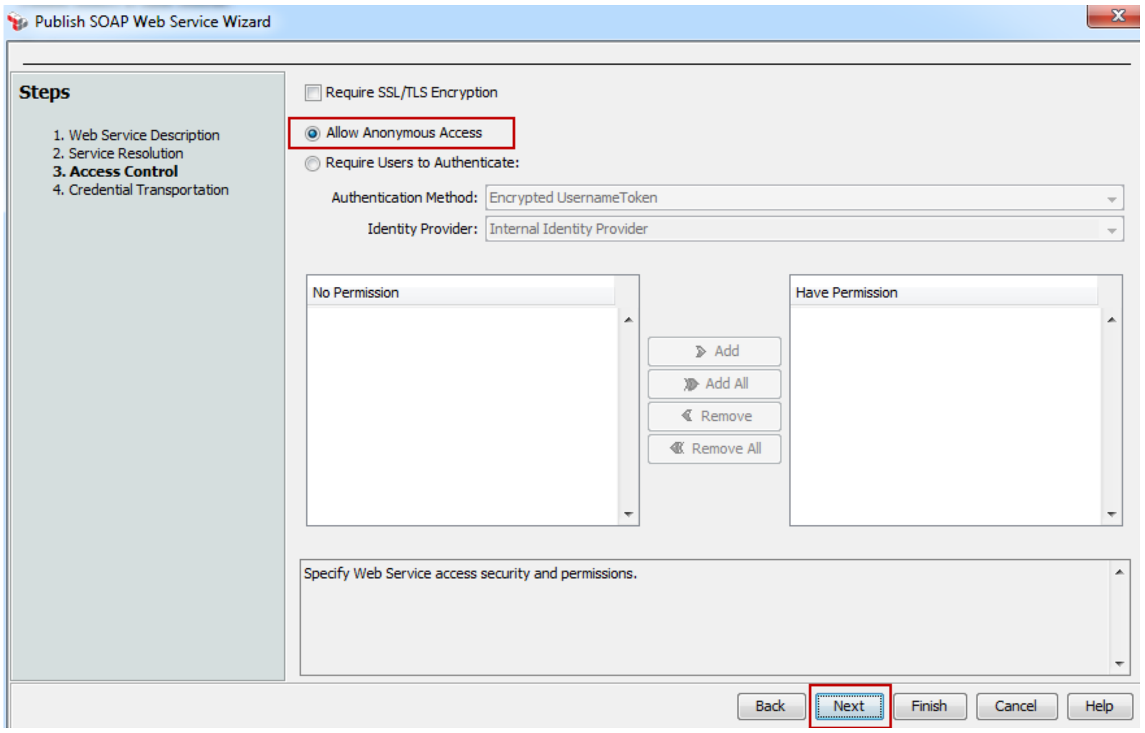
The Gateway URL that receives requests for this service:

http(s)://aha1.l7tech.com:[port]/lab1

Note: The resolution path applies to both HTTP and FTP transport protocols.

The default option 'No resolution path' sets the service resolution path to '/ssg/soap'. To use another resolution path instead, select 'Custom resolution path' and then specify a custom URI. The resolution path can also be modified under the [HTTP/FTP] tab in the Published Service Properties later.

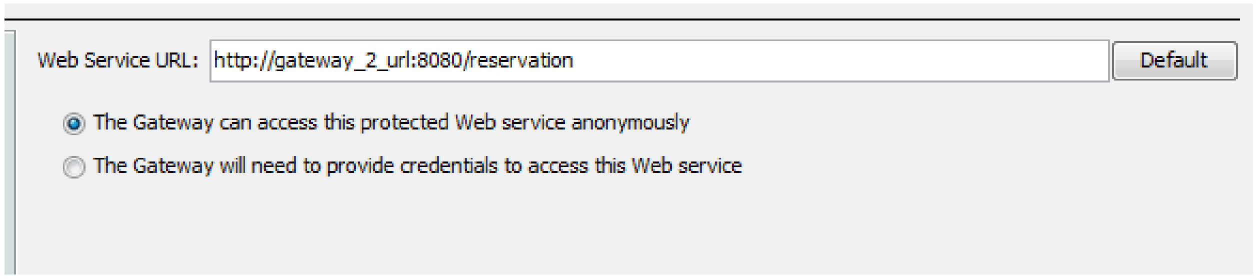
Step 4: Click **Next** again to allow anonymous access.



Require SSL/TLS Encryption [ ]
[✓ Allow Anonymous Access]
Require Users to Authenticate:
Authentication Method: Encrypted UsernameToken
Identity Provider: Internal Identity Provider

Specify Web Service access security and permissions.

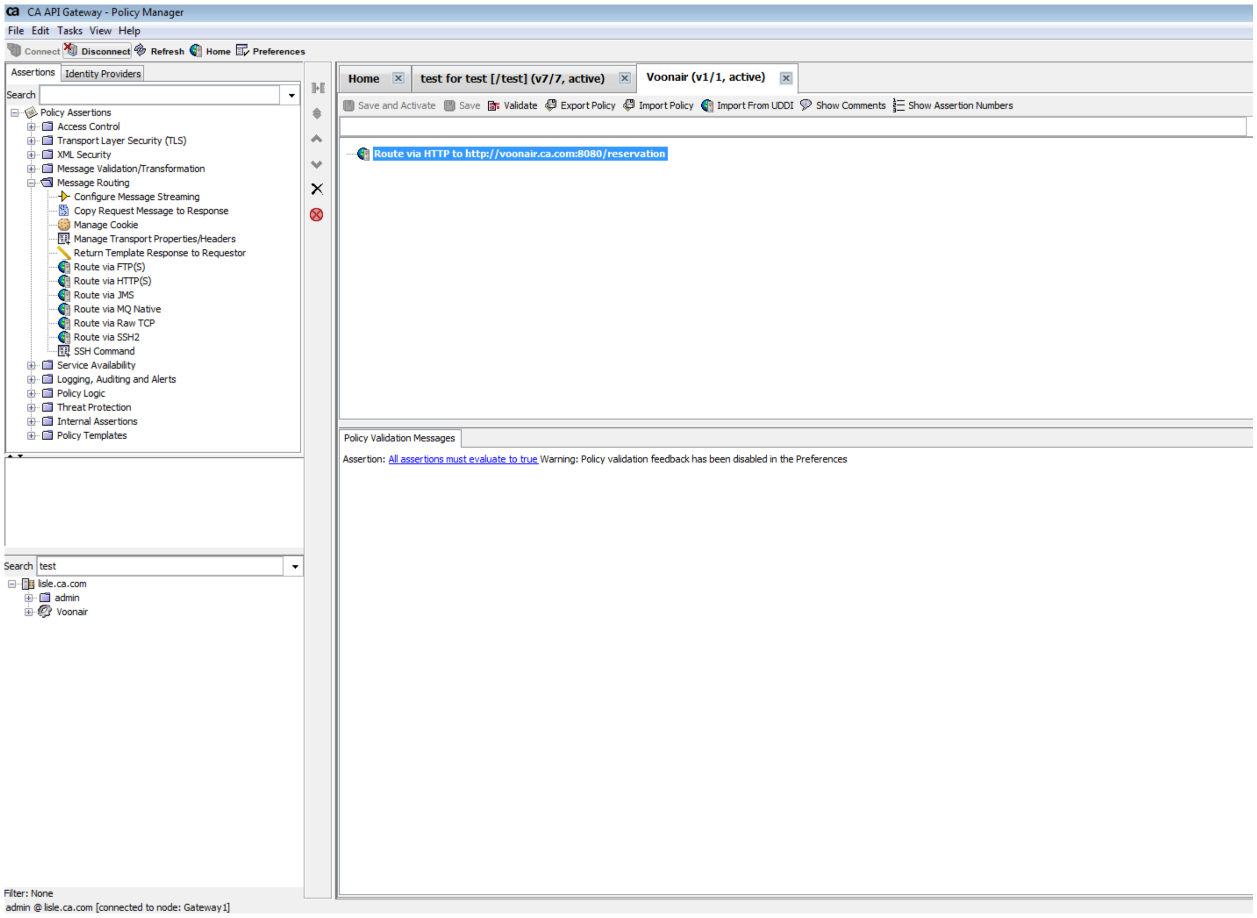
Step 5: Click in the url window and select **Change**. Replace the default URL with the one in the box below. **This is case sensitive**. Click **Finish**. The newly created Web Service will appear in the bottom left Services Panel of your Policy Manager.



Web Service URL: http://gateway_2_url:8080/reservation

[✓ The Gateway can access this protected Web service anonymously]
The Gateway will need to provide credentials to access this Web service

Below is where you can see your newly created Web Service.


CA API Gateway - Policy Manager
File Edit Tasks View Help
Connect Disconnect Refresh Home Preferences
Assertions Identity Providers
Search
Policy Assertions
Access Control
Transport Layer Security (TLS)
XML Security
Message Validation/Transformation
Message Routing
Configure Message Streaming
Copy Request Message to Response
Manage Cookie
Manage Transport Properties/Headers
Return Template Response to Requestor
Route via FTP(S)
Route via HTTP(S)
Route via JMS
Route via MQ Native
Route via Raw TCP
Route via SSH2
SSH Command
Service Availability
Logging, Auditing and Alerts
Policy Logic
Threat Protection
Internal Assertions
Policy Templates
Search test
lisle.ca.com
admin
Voonair
Filter: None
admin @ lisle.ca.com [connected to node: Gateway1]
Home
test for test [/test] (v7/7, active)
Voonair (v1/1, active)
Save and Activate Save Validate Export Policy Import Policy Import From UDDI Show Comments Show Assertion Numbers
Route via HTTP to http://voonair.ca.com:8080/reservation
Policy Validation Messages

Assertion: All assertions must evaluate to true Warning: Policy validation feedback has been disabled in the Preferences

You have successfully published a web service!

LAB 1B: SETUP THE SOAPUI CLIENT APP REQUEST

THE SITUATION

Now that Voonair has published a SOAP web service and generated a URL that can receive requests for the service, you need to create a client request. Think of this request like individual requesting flight information from a remote location.

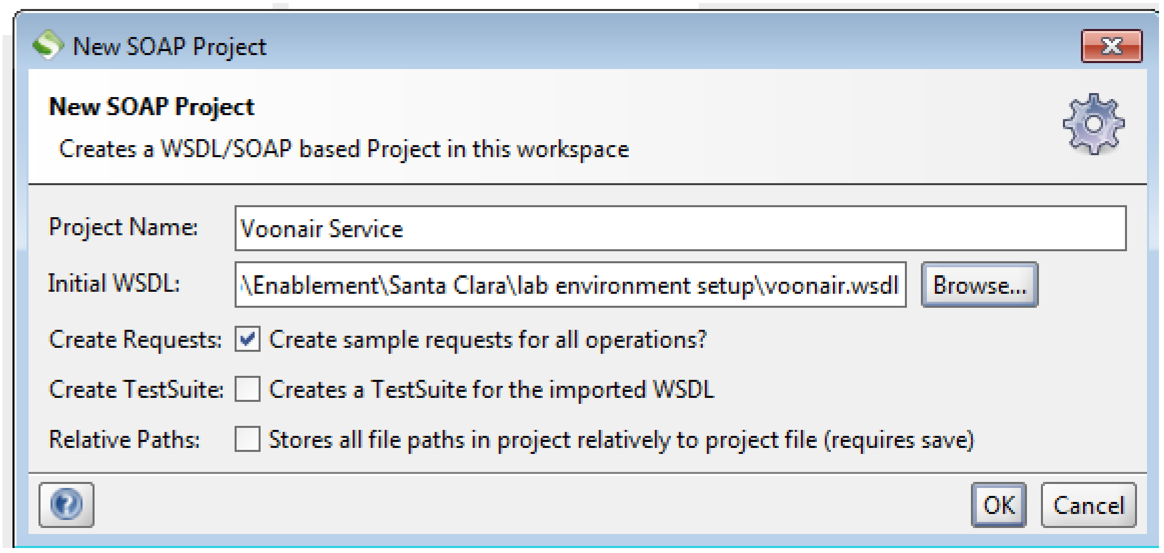
THE SOLUTION

We will use soapUI to create an external request and point it to the published Gateway service. Think of it like a 2-way conversation that we're setting up. However, our 2-way conversation is going to be moderated and mediated and, in effect, *changed* by the Gateway as the request is passed through it. The Gateway is moderating the conversation between the customer and Voonair's flight inventory.

Step 1: Start the **soapUI** application**.** *(from the SmartBear folder in your Program Files via your Start Menu)*

Step 2: Click **File,** then **New SOAP Project**. Add a project name.

Step 3: Click Browse. Locate the WSDL titled: voonair.*wsdl* in the training folder provided.



New SOAP Project

New SOAP Project

Creates a WSDL/SOAP based Project in this workspace

Project Name: Voonair Service

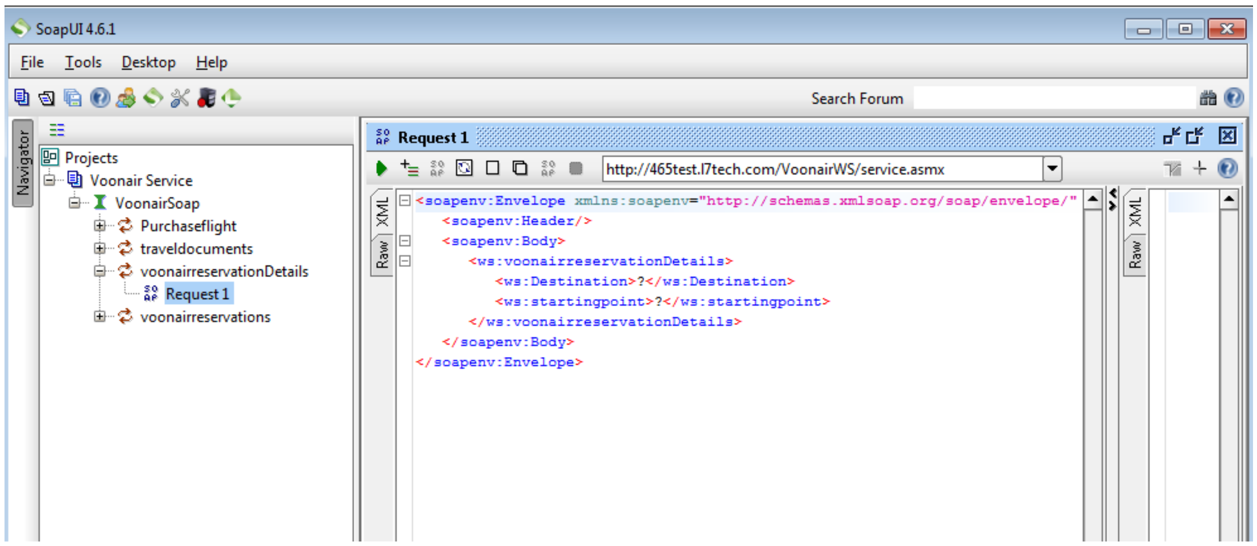
Initial WSDL: \Enablement\Santa Clara\lab environment setup\voonair.wsdl

Create Requests: [✓] Create sample requests for all operations?

Create TestSuite: [ ] Creates a TestSuite for the imported WSDL

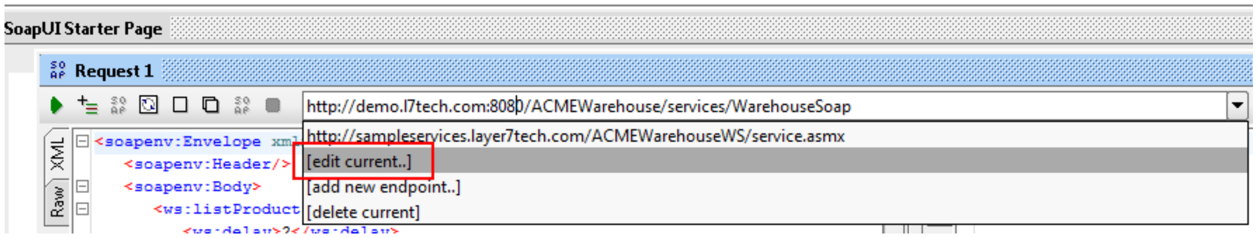
Relative Paths: [ ] Stores all file paths in project relatively to project file (requires save)

Step 4: Setup your client request by expanding <+> the **voonairreservationDetails** menu option and **double-clicking Request 1.**

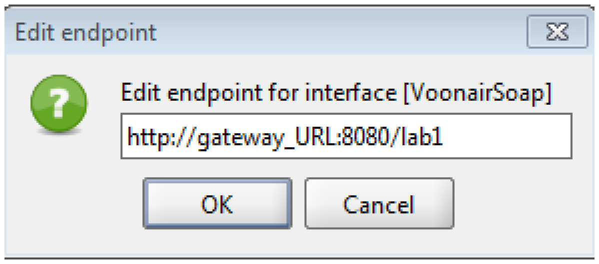


```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   <soapenv:Header/>
   <soapenv:Body>
      <ws:voonairreservationDetails>
         <ws:Destination>?</ws:Destination>
         <ws:startingpoint>?</ws:startingpoint>
      </ws:voonairreservationDetails>
   </soapenv:Body>
</soapenv:Envelope>
```

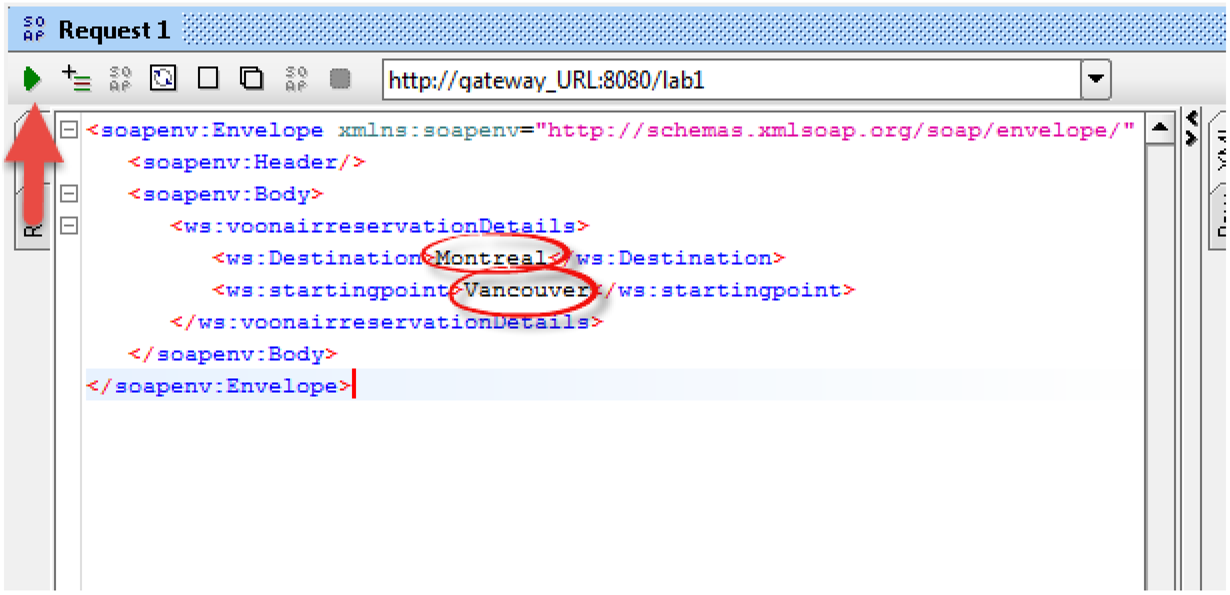
Step 5: Edit your **endpoint** to point to your newly published service on the gateway (e.g. that resolution path you set as /lab1). You can edit the endpoint by **clicking the drop-down arrow to the right in the URL field** and selecting [**edit current ..]**



Step 6: Manually change port 6060 to port 8080 as you edit the endpoint to be the Gateway URL that receives requests for this service. Refer to:

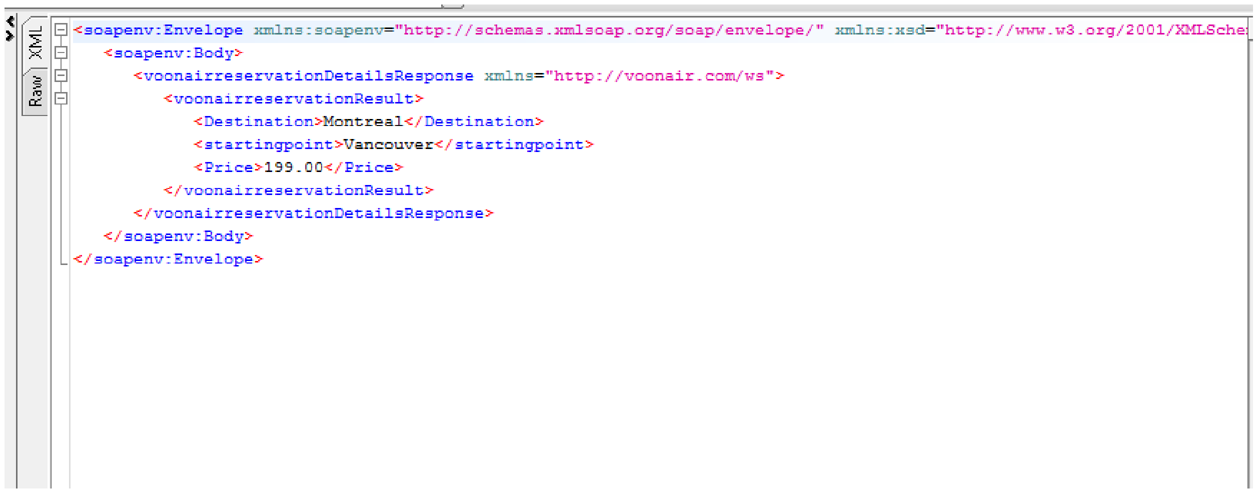

Edit endpoint
Edit endpoint for interface [VoonairSoap]
http://gateway_URL:8080/lab1
OK
Cancel

Step 7 – Change the client message element “Destination” to a WSDL defined name (eg Montreal) and “startingpoint” to a WSDL defined name (eg. Vancouver). Run the request by selecting the green arrow.



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   <soapenv:Header/>
   <soapenv:Body>
      <ws:voonairreservationDetails>
         <ws:Destination>Montreal</ws:Destination>
         <ws:startingpoint>Vancouver</ws:startingpoint>
      </ws:voonairreservationDetails>
   </soapenv:Body>
</soapenv:Envelope>
```

Your response should look like the following message if successful:



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSche
   <soapenv:Body>
      <voonairreservationDetailsResponse xmlns="http://voonair.com/ws">
         <voonairreservationResult>
            <Destination>Montreal</Destination>
            <startingpoint>Vancouver</startingpoint>
            <Price>199.00</Price>
         </voonairreservationResult>
      </voonairreservationDetailsResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

LAB 1C: PUBLISH A WEB API

THE SITUATION

The first thing Voonair wants to do is make it easier for their customers to book and check flights. To do this, they need to find a way to get their reservation system exposed via an API.

Voonair has just recently purchased the gateway to expose their reservation system to partners, such as travel agencies. When exposing APIs to external partners they want to make sure their systems are secure when they do this. They already have their reservation application running and have provided us with a REST API to expose via the Gateway.

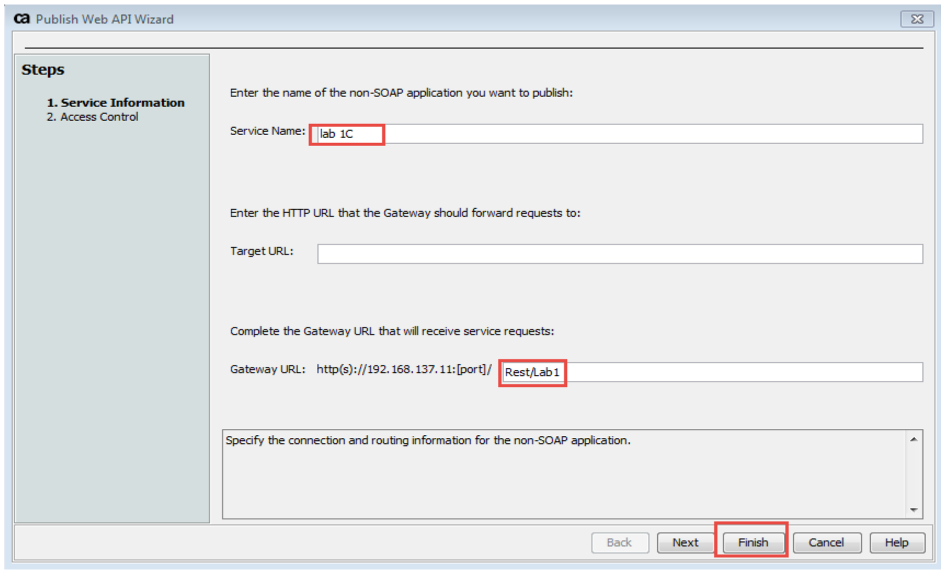
THE SOLUTION

In order for Voonair to accomplish this goal, they'll need to **publish a web API (Publish Web API)** which will expose their data to the travel agencies.

This process can be started in the Layer 7 Policy Manager tool by writing and publishing a web API service. Let's go!

Step 1: Start by clicking Tasks and selecting ***"Publish Web API"***

Step 2: The Publish Web API Wizard appears:



Enter the name of the non-SOAP application you want to publish:

Service Name: lab 1C

Enter the HTTP URL that the Gateway should forward requests to:

Target URL:

Complete the Gateway URL that will receive service requests:

Gateway URL: http(s)://192.168.137.11:[port]/ Rest/Lab1

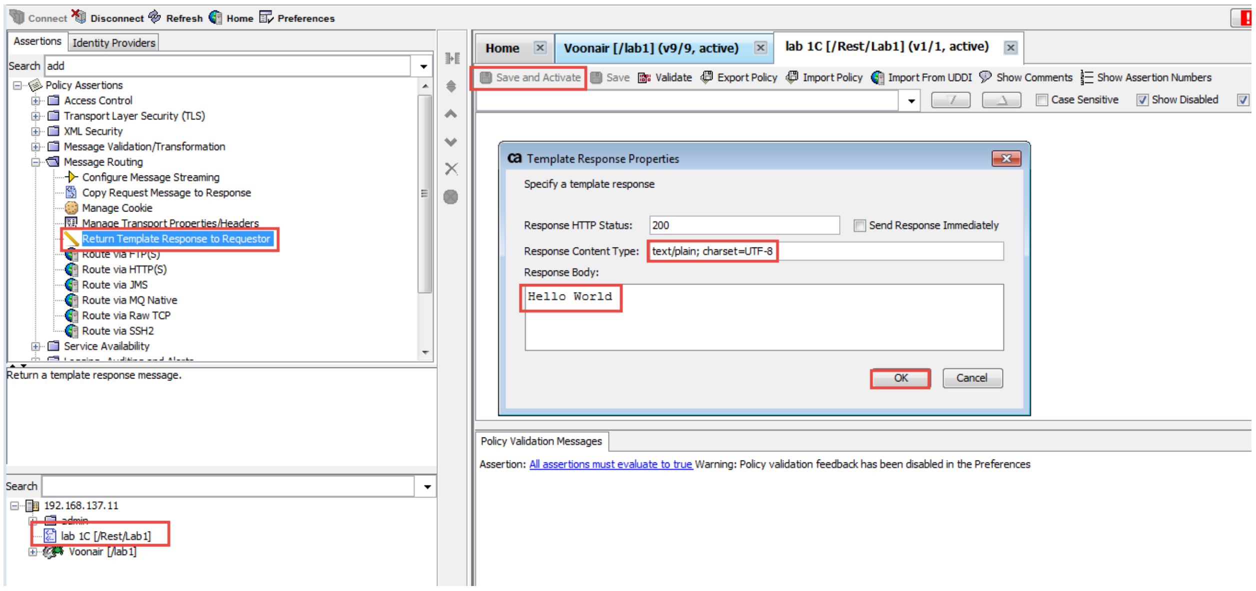
Specify the connection and routing information for the non-SOAP application.

Service Name – Is a description of the API – Lab 1 C

Gateway URL – Is the gateway URL that will expose this API – Rest/Lab1

Step 3: Click Finish.

Step 4: In order for us to test this service with a browser, we need to add a "Return Template Response to Requestor" assertion.



Search: add

Return a template response message.

Specify a template response

Response HTTP Status: 200 ☐ Send Response Immediately

Response Content Type: text/plain; charset=UTF-8

Response Body:

```
Hello World
```

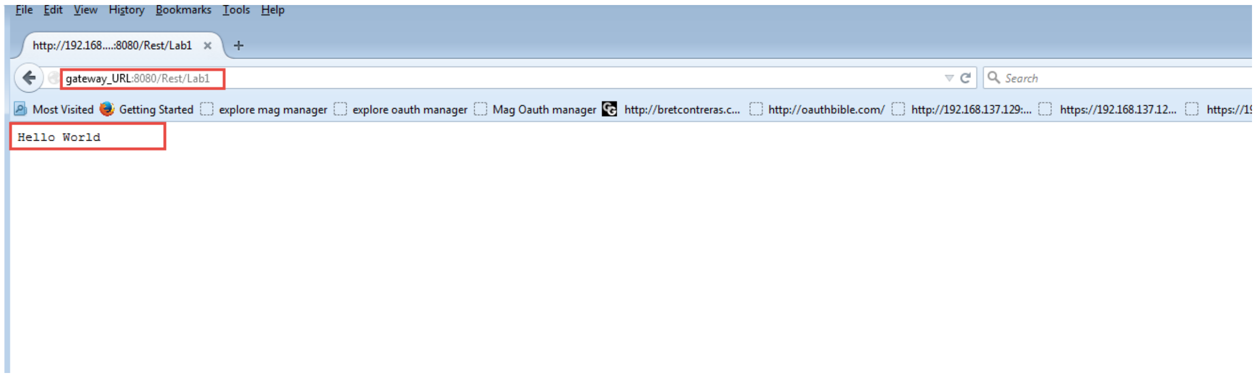
Assertion: All assertions must evaluate to true Warning: Policy validation feedback has been disabled in the Preferences

Drag "Return Template Response to Requestor" to the policy development window. Type in the Response Body – "***Hello World***" and in the Response Content Type – "***text/plain; charset=UTF-8***"

Click ***Ok*** and ***Save and Activate***

Step 5 – In order to test this service out, you can bring up a Web Browser. Place the Gateway URL in the browser area:

http://gateway_URL:8080/Rest/Lab1


File Edit View History Bookmarks Tools Help
http://192.168....:8080/Rest/Lab1
gateway_URL:8080/Rest/Lab1
Search
Most Visited
Getting Started
explore mag manager
explore oauth manager
Mag Oauth manager
http://bretcontreras.c...
http://oauthbible.com/
http://192.168.137.129:...
https://192.168.137.12...

```
Hello World
```

MODULE 2: TROUBLESHOOTING

LAB 2A: ADD AUDIT DETAILS

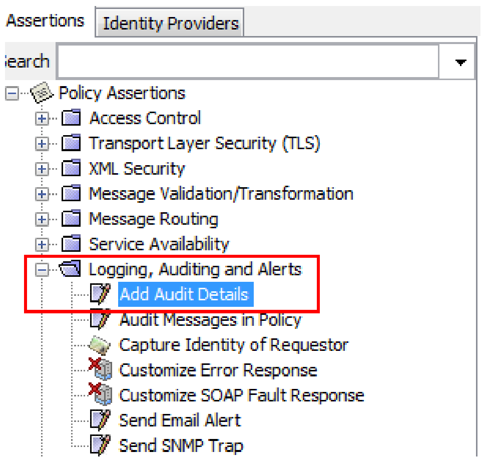
THE SITUATION:

Developer Dave is upset. A peer mistakenly deleted one of his policies. Dave needs to learn how to debug and recover his work. Not only is he having trouble debugging problems, he needs to create more new services

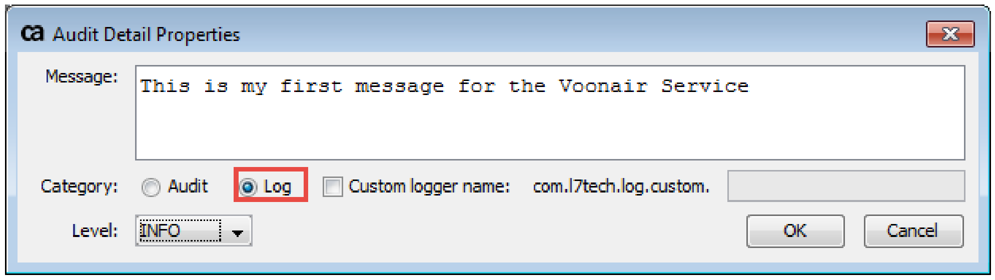
THE SOLUTION:

Dave delves into the Gateway functionality for logging, debug mode, and recovery. Follow the instructions in this series of labs to learn how to set up audits within the Policy Manager.

Step 1 – Drag and drop the **Add Audit Details** assertion into your new policy before the routing assertion.



Step 2 - Add a message to the assertion that you want logged in the policy and select Log and OK.

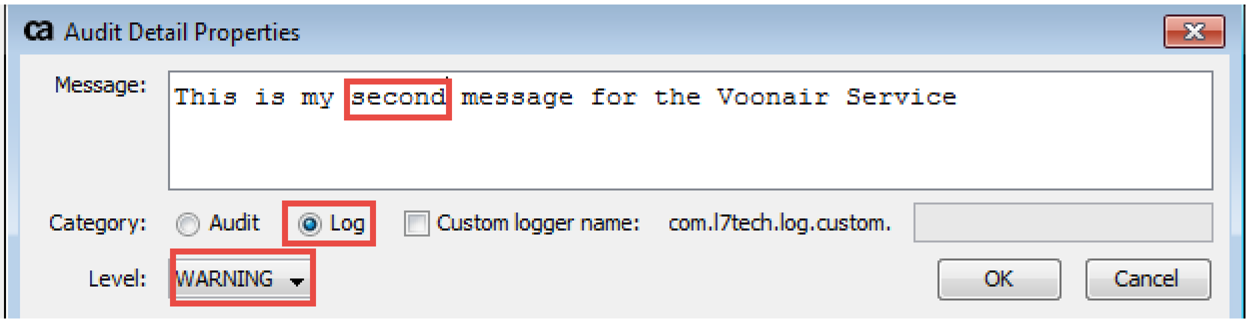

Audit Detail Properties

Message: This is my first message for the Voonair Service

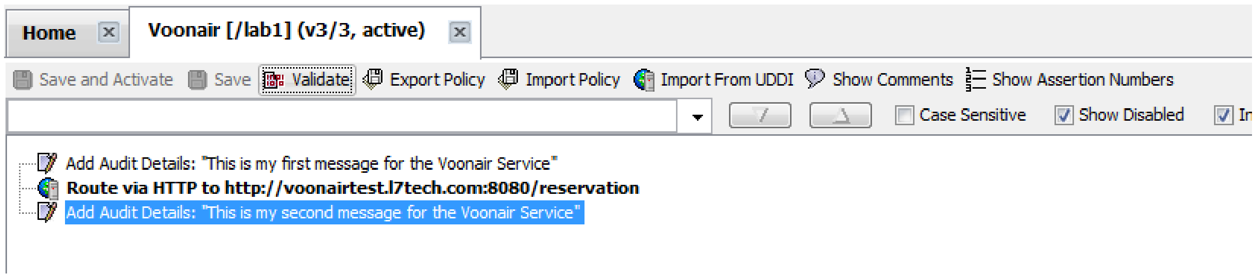
Category: Audit [✓ Log] Custom logger name: com.l7tech.log.custom.

Level: INFO

Step 3 – Add a second message to the assertion that you want logged ***after*** the routing assertion and select Log, change the drop-down menu to WARNING and then OK. **Save and Activate.**

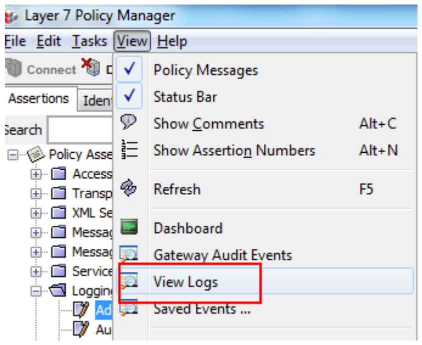


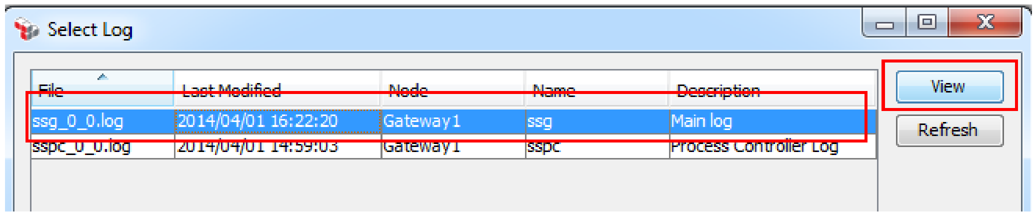
You will now have **3 lines** in your policy:



```
Add Audit Details: "This is my first message for the Voonair Service"
Route via HTTP to http://voonairtest.l7tech.com:8080/reservation
Add Audit Details: "This is my second message for the Voonair Service"
```

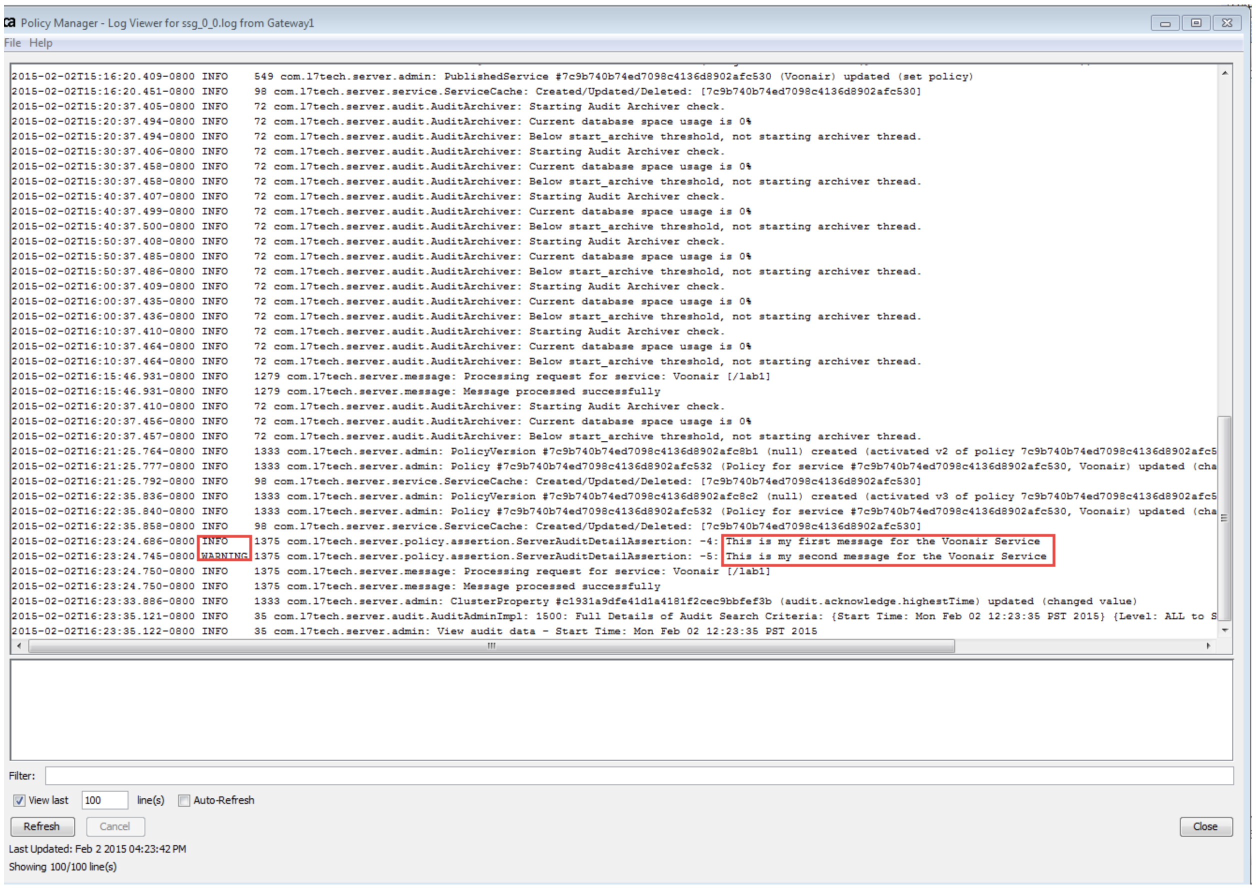
Step 4 – Send a SOAPUI request to the gateway service listening on port 8080 or port 8443. Then, within the Policy Manager select **View**, then **View Logs** to see your message.





File	Last Modified	Node	Name	Description
ssg_0_0.log	2014/04/01 16:22:20	Gateway1	ssg	Main log
sspc_0_0.log	2014/04/01 14:59:03	Gateway1	sspc	Process Controller Log

Step 5: Wait for the list to generate. Scroll to the bottom of the page,you'll see your latest soapUI requests including the first message and second message (info vs warning) that the audits generated.



```
2015-02-02T15:16:20.409-0800 INFO    549 com.l7tech.server.admin: PublishedService #7c9b740b74ed7098c4136d8902afc530 (Voonair) updated (set policy)
2015-02-02T15:16:20.451-0800 INFO    98 com.l7tech.server.service.ServiceCache: Created/Updated/Deleted: [7c9b740b74ed7098c4136d8902afc530]
2015-02-02T15:20:37.405-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Starting Audit Archiver check.
2015-02-02T15:20:37.494-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Current database space usage is 0%
2015-02-02T15:20:37.494-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Below start_archive threshold, not starting archiver thread.
2015-02-02T15:30:37.406-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Starting Audit Archiver check.
2015-02-02T15:30:37.458-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Current database space usage is 0%
2015-02-02T15:30:37.458-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Below start_archive threshold, not starting archiver thread.
2015-02-02T15:40:37.407-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Starting Audit Archiver check.
2015-02-02T15:40:37.499-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Current database space usage is 0%
2015-02-02T15:40:37.500-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Below start_archive threshold, not starting archiver thread.
2015-02-02T15:50:37.408-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Starting Audit Archiver check.
2015-02-02T15:50:37.485-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Current database space usage is 0%
2015-02-02T15:50:37.486-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Below start_archive threshold, not starting archiver thread.
2015-02-02T16:00:37.409-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Starting Audit Archiver check.
2015-02-02T16:00:37.435-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Current database space usage is 0%
2015-02-02T16:00:37.436-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Below start_archive threshold, not starting archiver thread.
2015-02-02T16:10:37.410-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Starting Audit Archiver check.
2015-02-02T16:10:37.464-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Current database space usage is 0%
2015-02-02T16:10:37.464-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Below start_archive threshold, not starting archiver thread.
2015-02-02T16:15:46.931-0800 INFO    1279 com.l7tech.server.message: Processing request for service: Voonair [/lab1]
2015-02-02T16:15:46.931-0800 INFO    1279 com.l7tech.server.message: Message processed successfully
2015-02-02T16:20:37.410-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Starting Audit Archiver check.
2015-02-02T16:20:37.456-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Current database space usage is 0%
2015-02-02T16:20:37.457-0800 INFO    72 com.l7tech.server.audit.AuditArchiver: Below start_archive threshold, not starting archiver thread.
2015-02-02T16:21:25.764-0800 INFO    1333 com.l7tech.server.admin: PolicyVersion #7c9b740b74ed7098c4136d8902afc8b1 (null) created (activated v2 of policy 7c9b740b74ed7098c4136d8902afc5
2015-02-02T16:21:25.777-0800 INFO    1333 com.l7tech.server.admin: Policy #7c9b740b74ed7098c4136d8902afc532 (Policy for service #7c9b740b74ed7098c4136d8902afc530, Voonair) updated (cha
2015-02-02T16:21:25.792-0800 INFO    98 com.l7tech.server.service.ServiceCache: Created/Updated/Deleted: [7c9b740b74ed7098c4136d8902afc530]
2015-02-02T16:22:35.836-0800 INFO    1333 com.l7tech.server.admin: PolicyVersion #7c9b740b74ed7098c4136d8902afc8c2 (null) created (activated v3 of policy 7c9b740b74ed7098c4136d8902afc5
2015-02-02T16:22:35.840-0800 INFO    1333 com.l7tech.server.admin: Policy #7c9b740b74ed7098c4136d8902afc532 (Policy for service #7c9b740b74ed7098c4136d8902afc530, Voonair) updated (cha
2015-02-02T16:22:35.858-0800 INFO    98 com.l7tech.server.service.ServiceCache: Created/Updated/Deleted: [7c9b740b74ed7098c4136d8902afc530]
2015-02-02T16:23:24.686-0800 INFO    1375 com.l7tech.server.policy.assertion.ServerAuditDetailAssertion: -4: This is my first message for the Voonair Service
2015-02-02T16:23:24.745-0800 WARNING 1375 com.l7tech.server.policy.assertion.ServerAuditDetailAssertion: -5: This is my second message for the Voonair Service
2015-02-02T16:23:24.750-0800 INFO    1375 com.l7tech.server.message: Processing request for service: Voonair [/lab1]
2015-02-02T16:23:24.750-0800 INFO    1375 com.l7tech.server.message: Message processed successfully
2015-02-02T16:23:33.886-0800 INFO    1333 com.l7tech.server.admin: ClusterProperty #c1931a9dfe41d1a4181f2cec9bbfef3b (audit.acknowledge.highestTime) updated (changed value)
2015-02-02T16:23:35.121-0800 INFO    35 com.l7tech.server.audit.AuditAdminImpl: 1500: Full Details of Audit Search Criteria: {Start Time: Mon Feb 02 12:23:35 PST 2015} {Level: ALL to S
2015-02-02T16:23:35.122-0800 INFO    35 com.l7tech.server.admin: View audit data - Start Time: Mon Feb 02 12:23:35 PST 2015
```

Filter:

[✓] View last 100 line(s) [ ] Auto-Refresh

Last Updated: Feb 2 2015 04:23:42 PM
Showing 100/100 line(s)

Does this help Developer Dave? Not enough. So let's move on to the next stage in this lab.

LAB 2B: AUDIT MESSAGE IN POLICY

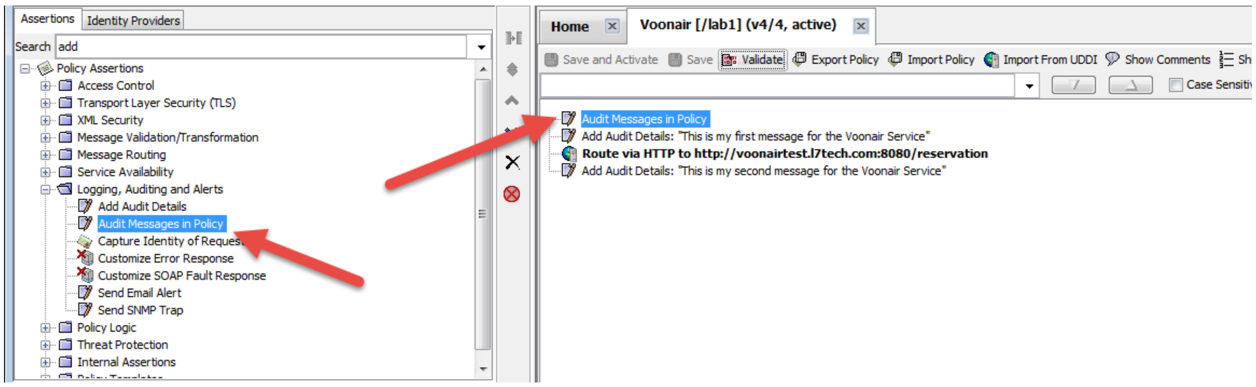
THE SITUATION

This isn't enough. Dave needs to see both the successful AND failed attempts when he runs through his tests. He will need to override the overall audit settings in order to view both the request and response messages.

THE SOLUTION

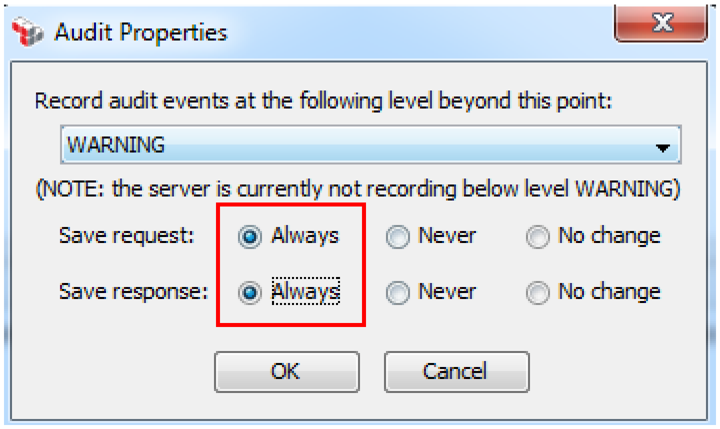
Dave turns on additional detail auditing by inserting "Audit Messages in Policy" at the top of his policy. This will elevate all messages to WARNING and capture the request and response in the audits.

Step 1 – Using the same soap service**,** drag and drop the Audit Message in Policy assertion into your soap policy before the add audit details assertion.



Audit Messages in Policy
Add Audit Details: "This is my first message for the Voonair Service"
Route via HTTP to http://voonairtest.l7tech.com:8080/reservation
Add Audit Details: "This is my second message for the Voonair Service"

Step 2 – Double click on the Audit Message in Policy. Change "**save request**" and "**save response**" buttons to Always. **Click OK**.



Record audit events at the following level beyond this point:

WARNING

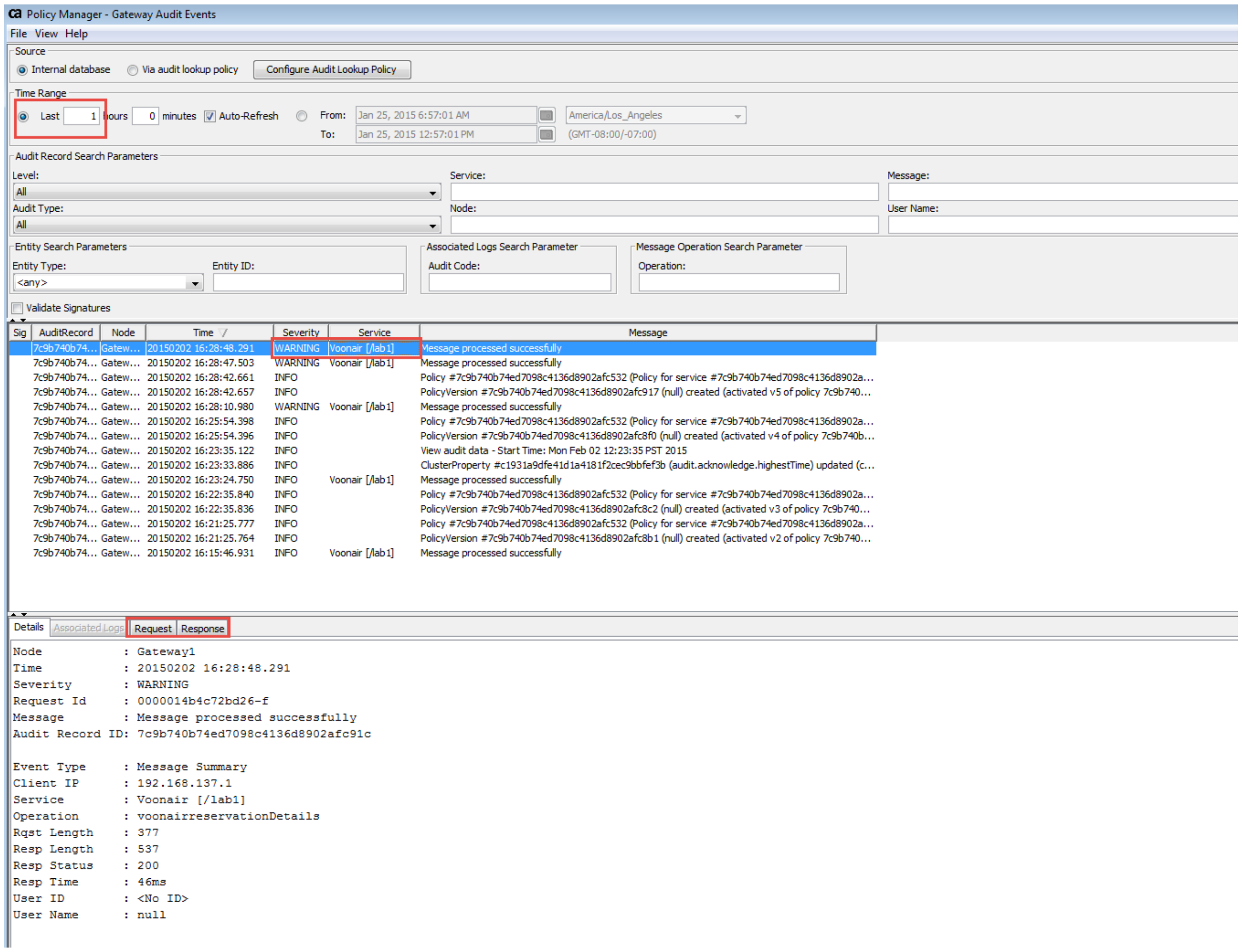
(NOTE: the server is currently not recording below level WARNING)

Save request: [✓ Always] Never No change

Save response: [✓ Always] Never No change

Step 3 – ***Save and activate your policy***

Step 4 - Send a soapUI request to the gateway service listening on port 8080 or port 8443 and then within the Policy Manager select “View” -> “Gateway Audit Events” to see your audits.


Policy Manager - Gateway Audit Events
File View Help
Configure Audit Lookup Policy
Details
Associated Logs
Request
Response

Source: [✓ Internal database] Via audit lookup policy

Time Range: [✓ Last] 1 hours 0 minutes [✓ Auto-Refresh] From: Jan 25, 2015 6:57:01 AM To: Jan 25, 2015 12:57:01 PM America/Los_Angeles (GMT-08:00/-07:00)

Audit Record Search Parameters — Level: All; Audit Type: All; Service:; Node:; Message:; User Name:

Entity Search Parameters — Entity Type: <any>; Entity ID:

Associated Logs Search Parameter — Audit Code:

Message Operation Search Parameter — Operation:

Validate Signatures

Sig	AuditRecord	Node	Time	Severity	Service	Message
	7c9b740b74...	Gatew...	20150202 16:28:48.291	WARNING	Voonair [/lab1]	Message processed successfully
	7c9b740b74...	Gatew...	20150202 16:28:47.503	WARNING	Voonair [/lab1]	Message processed successfully
	7c9b740b74...	Gatew...	20150202 16:28:42.661	INFO		Policy #7c9b740b74ed7098c4136d8902afc532 (Policy for service #7c9b740b74ed7098c4136d8902a...
	7c9b740b74...	Gatew...	20150202 16:28:42.657	INFO		PolicyVersion #7c9b740b74ed7098c4136d8902afc917 (null) created (activated v5 of policy 7c9b740...
	7c9b740b74...	Gatew...	20150202 16:28:10.980	WARNING	Voonair [/lab1]	Message processed successfully
	7c9b740b74...	Gatew...	20150202 16:25:54.398	INFO		Policy #7c9b740b74ed7098c4136d8902afc532 (Policy for service #7c9b740b74ed7098c4136d8902a...
	7c9b740b74...	Gatew...	20150202 16:25:54.396	INFO		PolicyVersion #7c9b740b74ed7098c4136d8902afc8f0 (null) created (activated v4 of policy 7c9b740b...
	7c9b740b74...	Gatew...	20150202 16:23:35.122	INFO		View audit data - Start Time: Mon Feb 02 12:23:35 PST 2015
	7c9b740b74...	Gatew...	20150202 16:23:33.886	INFO		ClusterProperty #c1931a9dfe41d1a4181f2cec9bbfef3b (audit.acknowledge.highestTime) updated (c...
	7c9b740b74...	Gatew...	20150202 16:23:24.750	INFO	Voonair [/lab1]	Message processed successfully
	7c9b740b74...	Gatew...	20150202 16:22:35.840	INFO		Policy #7c9b740b74ed7098c4136d8902afc532 (Policy for service #7c9b740b74ed7098c4136d8902a...
	7c9b740b74...	Gatew...	20150202 16:22:35.836	INFO		PolicyVersion #7c9b740b74ed7098c4136d8902afc8c2 (null) created (activated v3 of policy 7c9b740...
	7c9b740b74...	Gatew...	20150202 16:21:25.777	INFO		Policy #7c9b740b74ed7098c4136d8902afc532 (Policy for service #7c9b740b74ed7098c4136d8902a...
	7c9b740b74...	Gatew...	20150202 16:21:25.764	INFO		PolicyVersion #7c9b740b74ed7098c4136d8902afc8b1 (null) created (activated v2 of policy 7c9b740...
	7c9b740b74...	Gatew...	20150202 16:15:46.931	INFO	Voonair [/lab1]	Message processed successfully

```
Node            : Gateway1
Time            : 20150202 16:28:48.291
Severity        : WARNING
Request Id      : 0000014b4c72bd26-f
Message         : Message processed successfully
Audit Record ID: 7c9b740b74ed7098c4136d8902afc91c

Event Type      : Message Summary
Client IP       : 192.168.137.1
Service         : Voonair [/lab1]
Operation       : voonairreservationDetails
Rqst Length     : 377
Resp Length     : 537
Resp Status     : 200
Resp Time       : 46ms
User ID         : <No ID>
User Name       : null
```

So what does this tell us? Some info, but not all.

LAB 2C: SETTING DEBUG MODE ON

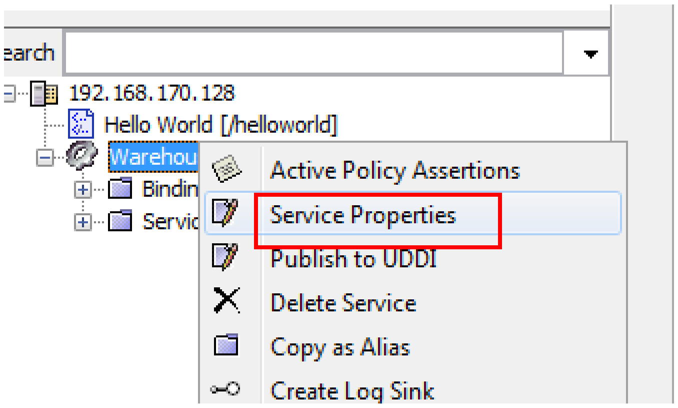
THE SITUATION

Not all is resolved. Dave needs more information on the success AND failure of each assertion in order to troubleshoot the coding issue.

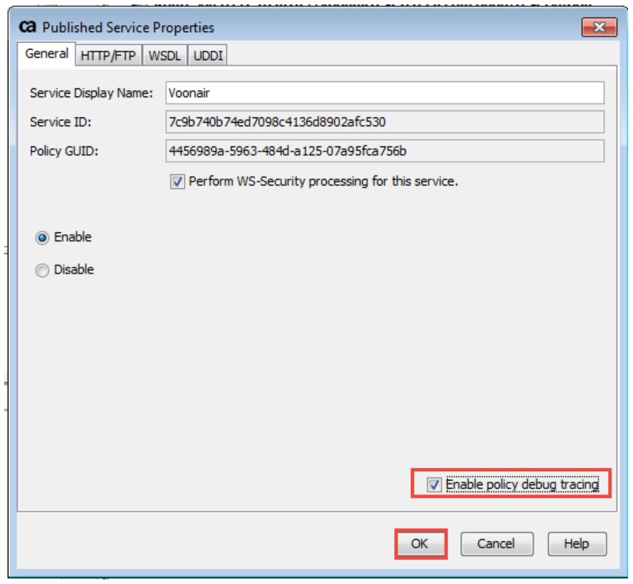
THE SOLUTION

Dave needs to enable the Debug feature

Step 1 – Right click on your existing service & pick "Service Properties"

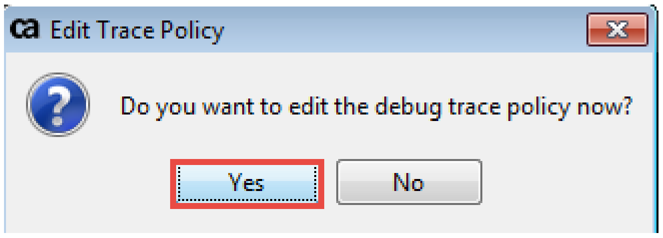


Step 2 – Toggle on the "Enable policy Debug Tracing"

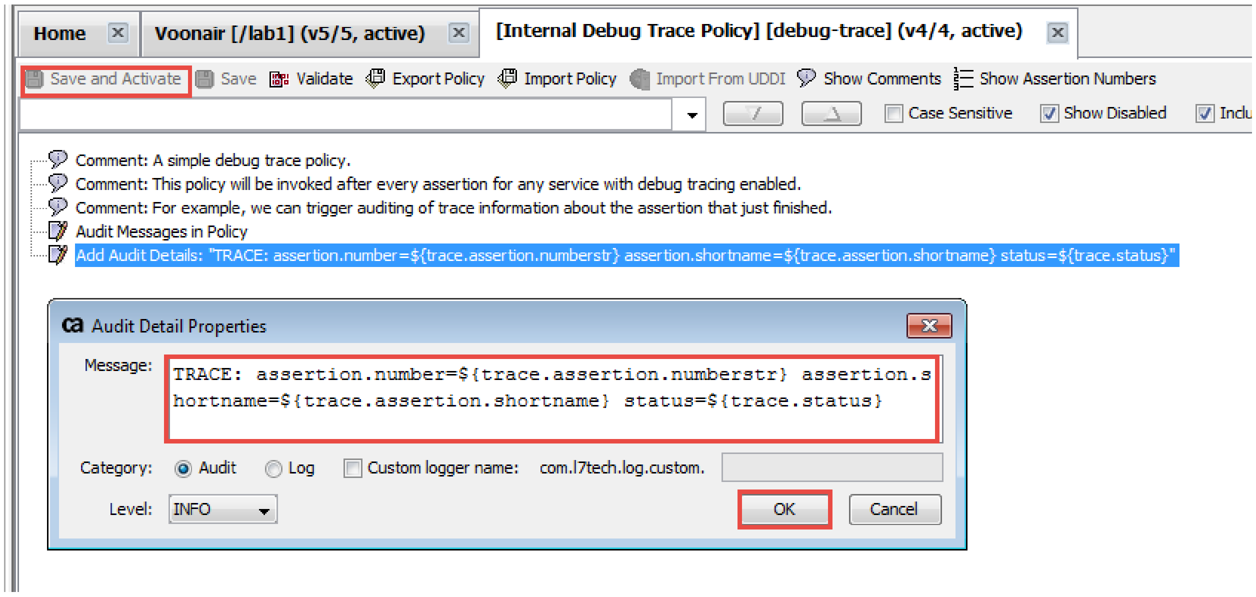


Service Display Name: Voonair
Service ID: 7c9b740b74ed7098c4136d8902afc530
Policy GUID: 4456989a-5963-484d-a125-07a95fca756b
[✓] Perform WS-Security processing for this service.
[● Enable]
Disable
[✓] Enable policy debug tracing

Step 3 – Click Yes to edit



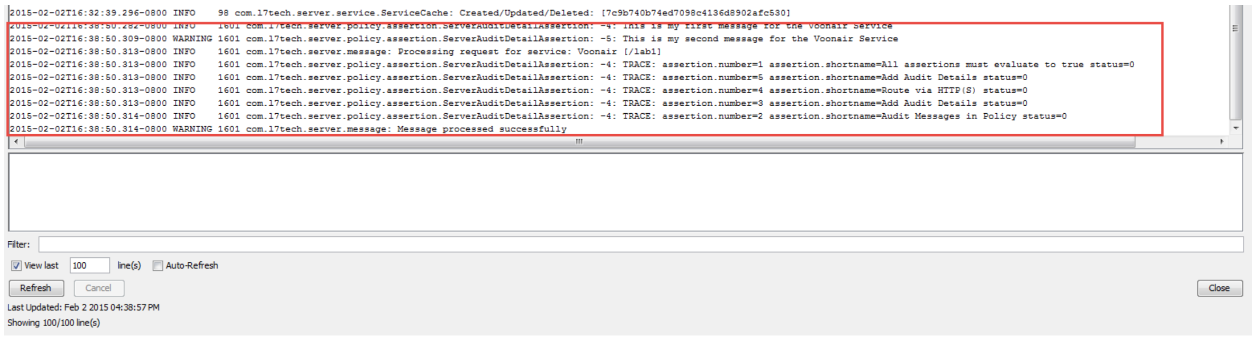
Step 4 – Edit the "Audit Detail" that's highlighted below. When you double click on the assertion, delete the first 3 trace info variables, and make sure the ones that are highlighted below are what's left within the assertion. Click OK and "Save and Activate" the debug trace policy.



Comment: A simple debug trace policy.
Comment: This policy will be invoked after every assertion for any service with debug tracing enabled.
Comment: For example, we can trigger auditing of trace information about the assertion that just finished.
Audit Messages in Policy
Add Audit Details: "TRACE: assertion.number=${trace.assertion.numberstr} assertion.shortname=${trace.assertion.shortname} status=${trace.status}"

Message:
```
TRACE: assertion.number=${trace.assertion.numberstr} assertion.s
hortname=${trace.assertion.shortname} status=${trace.status}
```

Step 5 – Send a soapUI test request and note the additional logging for your service the Policy Manager select "View" -> "View Logs"


```
2015-02-02T16:32:39.296-0800 INFO    98 com.l7tech.server.service.ServiceCache: Created/Updated/Deleted: [7c9b740b74ed7098c4136d8902afc530]
2015-02-02T16:38:50.282-0800 INFO    1601 com.l7tech.server.policy.assertion.ServerAuditDetailAssertion: -4: This is my first message for the Voonair Service
2015-02-02T16:38:50.309-0800 WARNING 1601 com.l7tech.server.policy.assertion.ServerAuditDetailAssertion: -5: This is my second message for the Voonair Service
2015-02-02T16:38:50.313-0800 INFO    1601 com.l7tech.server.message: Processing request for service: Voonair [/lab1]
2015-02-02T16:38:50.313-0800 INFO    1601 com.l7tech.server.policy.assertion.ServerAuditDetailAssertion: -4: TRACE: assertion.number=1 assertion.shortname=All assertions must evaluate to true status=0
2015-02-02T16:38:50.313-0800 INFO    1601 com.l7tech.server.policy.assertion.ServerAuditDetailAssertion: -4: TRACE: assertion.number=5 assertion.shortname=Add Audit Details status=0
2015-02-02T16:38:50.313-0800 INFO    1601 com.l7tech.server.policy.assertion.ServerAuditDetailAssertion: -4: TRACE: assertion.number=4 assertion.shortname=Route via HTTP(S) status=0
2015-02-02T16:38:50.313-0800 INFO    1601 com.l7tech.server.policy.assertion.ServerAuditDetailAssertion: -4: TRACE: assertion.number=3 assertion.shortname=Add Audit Details status=0
2015-02-02T16:38:50.314-0800 INFO    1601 com.l7tech.server.policy.assertion.ServerAuditDetailAssertion: -4: TRACE: assertion.number=2 assertion.shortname=Audit Messages in Policy status=0
2015-02-02T16:38:50.314-0800 WARNING 1601 com.l7tech.server.message: Message processed successfully
```

Last Updated: Feb 2 2015 04:38:57 PM
Showing 100/100 line(s)

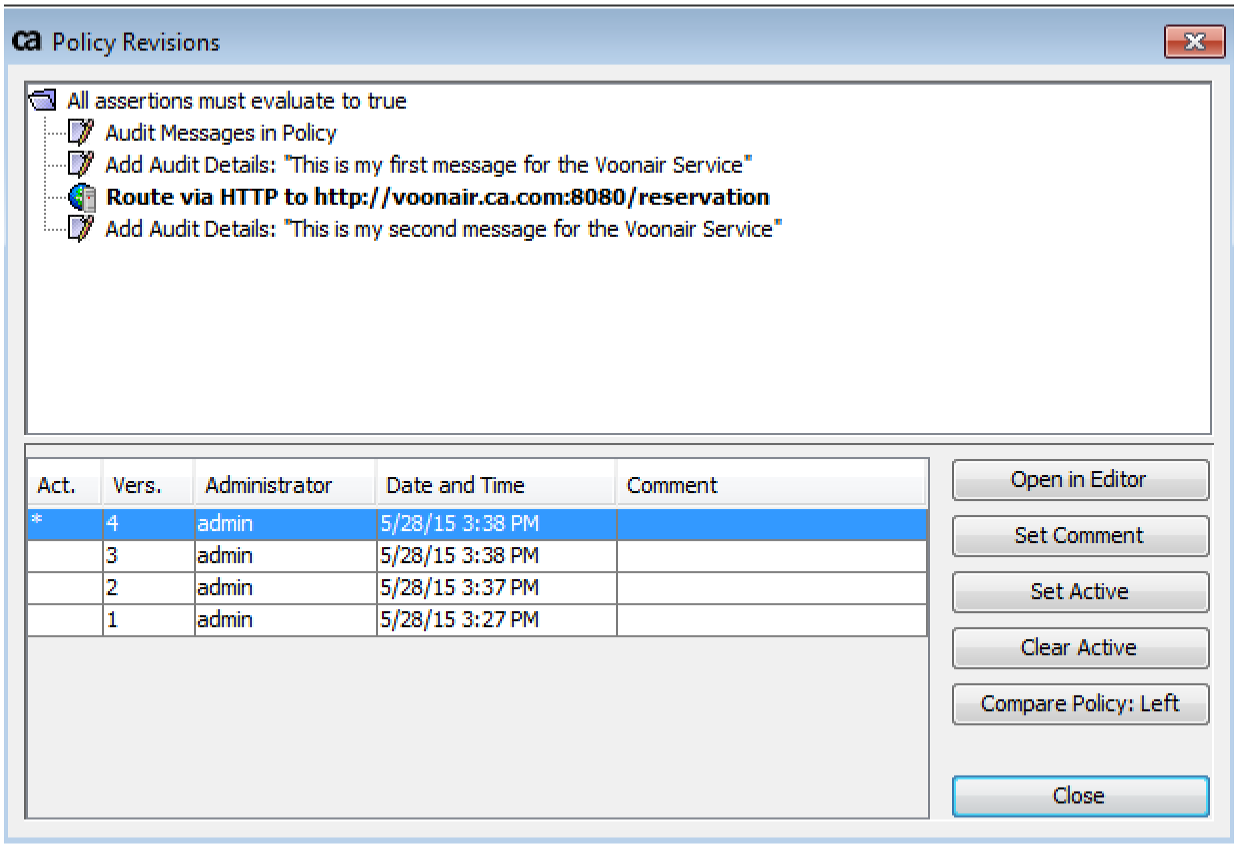
LAB 2D: REVISIONS

THE SITUATION

Dave needs to recover work from a previous version after making some changes that are causing errors in the policy.

THE SOLUTION

Use the policy revisions feature to recover an old version. You have the choice of making it active or just viewing it for reference.



All assertions must evaluate to true
- Audit Messages in Policy
- Add Audit Details: "This is my first message for the Voonair Service"
- **Route via HTTP to http://voonair.ca.com:8080/reservation**
- Add Audit Details: "This is my second message for the Voonair Service"

Act.	Vers.	Administrator	Date and Time	Comment
*	4	admin	5/28/15 3:38 PM	
	3	admin	5/28/15 3:38 PM	
	2	admin	5/28/15 3:37 PM	
	1	admin	5/28/15 3:27 PM	

LAB 2E: EXPORT/IMPORT POLICIES

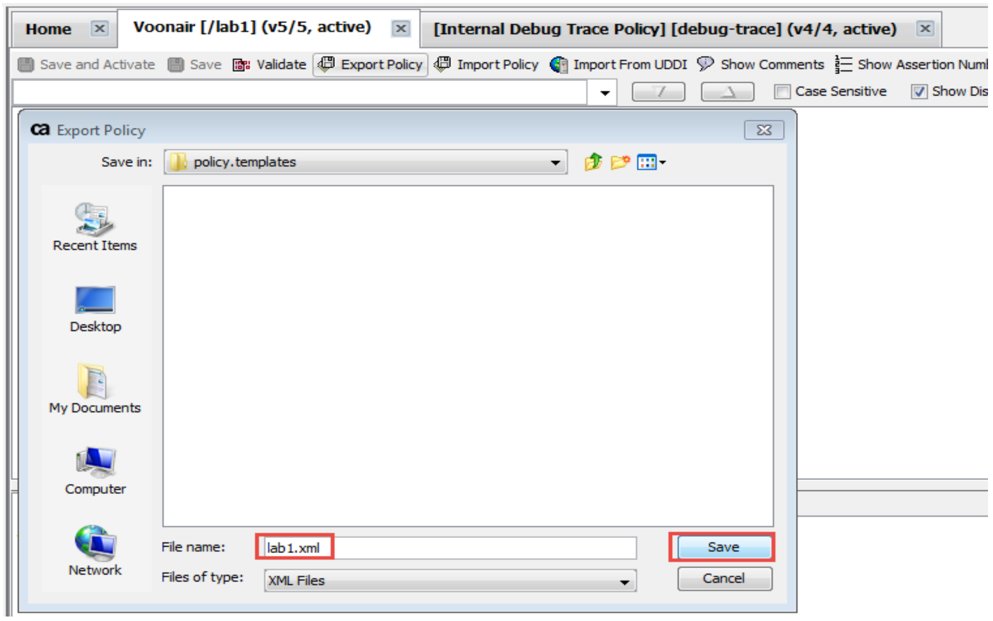
THE SITUATION

Dave heard that there is a library of awesome assertions that he can download from the CA Layer 7 Support Portal. This will save him SO much time. So …

THE SOLUTION

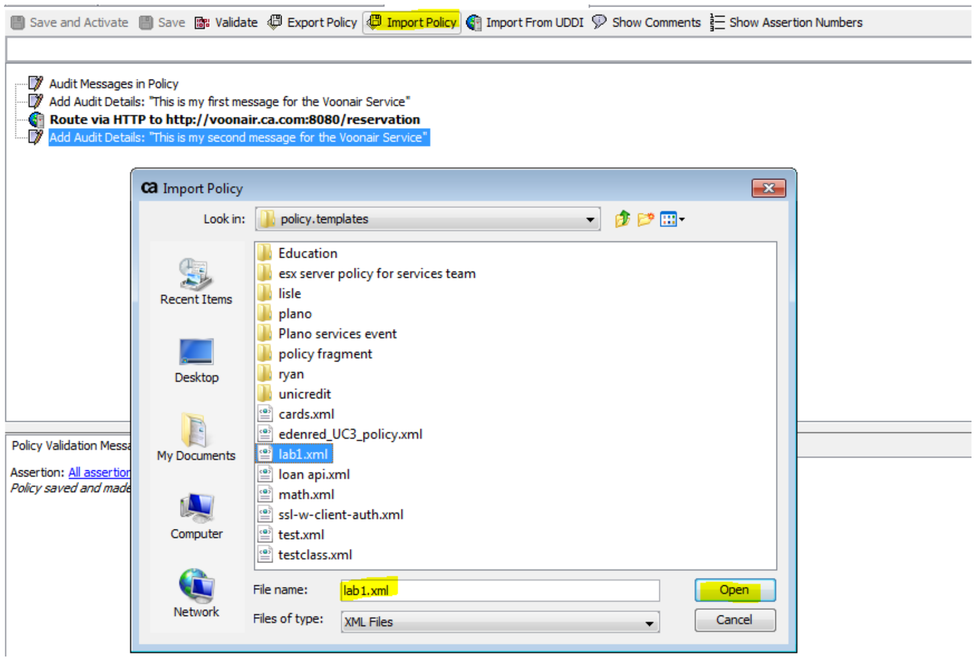
Dave goes to the support portal to download policies. He saves them and then wants to import them into the Policy Manager to view. Show Dave how to export and import policies to make his life easier….

Step 1 – Click on the "Export Policy" and save the name of exported file (e.g. lab1). View the contents of this file in an editor and not that it is a readable XML format.



Step 2 – Make some changes to the current master policy and "Save and Activate" the policy

Step 3 – Click on the "Import Policy" from the upper tool bar and navigate to the previously saved policy.



Audit Messages in Policy
Add Audit Details: "This is my first message for the Voonair Service"
Route via HTTP to http://voonair.ca.com:8080/reservation
Add Audit Details: "This is my second message for the Voonair Service"

Policy Validation Messa
Assertion: All assertion
Policy saved and made

Education
esx server policy for services team
lisle
plano
Plano services event
policy fragment
ryan
unicredit
cards.xml
edenred_UC3_policy.xml
lab1.xml
loan api.xml
math.xml
ssl-w-client-auth.xml
test.xml
testclass.xml

Step 4 – note that the changes you made are removed and you have recovered your original version that was exported. Click “Save and Activate”.

LAB 2F: SERVICE DEBUGGER

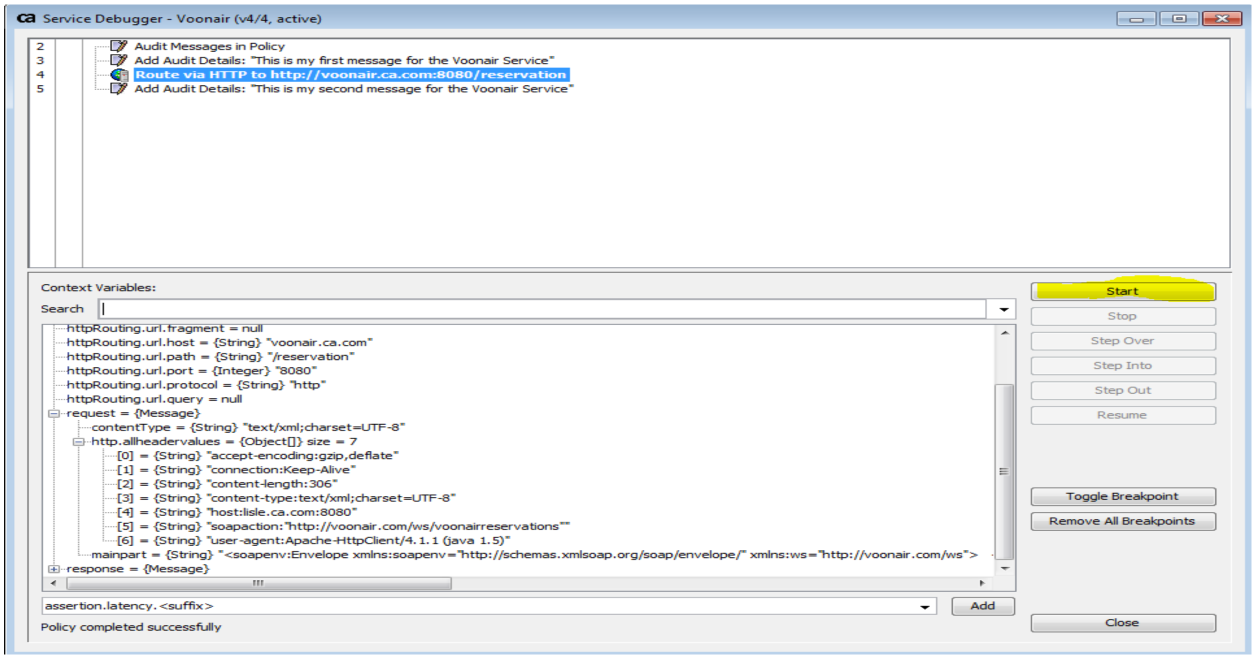
THE SITUATION

Dave needs to be able to look at different break points within policy. He needs to know what variables that are being set, which path the policy is taking and be able to follow the different encapsulated assertions that get called within policy. So if the policy refers to a policy fragment / encapsulated assertion than the service debugger will follow through to those outside of policy.

THE SOLUTION

Dave wants to increase his troubleshooting capabilities by wanting to see what happens during runtime. He would like to follow the path of the service to see each assertion being invoked during runtime.

1. When wanting to do this Dave will need to right click on the service in the policy/service area click “service debugger”.
2. Click on Start to set the debugger to run for a test case
3. Run a request through to that service and watch the bottom window fill up with variables and flow of the policy assertions.


Service Debugger - Voonair (v4/4, active)

```
2   Audit Messages in Policy
3   Add Audit Details: "This is my first message for the Voonair Service"
4   Route via HTTP to http://voonair.ca.com:8080/reservation
5   Add Audit Details: "This is my second message for the Voonair Service"
```

Context Variables:

Search

```
httpRouting.url.fragment = null
httpRouting.url.host = {String} "voonair.ca.com"
httpRouting.url.path = {String} "/reservation"
httpRouting.url.port = {Integer} "8080"
httpRouting.url.protocol = {String} "http"
httpRouting.url.query = null
request = {Message}
    contentType = {String} "text/xml;charset=UTF-8"
    http.allheadervalues = {Object[]} size = 7
        [0] = {String} "accept-encoding:gzip,deflate"
        [1] = {String} "connection:Keep-Alive"
        [2] = {String} "content-length:306"
        [3] = {String} "content-type:text/xml;charset=UTF-8"
        [4] = {String} "host:lisle.ca.com:8080"
        [5] = {String} "soapaction:"http://voonair.com/ws/voonairreservations""
        [6] = {String} "user-agent:Apache-HttpClient/4.1.1 (java 1.5)"
    mainpart = {String} "<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://voonair.com/ws">
response = {Message}
```

assertion.latency.<suffix>

Policy completed successfully

MODULE 3: MESSAGE ROUTING & ERROR HANDLING

LAB 3A: CREATE TEMPLATE RESPONSE

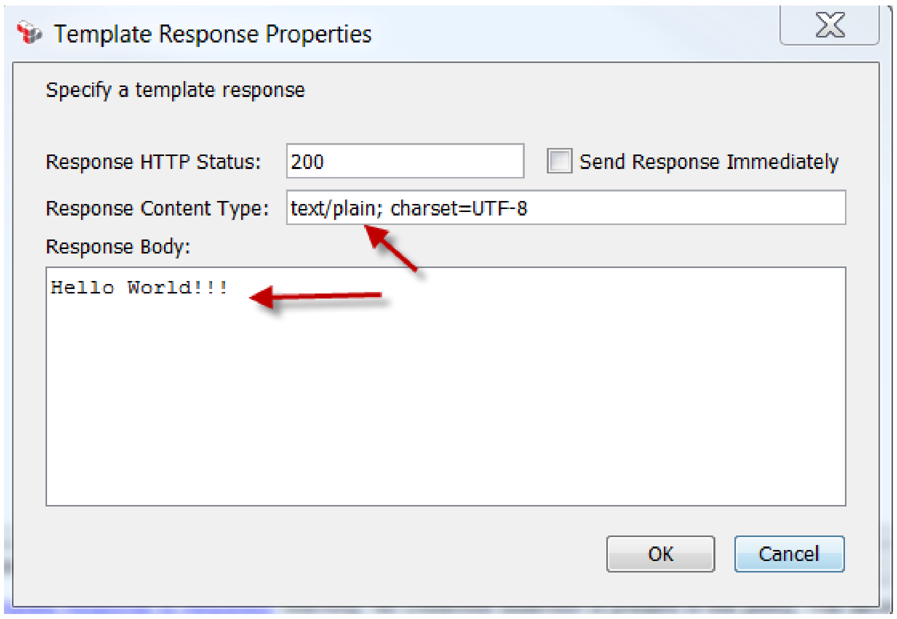
THE SITUATION

Dave needs to return a custom response to the client application to fulfill the business requirements for the new Voonair WebService

THE SOLUTION

There is a feature in the Policy Manager known as the **return template response**. Use it within an error condition to aid in customizing a response or troubleshooting.

Step 1 – Within your current service created in earlier lab add the "Return Template Response to Requestor" by dragging and dropping the assertion into your policy. Make sure it is after the current "HTTP/S Route" assertion so it overrides the default response.



Template Response Properties

Specify a template response

Response HTTP Status: 200

Send Response Immediately: [ ]

Response Content Type: text/plain; charset=UTF-8

Response Body:

```
Hello World!!!
```

Step 2 – Make sure the template response properties matches the above. Response Content Type = text/plain; charset=UTF-8. Response Body = Hello World!!!

Step 3 – Save and Activate. Send a request to the service from SOAPUI and note the response you get returned now has the custom text you wanted returned.

LAB 3B: HTTP ROUTING ASSERTION

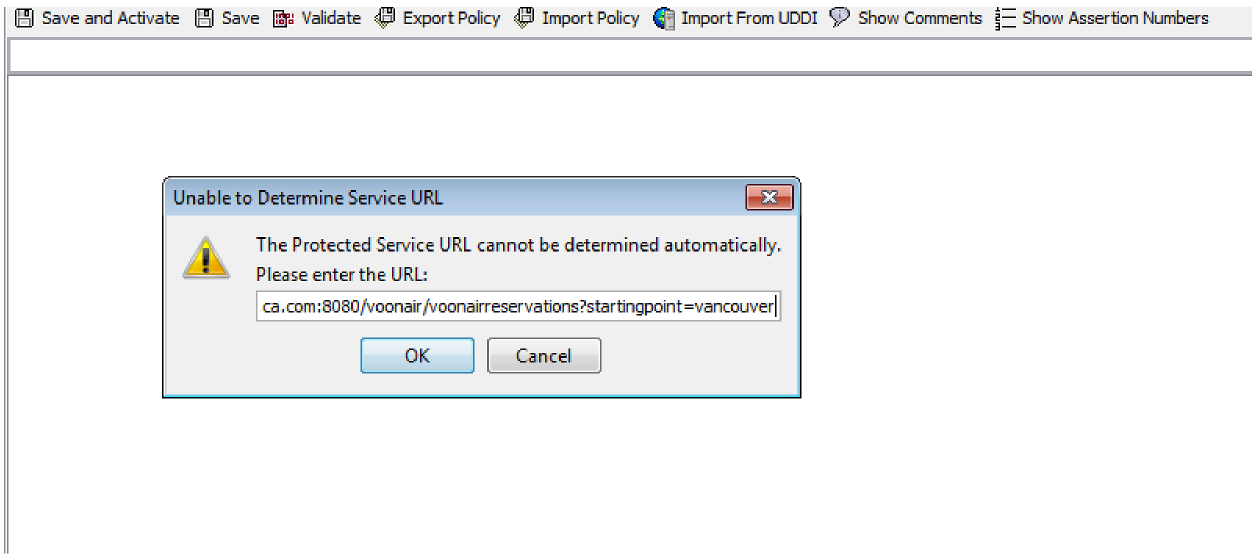
THE SITUATION

Dave needs to send a certain request off to the voonair web service.

THE SOLUTION

The Http Routing assertion allows Dave to direct a particular service request to a specific voonair web service.

Step 1 – Create a new “Publish Web API” service and navigate to the “Message Routing” assertion folder. Highlight “Route via http(s)”. Drag the assertion into your policy development window.



Unable to Determine Service URL

The Protected Service URL cannot be determined automatically.
Please enter the URL:

ca.com:8080/voonair/voonairreservations?startingpoint=vancouver

Place the location of the voonair rest service.

For example = http://voonair.ca.com:8080/voonair/voonairreservations?startingpoint=vancouver

Step 2 – Save and Activate

Step 3 – Using your rest client run through a request

LAB 3C: CUSTOMIZE AN ERROR RESPONSE

THE SITUATION

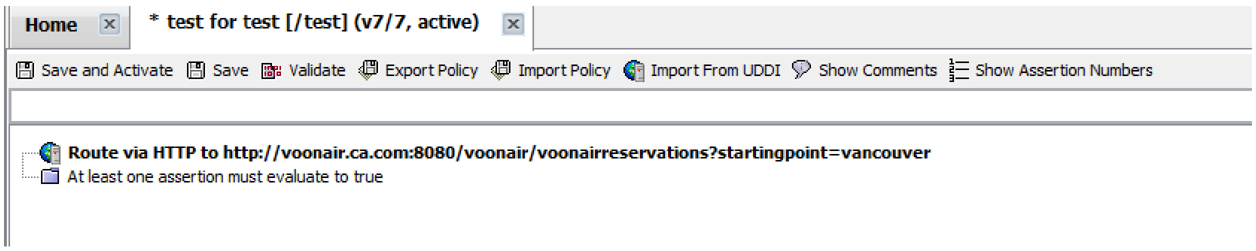
Dave needs to return a custom error response to the client application to fulfill the business requirements for the new Voonair WebService. Dave's requirement is to route to a particular back end service but if it doesn't respond than he needs to send the client a proper error message. He also needs to fail the policy to alert the proper teams of the failure.

THE SOLUTION

There is a feature in the Policy Manager known as the "**Customize Error Response**". Dave can use it within an error condition to aid in customizing a response or troubleshooting. The above assertion plus a failure within policy (stop processing assertion) will provide exactly what Dave is requiring.

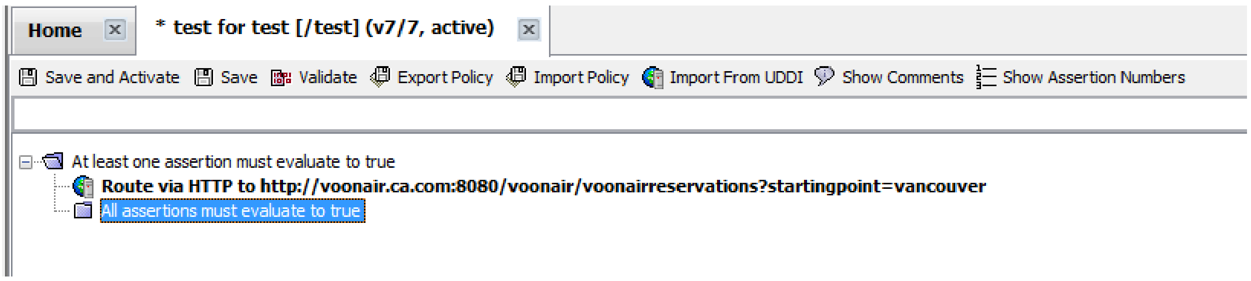
Step 1 – Add to the existing service that you had built for the routing assertion example.

Step 2 – Highlight the routing assertion within policy and right click. Select "Add At least one assertion".



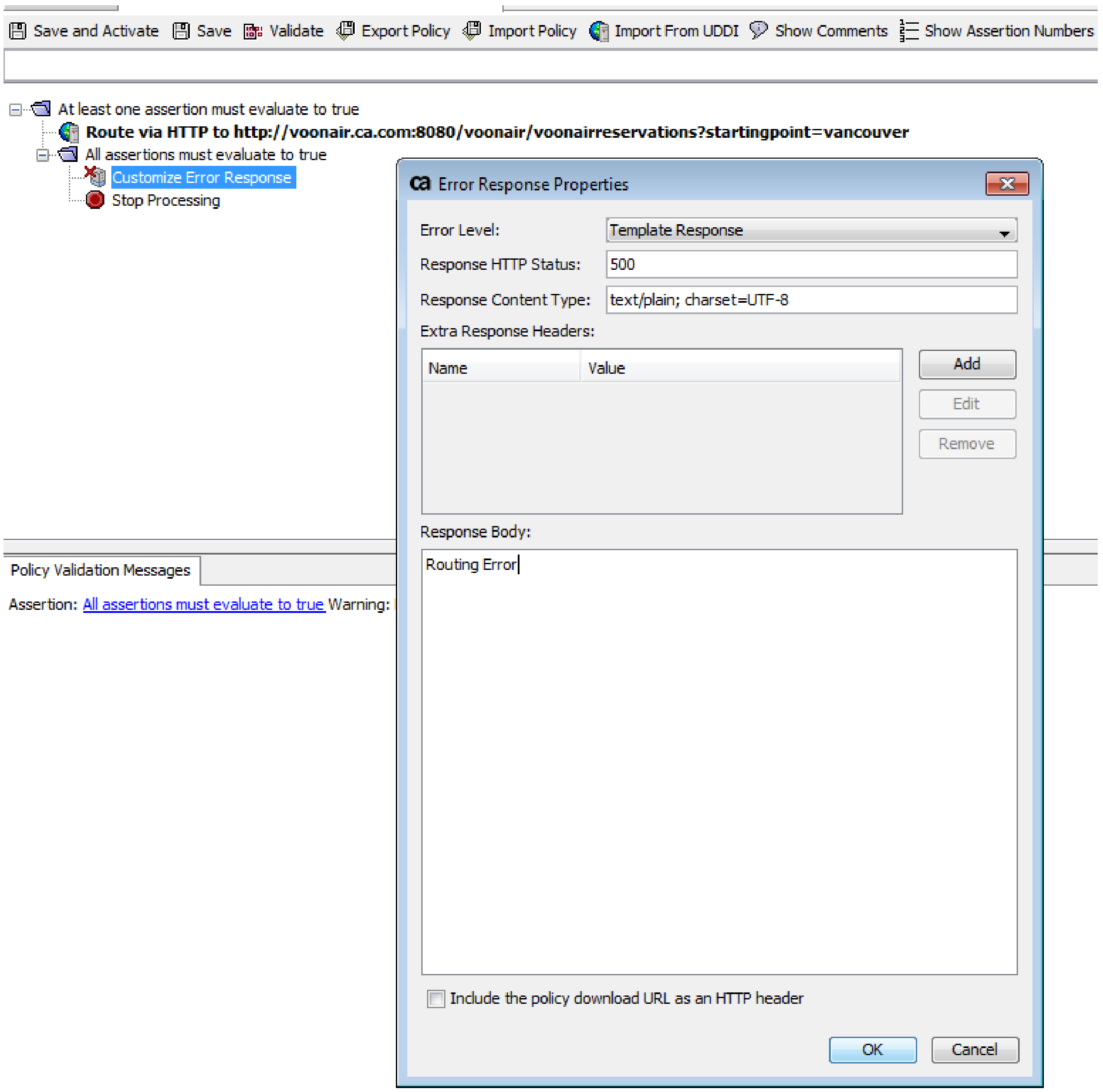
Route via HTTP to http://voonair.ca.com:8080/voonair/voonairreservations?startingpoint=vancouver
At least one assertion must evaluate to true

Step 3 – Drag the "Route via HTTP://..." assertion into the "At least one.." folder. Highlight the "At least one folder" and click Add "And ALL..folder".



At least one assertion must evaluate to true
Route via HTTP to http://voonair.ca.com:8080/voonair/voonairreservations?startingpoint=vancouver
All assertions must evaluate to true

Step 4 – Within the policy assertion pallet search field do a look up "stop processing". Highlight and drag into the all assertion folder. Next is to do a lookup of the "Customize Error Response" within the assertion pallet as well. Highlight and drag the assertion into the all folder, above the stop processing assertion. Within the Response Body of the Customize Error Response, type "Routing Error" and make sure the Error Level is a "Template Response" with a response content type of "test/plain…"


Save and Activate | Save | Validate | Export Policy | Import Policy | Import From UDDI | Show Comments | Show Assertion Numbers

At least one assertion must evaluate to true
- **Route via HTTP to http://voonair.ca.com:8080/voonair/voonairreservations?startingpoint=vancouver**
- All assertions must evaluate to true
  - Customize Error Response
  - Stop Processing

Policy Validation Messages

Assertion: All assertions must evaluate to true Warning:

Error Response Properties

Error Level: Template Response

Response HTTP Status: 500

Response Content Type: text/plain; charset=UTF-8

Extra Response Headers:

Name	Value

Add | Edit | Remove

Response Body:

Routing Error

Include the policy download URL as an HTTP header

OK | Cancel

Step 5 – Save and Activate the service. Do a successful route test and then a failed attempt. To invoke a failure just make an adjustment to the Routing statement. (ie., change the port number from 8080 to 8000.)

MODULE 4: IDM & RBAC

LAB 4A: CREATE USERS

THE SITUATION

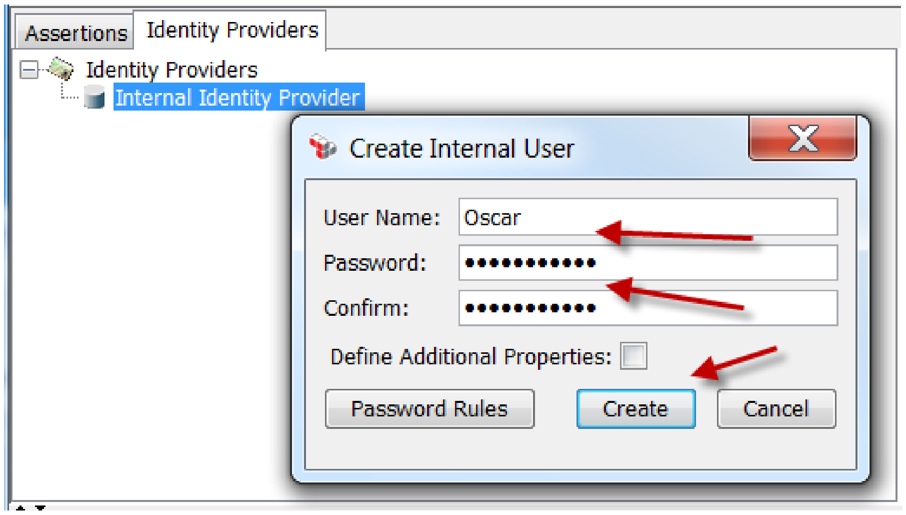
Operations Oscar needs to create users within the Internal Identity Provider in order for them to access the Policy Manager.

THE SOLUTION

Create users within the Policy Manager with the Internal Identity feature.

Step 1: Log into the Policy Manager. Select Tasks / Create Internal User

Step 2: Create 3 different users – Oscar / Andy / Dave



User Name: Oscar
Password: ••••••••••••
Confirm: ••••••••••••
Define Additional Properties: ☐

- User Name = Oscar
- Password = L7Secure$0@
- Repeat for the 2 other team members.

Step 3: Create a **group** with the three users you have created. Most often identity management is done with a group so single users are not required to be managed in a master policy.

LAB 4B: CREATE LDAP RESOURCE

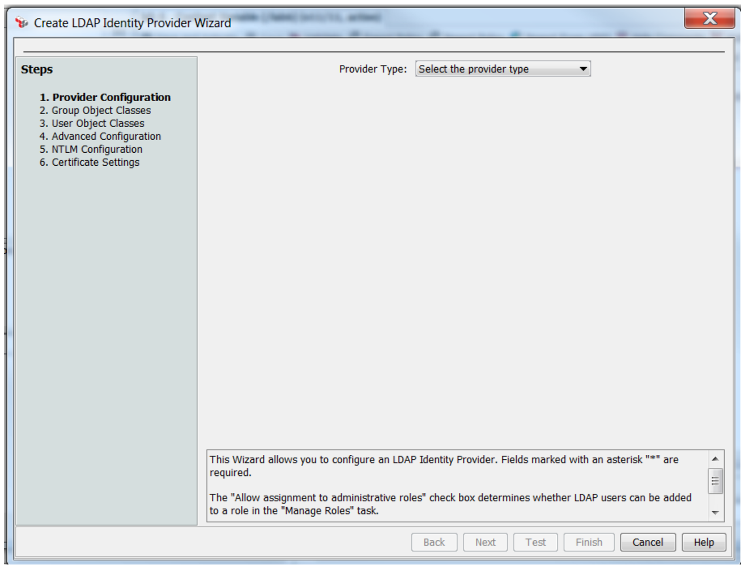
THE SITUATION

Users within Voonair need to be authenticated. Their LDAP connection will be used for both authentications within policy runtime and accessing the Policy Manager moving forward.

THE SOLUTION

Oscar needs to set up an LDAP connection to their internal directory service.

Step 1 – To create an LDAP identity provider for the SDE environment, you must select – Tasks / Create Identity Provider / Create LDAP Identity Provider.



Provider Type: Select the provider type

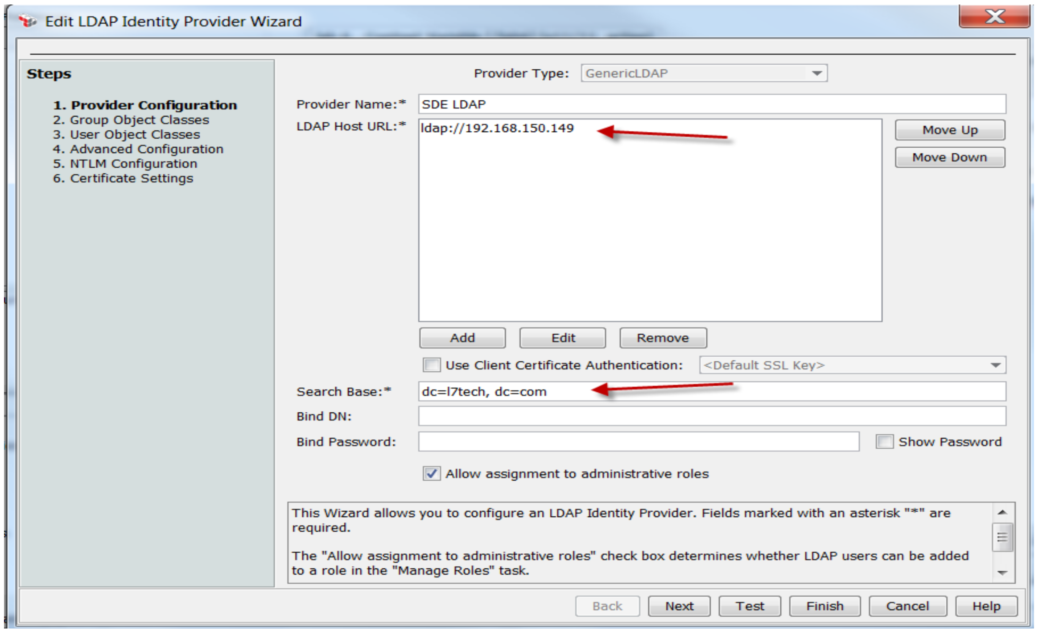
This Wizard allows you to configure an LDAP Identity Provider. Fields marked with an asterisk "*" are required.

The "Allow assignment to administrative roles" check box determines whether LDAP users can be added to a role in the "Manage Roles" task.

Step 2 – In the drop down arrow "Provider Type", select GenericLDAP.

Step 3 – Give the LDAP connection parameters specific to your gateway configured:

- Host URL = ldap://your2ndGatewayIP
- Search Base = dc=l7tech, dc=com
- Make Sure the "Allow assignment to administrative roles" box is checked.



Provider Type: GenericLDAP
Provider Name:* SDE LDAP
LDAP Host URL:* ldap://192.168.150.149
☐ Use Client Certificate Authentication: <Default SSL Key>
Search Base:* dc=l7tech, dc=com
Bind DN:
Bind Password:
☑ Allow assignment to administrative roles

This Wizard allows you to configure an LDAP Identity Provider. Fields marked with an asterisk "*" are required.

The "Allow assignment to administrative roles" check box determines whether LDAP users can be added to a role in the "Manage Roles" task.

LAB 4C: ROLE BASED ACCESS CONTROL

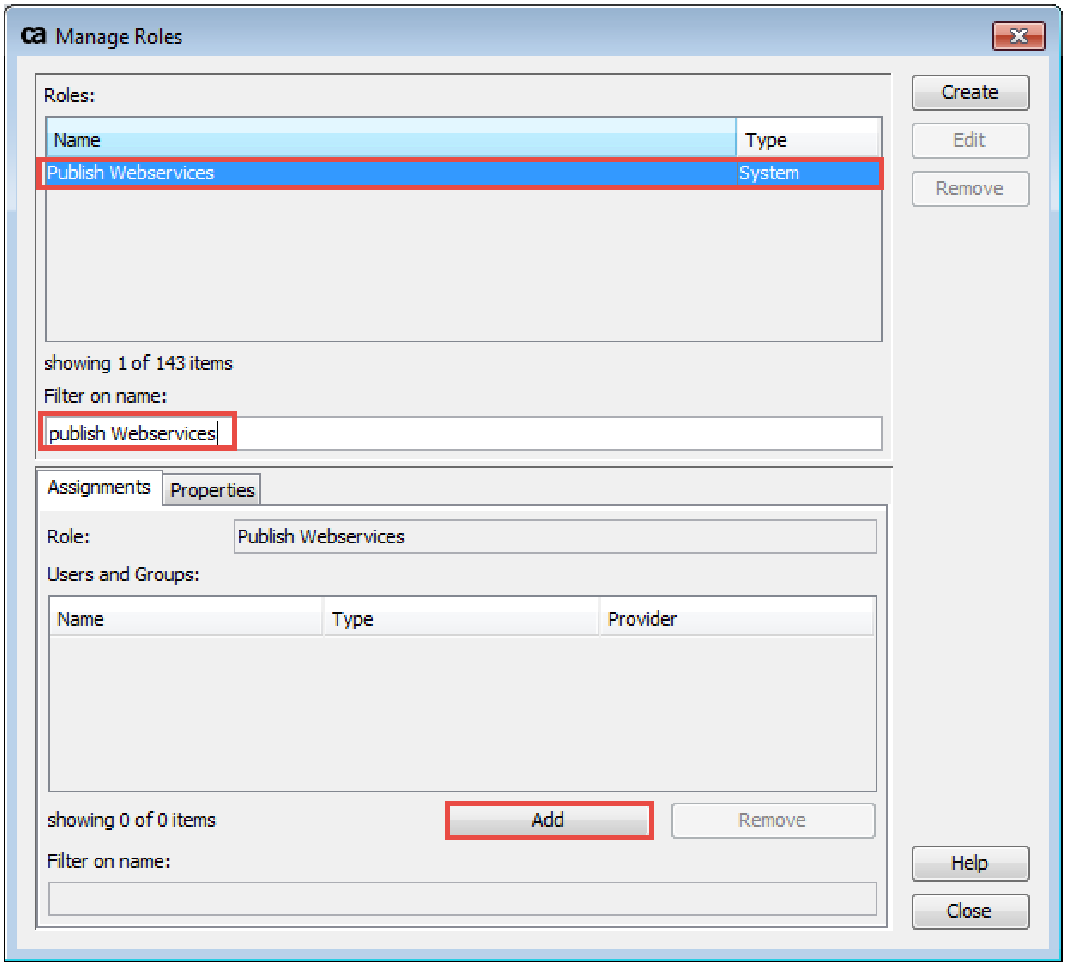
THE SITUATION

Architect Andy phones Operations Ollie. He wants to restrict access for developers to specific folders. He's worried they may accidently change some of Ollie's critical configuration for LDAP.

THE SOLUTION

RBAC, or 'Role Based Assignment' is possible within the Gateway. Help Ollie to limit the developers access to developer functional roles like "Publish Webservices"

Step 1 – From the Menu Item **Tasks** -> **Manage Roles**, use LDAP server and "Consultants" group so they can access the policy manager in a restricted view. Give this group the role of "Publish WebServices" and "Gateway Maintenance". Type "Publish Webservices" in the "Filter on Name: box". Highlight Publish Webservices and click "Add" at the bottom.



Roles:

Name	Type
Publish Webservices	System

showing 1 of 143 items

Filter on name: publish Webservices

Role: Publish Webservices

Users and Groups:

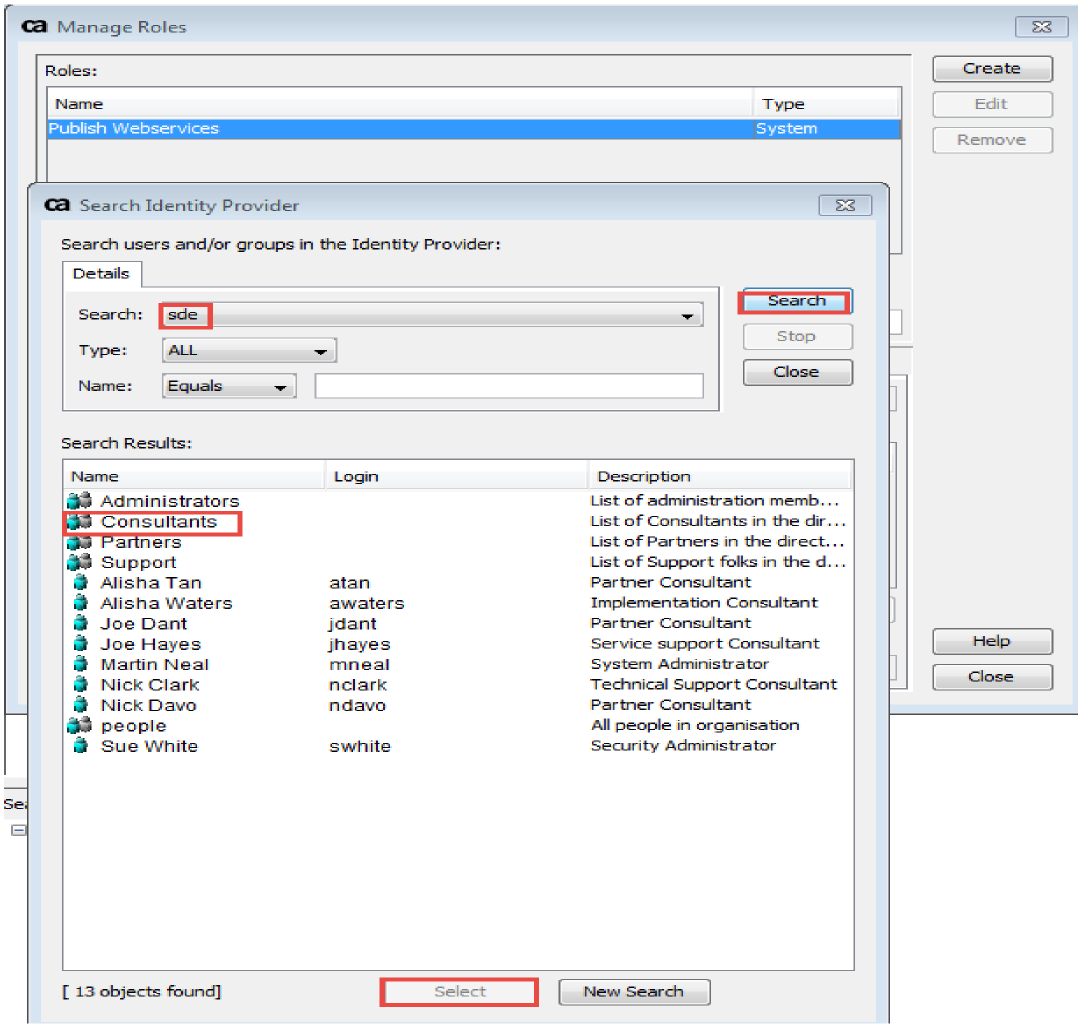
Name	Type	Provider

showing 0 of 0 items

Filter on name:

Step 2 – In the Search menu, select the sde ldap from the list. Press Search.

Select “Consultants” and press “Select” at the bottom.



Manage Roles

Roles:

Name	Type
Publish Webservices	System

Search Identity Provider

Search users and/or groups in the Identity Provider:

Details

Search: sde

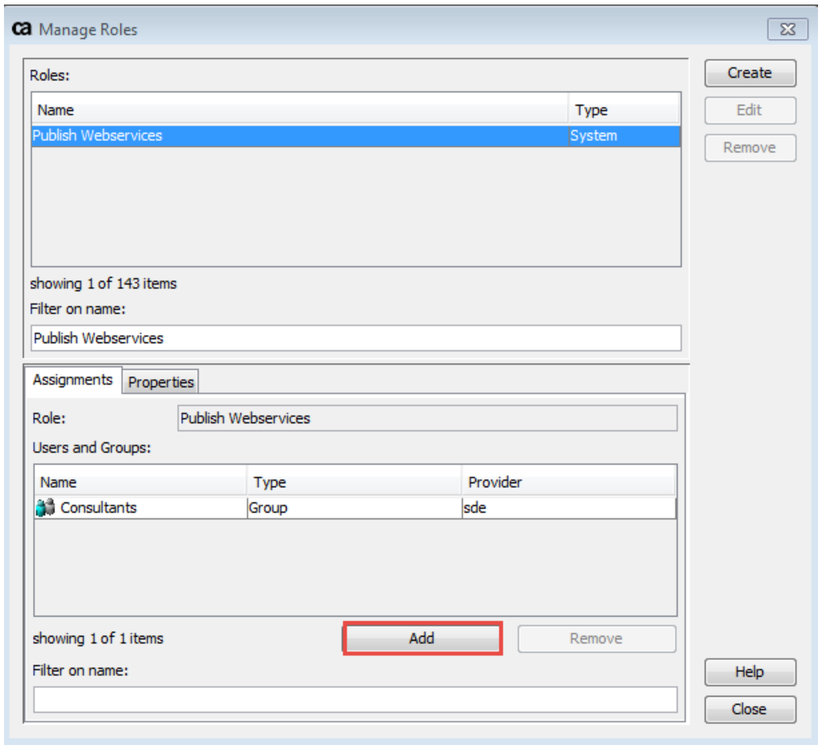
Type: ALL

Name: Equals

Search Results:

Name	Login	Description
Administrators		List of administration memb…
Consultants		List of Consultants in the dir…
Partners		List of Partners in the direct…
Support		List of Support folks in the d…
Alisha Tan	atan	Partner Consultant
Alisha Waters	awaters	Implementation Consultant
Joe Dant	jdant	Partner Consultant
Joe Hayes	jhayes	Service support Consultant
Martin Neal	mneal	System Administrator
Nick Clark	nclark	Technical Support Consultant
Nick Davo	ndavo	Partner Consultant
people		All people in organisation
Sue White	swhite	Security Administrator

[13 objects found]



Roles:

Name	Type
Publish Webservices	System

showing 1 of 143 items

Filter on name: Publish Webservices

Role: Publish Webservices

Users and Groups:

Name	Type	Provider
Consultants	Group	sde

showing 1 of 1 items

Filter on name:

Step 3 – Repeat for “Gateway Maintenance”

Step 4 - Disconnect from the Policy Manager and login as ***awaters*** (7layer) and see restricted permissions.

MODULE 5: POLICY LOGIC

LAB 5A: CLUSTER WIDE PROPERTIES

THE SITUATION

Operations Oscar needs a better approach to maintain global environment variables. Instead of having to update multiple services, he would prefer to update one area.

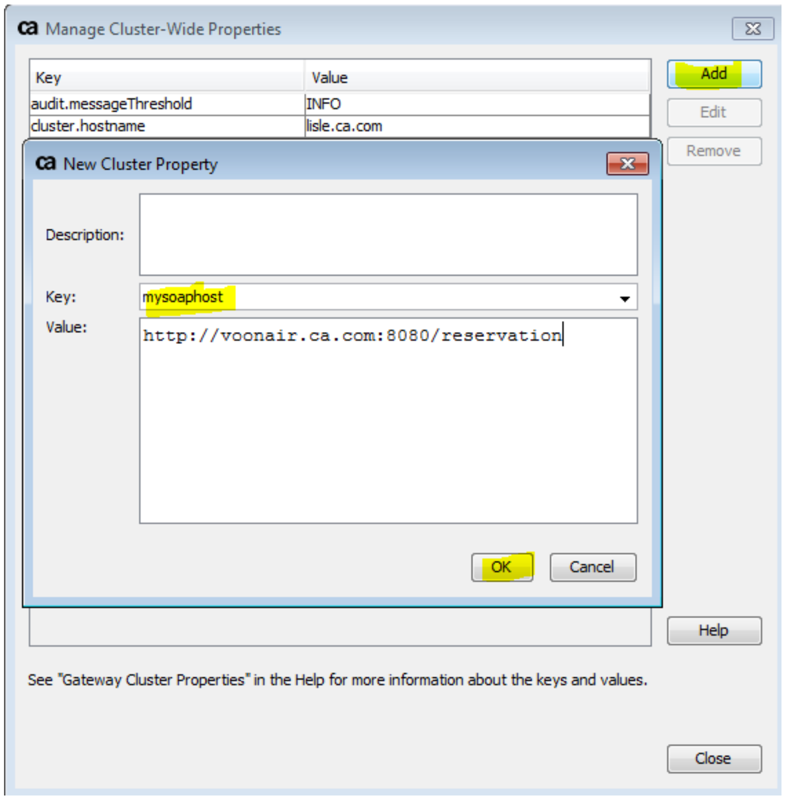
THE SOLUTION

You can do this by managing cluster-wide properties. Show Oscar how to manage environments by adding a cluster wide property to your policy routing.

Step 1 – Select "Tasks", "Manage Cluster-Wide Property" and click Add.

Step 2 – Highlight what's in the key and delete. Tab to the value field, which will erase all that's there.

Step 3 – In the key field, type ***mysoaphost***. Tab to the value field and place your 2nd gateway acting as your back end web service.



Key	Value
audit.messageThreshold	INFO
cluster.hostname	lisle.ca.com

Description:

Key: mysoaphost

Value: http://voonair.ca.com:8080/reservation

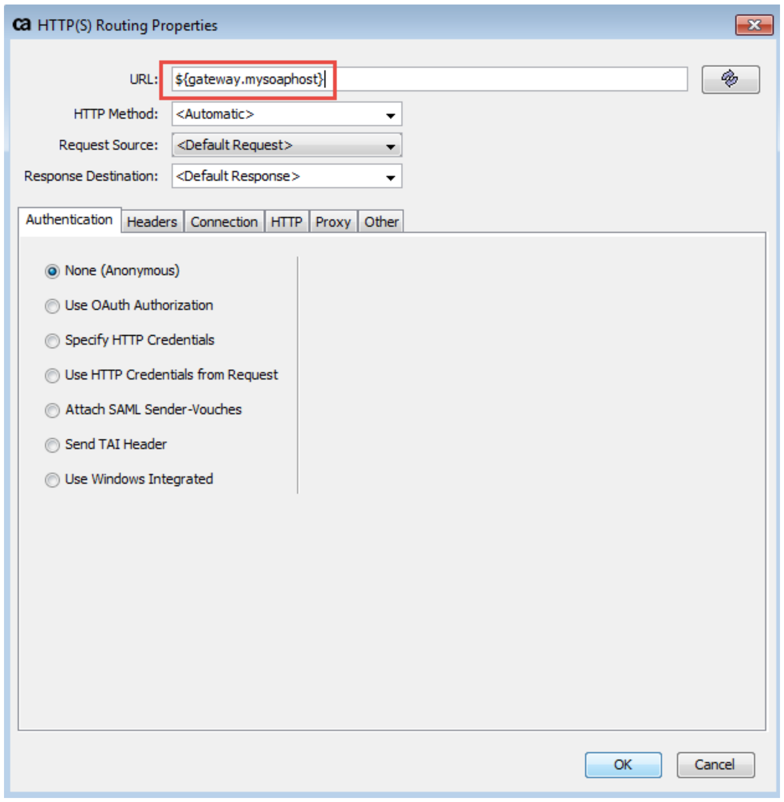
See "Gateway Cluster Properties" in the Help for more information about the keys and values.

Step 4 – Click Ok.

Step 5 – Create another Cluster wide property and name it "***myresthost*".**

Use the following Value = http://voonair.ca.com:8080/voonair/voonairreservations?startingpoint=montreal

Step 6 – Within your existing published service click on the "Route via HTTP to …..". Update the URL of the route with the new cluster wide property. Place the context variable calling the cluster wide property ${gateway.mysoaphost} in the URL. The prefix reference "gateway." always is used to identify a cluster wide property.



URL: ${gateway.mysoaphost}
HTTP Method: <Automatic>
Request Source: <Default Request>
Response Destination: <Default Response>

- [✓] None (Anonymous)
- Use OAuth Authorization
- Specify HTTP Credentials
- Use HTTP Credentials from Request
- Attach SAML Sender-Vouches
- Send TAI Header
- Use Windows Integrated

Step 6 – Save and Activate. Send a request from SOAPUI to the service and check that you get a successful response.

LAB 5B: CONTEXT VARIABLES

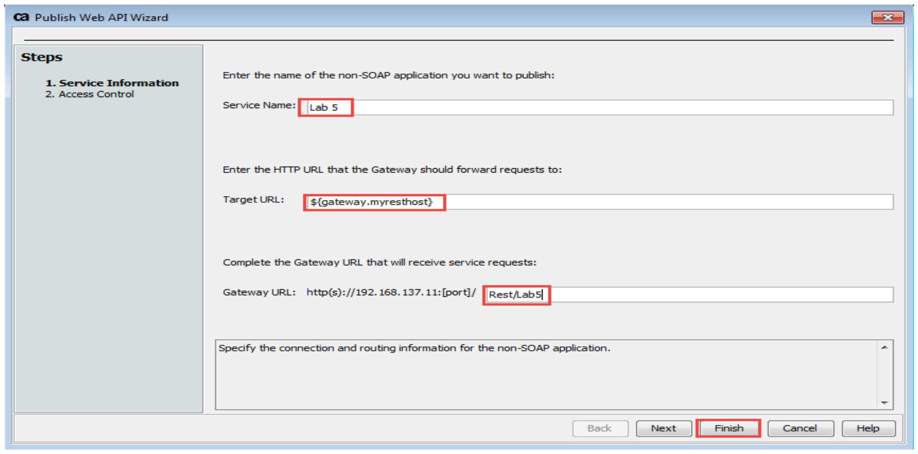
THE SITUATION

Dave needs to find out what variables are being set during run-time and also needs to know how to create context variables during this policy runtime.

THE SOLUTION

This lab walks you through how to create and reference context variables.

Step 1 — Create a new REST service by using "Publish Web API". The new service name should be ***/Rest/Lab5***.



Enter the name of the non-SOAP application you want to publish:

Service Name: Lab 5

Enter the HTTP URL that the Gateway should forward requests to:

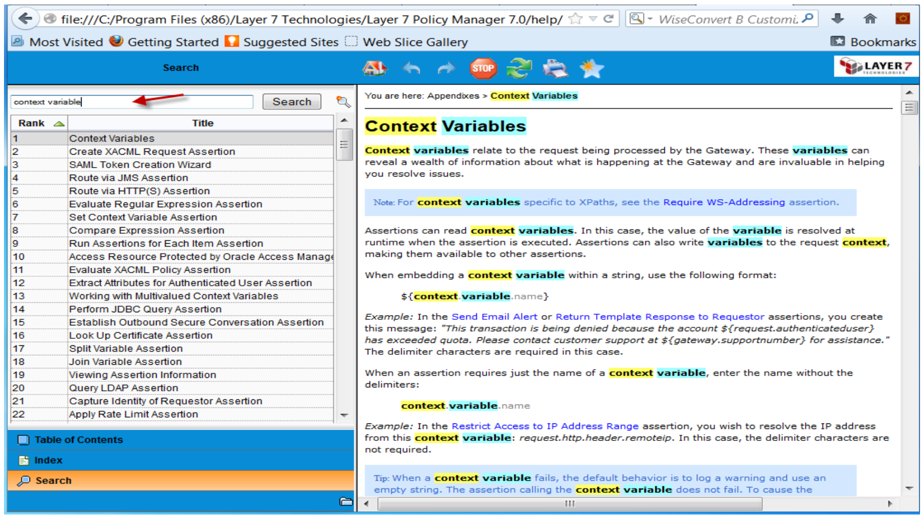
Target URL: ${gateway.myresthost}

Complete the Gateway URL that will receive service requests:

Gateway URL: http(s)://192.168.137.11:[port]/ Rest/Lab5

Specify the connection and routing information for the non-SOAP application.

Step 2 – Press F1, to go to the Help menu for the layer 7 gateway. In the Search window, type context variable and click on "Search".



context variable | Search

Rank	Title
1	Context Variables
2	Create XACML Request Assertion
3	SAML Token Creation Wizard
4	Route via JMS Assertion
5	Route via HTTP(S) Assertion
6	Evaluate Regular Expression Assertion
7	Set Context Variable Assertion
8	Compare Expression Assertion
9	Run Assertions for Each Item Assertion
10	Access Resource Protected by Oracle Access Manage
11	Evaluate XACML Policy Assertion
12	Extract Attributes for Authenticated User Assertion
13	Working with Multivalued Context Variables
14	Perform JDBC Query Assertion
15	Establish Outbound Secure Conversation Assertion
16	Look Up Certificate Assertion
17	Split Variable Assertion
18	Join Variable Assertion
19	Viewing Assertion Information
20	Query LDAP Assertion
21	Capture Identity of Requestor Assertion
22	Apply Rate Limit Assertion

You are here: Appendixes > Context Variables

Context Variables

Context variables relate to the request being processed by the Gateway. These variables can reveal a wealth of information about what is happening at the Gateway and are invaluable in helping you resolve issues.

Note: For context variables specific to XPaths, see the Require WS-Addressing assertion.

Assertions can read context variables. In this case, the value of the variable is resolved at runtime when the assertion is executed. Assertions can also write variables to the request context, making them available to other assertions.

When embedding a context variable within a string, use the following format:

${context.variable.name}

Example: In the Send Email Alert or Return Template Response to Requestor assertions, you create this message: *"This transaction is being denied because the account ${request.authenticateduser} has exceeded quota. Please contact customer support at ${gateway.supportnumber} for assistance."* The delimiter characters are required in this case.

When an assertion requires just the name of a context variable, enter the name without the delimiters:

context.variable.name

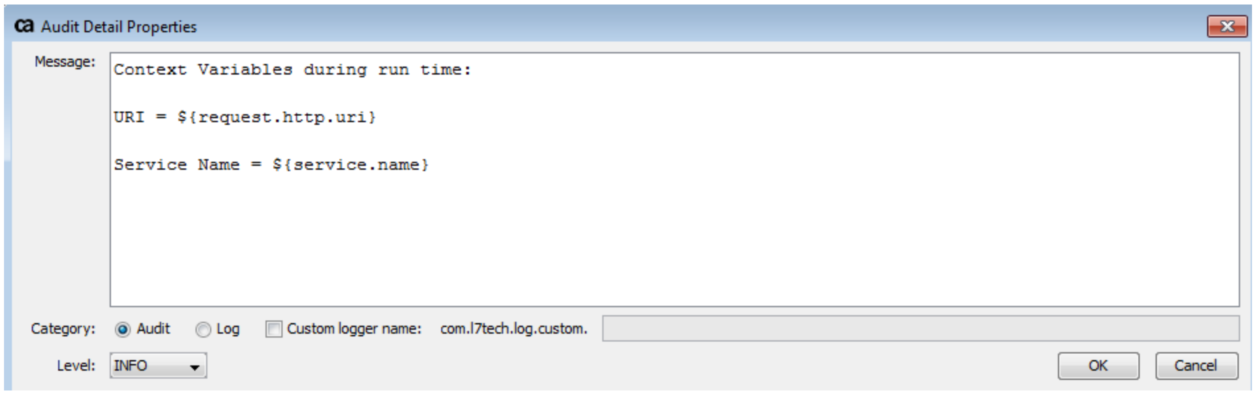
Example: In the Restrict Access to IP Address Range assertion, you wish to resolve the IP address from this context variable: *request.http.header.remoteip*. In this case, the delimiter characters are not required.

Tip: When a context variable fails, the default behavior is to log a warning and use an empty string. The assertion calling the context variable does not fail. To cause the

Step 3 – Look up Message Layer Variables and Message Routing variables – Use the following variables:

- ${request.http.uri} ${service.name}

Step 4 – Find in the assertion palette "Add Audit Details". Drag this into above the route vis http assertion.



```
Context Variables during run time:

URI = ${request.http.uri}

Service Name = ${service.name}
```

Category: [✓ Audit] Log ☐ Custom logger name: com.l7tech.log.custom.

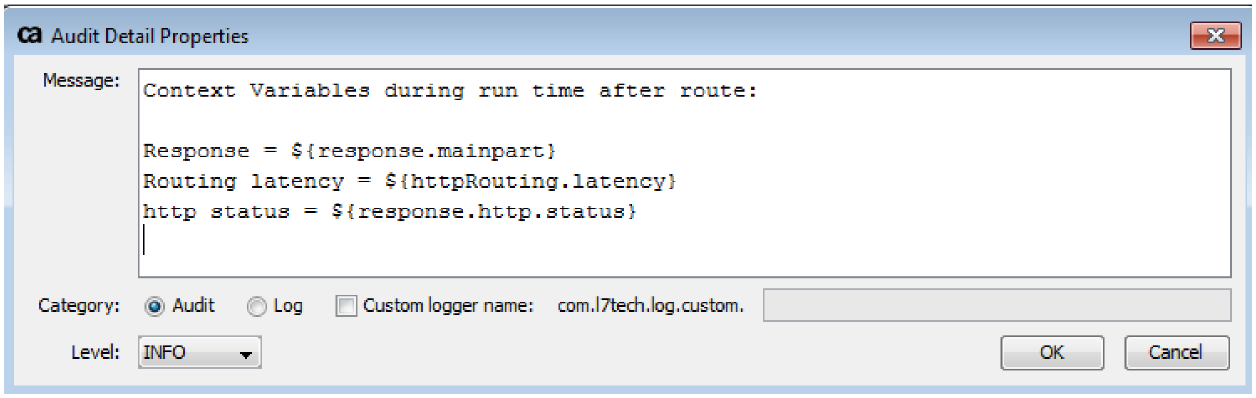
Level: INFO

Drag one more “Add Audit Details” after the route. This time using referencing the response message.

Response = ${response.mainpart}

Routing latency = ${httpRouting.latency}

http status = ${response.http.status}

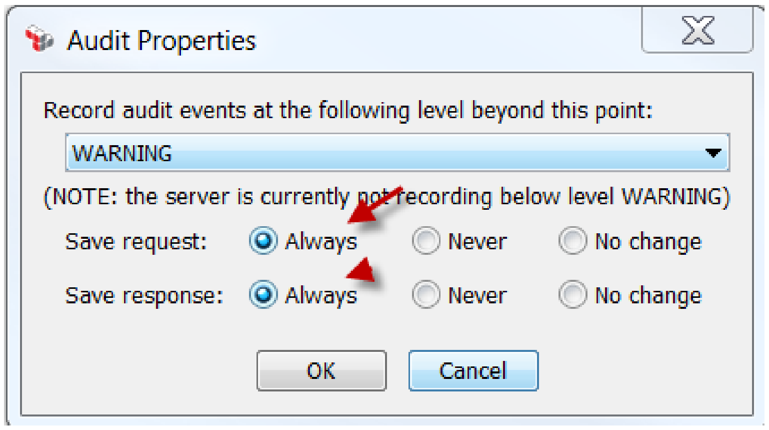


```
Context Variables during run time after route:

Response = ${response.mainpart}
Routing latency = ${httpRouting.latency}
http status = ${response.http.status}
```

Category: [✓ Audit] Log ☐ Custom logger name: com.l7tech.log.custom.

Level: INFO

Step 5 – Drag and drop the audit message in policy into the top of the service.



Record audit events at the following level beyond this point:

WARNING

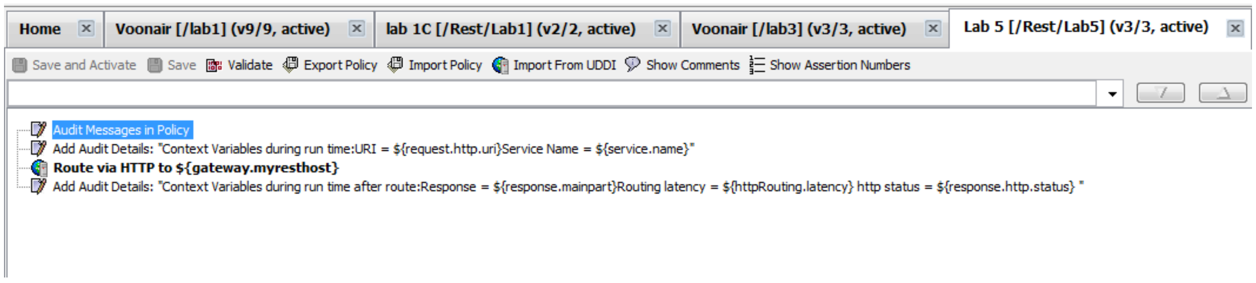
(NOTE: the server is currently not recording below level WARNING)

Save request: [✓ Always] Never No change

Save response: [✓ Always] Never No change

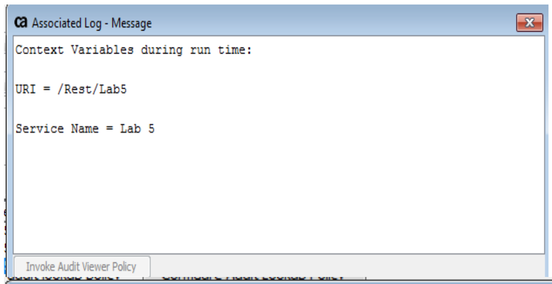
Step 6 – Save and Activate. Send a request through to the gateway. Check "view – Gateway Audits Events"

Step 7 – The service should look like this:

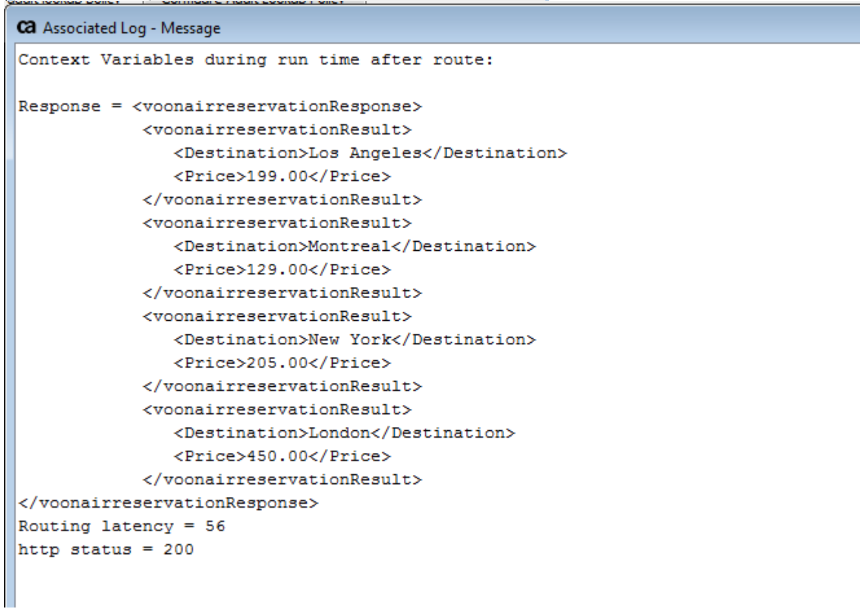

Home
Voonair [/lab1] (v9/9, active)
lab 1C [/Rest/Lab1] (v2/2, active)
Voonair [/lab3] (v3/3, active)
Lab 5 [/Rest/Lab5] (v3/3, active)
Save and Activate
Save
Validate
Export Policy
Import Policy
Import From UDDI
Show Comments
Show Assertion Numbers

Audit Messages in Policy
Add Audit Details: "Context Variables during run time:URI = ${request.http.uri}Service Name = ${service.name}"
Route via HTTP to ${gateway.myresthost}
Add Audit Details: "Context Variables during run time after route:Response = ${response.mainpart}Routing latency = ${httpRouting.latency} http status = ${response.http.status} "

Step 8 – **Save and Activate.** Test the service and take a look at the audits, by "view" and "Gateway Audits". Here's what it should look like. First Audit Details:



```
Context Variables during run time:

URI = /Rest/Lab5

Service Name = Lab 5
```



```
Context Variables during run time after route:

Response = <voonairreservationResponse>
            <voonairreservationResult>
               <Destination>Los Angeles</Destination>
               <Price>199.00</Price>
            </voonairreservationResult>
            <voonairreservationResult>
               <Destination>Montreal</Destination>
               <Price>129.00</Price>
            </voonairreservationResult>
            <voonairreservationResult>
               <Destination>New York</Destination>
               <Price>205.00</Price>
            </voonairreservationResult>
            <voonairreservationResult>
               <Destination>London</Destination>
               <Price>450.00</Price>
            </voonairreservationResult>
</voonairreservationResponse>
Routing latency = 56
http status = 200
```

LAB 5C: CREATE AND COMPARE CONTEXT VARIABLES

THE SITUATION

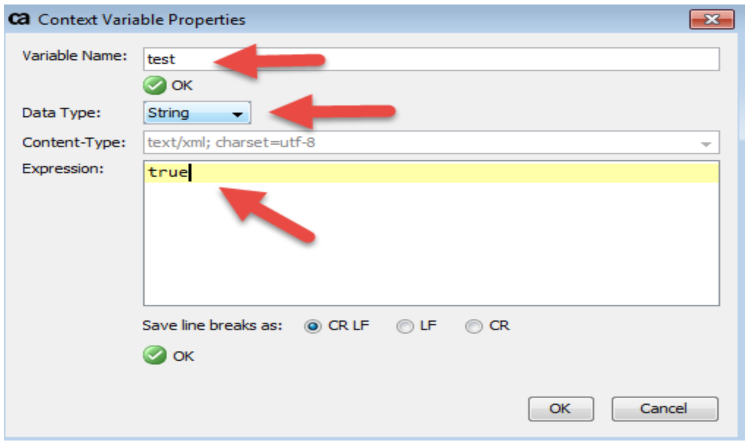
Developer Dave needs to do learn how to do comparisons for some business logic Business Man Bill has communicated. Help Dave out by creating a sample in policy logic.

THE SOLUTION

It's time to create some context variables and make use of them in policy logic.

Step 1 – Re-use a previous lab or copy to a unique lab to identify this exercise.

Step 2 – Drag and drop "Set Context Variable" into the service before the route. The assertion lives in the Policy Logic folder within the assertion palette.


Context Variable Properties

Variable Name: test

OK

Data Type: String

Content-Type: text/xml; charset=utf-8

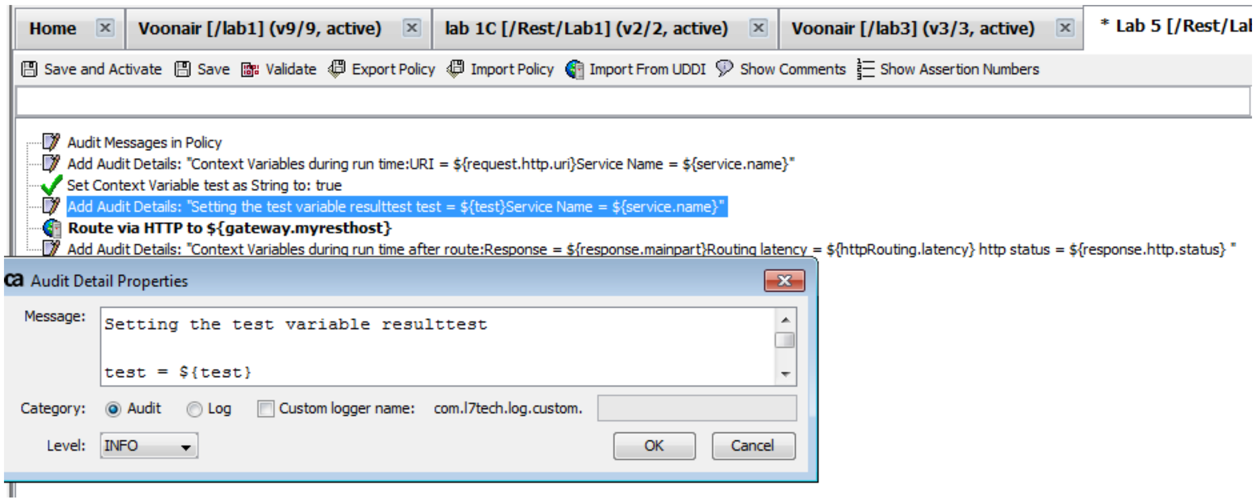
Expression: true

Save line breaks as: (● CR LF) LF CR

OK

Step 3 – Place the Variable name as "test". Keep the default setting of String for the data type and change the expression field to true. Select ok.

Step 4 – Drag and drop the "Add audit details" to below the set context variable assertion.



```
Audit Messages in Policy
Add Audit Details: "Context Variables during run time:URI = ${request.http.uri}Service Name = ${service.name}"
Set Context Variable test as String to: true
Add Audit Details: "Setting the test variable resulttest test = ${test}Service Name = ${service.name}"
Route via HTTP to ${gateway.myresthost}
Add Audit Details: "Context Variables during run time after route:Response = ${response.mainpart}Routing latency = ${httpRouting.latency} http status = ${response.http.status} "
```

Audit Detail Properties

Message:

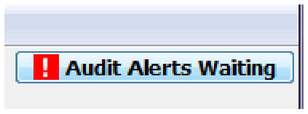
```
Setting the test variable resulttest

test = ${test}
```

Category: (● Audit) Log ☐ Custom logger name: com.l7tech.log.custom.

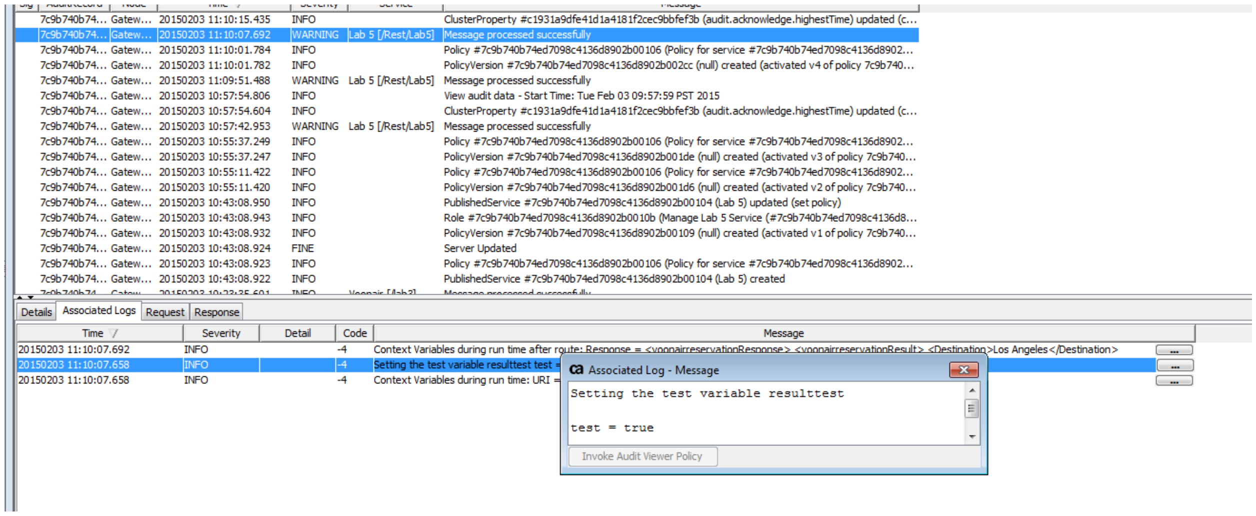
Level: INFO

Step 5 – Save and Activate. Send a request through to the gateway.

Step 6 – Check the Gateway Audit Events, by clicking on the shortcut "Audit Alerts Waiting" button in the upper right hand corner.



Click view audits:



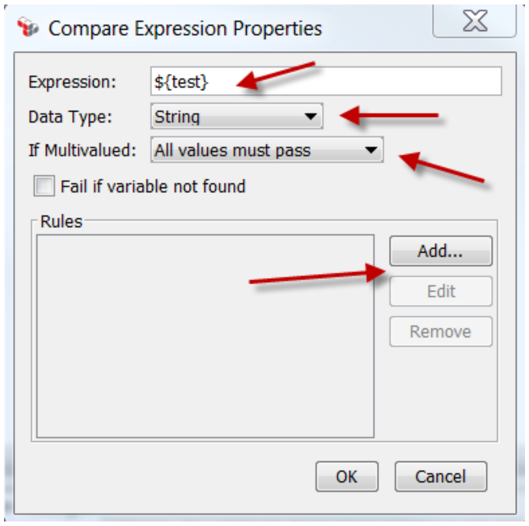
```
7c9b740b74... Gatew... 20150203 11:10:15.435 INFO                        ClusterProperty #c1931a9dfe41d1a4181f2cec9bbfef3b (audit.acknowledge.highestTime) updated (c...
7c9b740b74... Gatew... 20150203 11:10:07.692 WARNING Lab 5 [/Rest/Lab5]  Message processed successfully
7c9b740b74... Gatew... 20150203 11:10:01.784 INFO                        Policy #7c9b740b74ed7098c4136d8902b00106 (Policy for service #7c9b740b74ed7098c4136d8902...
7c9b740b74... Gatew... 20150203 11:10:01.782 INFO                        PolicyVersion #7c9b740b74ed7098c4136d8902b002cc (null) created (activated v4 of policy 7c9b740...
7c9b740b74... Gatew... 20150203 11:09:51.488 WARNING Lab 5 [/Rest/Lab5]  Message processed successfully
7c9b740b74... Gatew... 20150203 10:57:54.806 INFO                        View audit data - Start Time: Tue Feb 03 09:57:59 PST 2015
7c9b740b74... Gatew... 20150203 10:57:54.604 INFO                        ClusterProperty #c1931a9dfe41d1a4181f2cec9bbfef3b (audit.acknowledge.highestTime) updated (c...
7c9b740b74... Gatew... 20150203 10:57:42.953 WARNING Lab 5 [/Rest/Lab5]  Message processed successfully
7c9b740b74... Gatew... 20150203 10:55:37.249 INFO                        Policy #7c9b740b74ed7098c4136d8902b00106 (Policy for service #7c9b740b74ed7098c4136d8902...
7c9b740b74... Gatew... 20150203 10:55:37.247 INFO                        PolicyVersion #7c9b740b74ed7098c4136d8902b001de (null) created (activated v3 of policy 7c9b740...
7c9b740b74... Gatew... 20150203 10:55:11.422 INFO                        Policy #7c9b740b74ed7098c4136d8902b00106 (Policy for service #7c9b740b74ed7098c4136d8902...
7c9b740b74... Gatew... 20150203 10:55:11.420 INFO                        PolicyVersion #7c9b740b74ed7098c4136d8902b001d6 (null) created (activated v2 of policy 7c9b740...
7c9b740b74... Gatew... 20150203 10:43:08.950 INFO                        PublishedService #7c9b740b74ed7098c4136d8902b00104 (Lab 5) updated (set policy)
7c9b740b74... Gatew... 20150203 10:43:08.943 INFO                        Role #7c9b740b74ed7098c4136d8902b0010b (Manage Lab 5 Service (#7c9b740b74ed7098c4136d8...
7c9b740b74... Gatew... 20150203 10:43:08.932 INFO                        PolicyVersion #7c9b740b74ed7098c4136d8902b00109 (null) created (activated v1 of policy 7c9b740...
7c9b740b74... Gatew... 20150203 10:43:08.924 FINE                        Server Updated
7c9b740b74... Gatew... 20150203 10:43:08.923 INFO                        Policy #7c9b740b74ed7098c4136d8902b00106 (Policy for service #7c9b740b74ed7098c4136d8902...
7c9b740b74... Gatew... 20150203 10:43:08.922 INFO                        PublishedService #7c9b740b74ed7098c4136d8902b00104 (Lab 5) created
```

```
Time                   Severity  Detail  Code  Message
20150203 11:10:07.692  INFO              -4    Context Variables during run time after route: Response = <yoonairreservationResponse> <yoonairreservationResult> <Destination>Los Angeles</Destination>
20150203 11:10:07.658  INFO              -4    Setting the test variable resulttest test =
20150203 11:10:07.658  INFO              -4    Context Variables during run time: URI =
```

Associated Log - Message

```
Setting the test variable resulttest

test = true
```

Step 7 – Re-use the same service. Drag and drop the Compare Expression to above the Route.

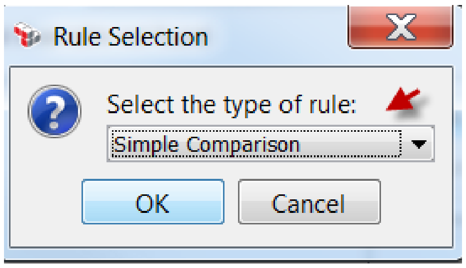


Compare Expression Properties

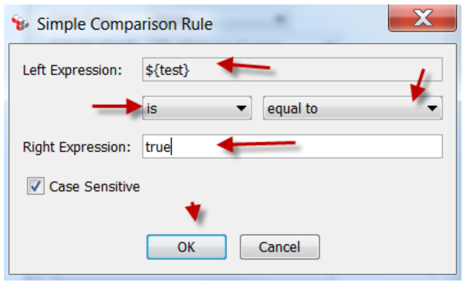
Expression:	${test}
Data Type:	String
If Multivalued:	All values must pass

☐ Fail if variable not found

Step 8 – Set up the Expression field with the context variable that you created earlier in the service. Change the Data Type to be the same as your variable that was set (string). If Multivalued field, should read "All values must pass". Click Add.

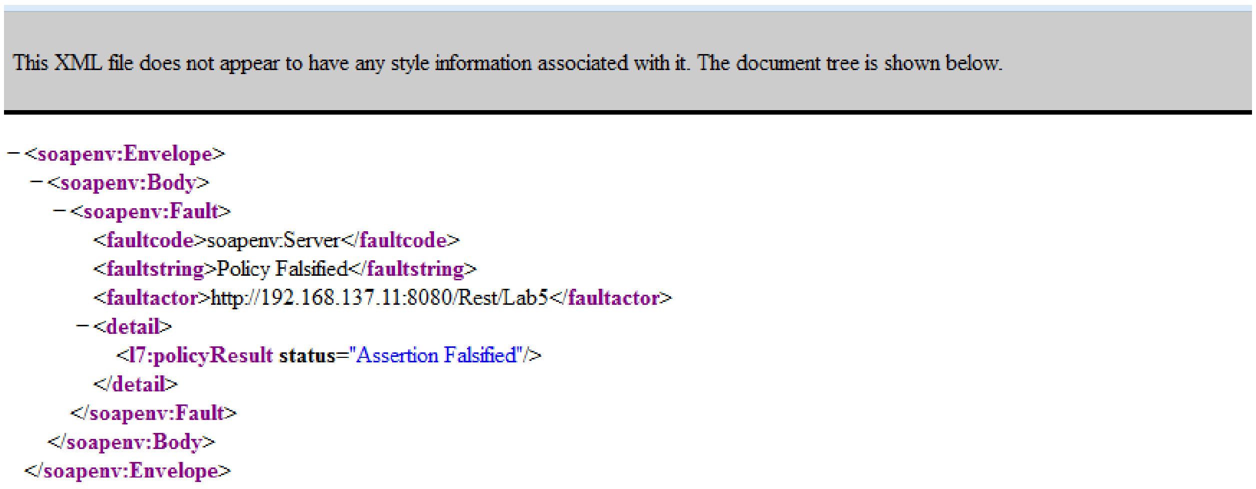


Step 9 – The above wizard shows up, click on Simple Comparison and click Ok.



Step 10 – Save and Activate. Send the request through. See that it passes; now, change the "test" context variable to **false**. See what happens when you send your request through now.

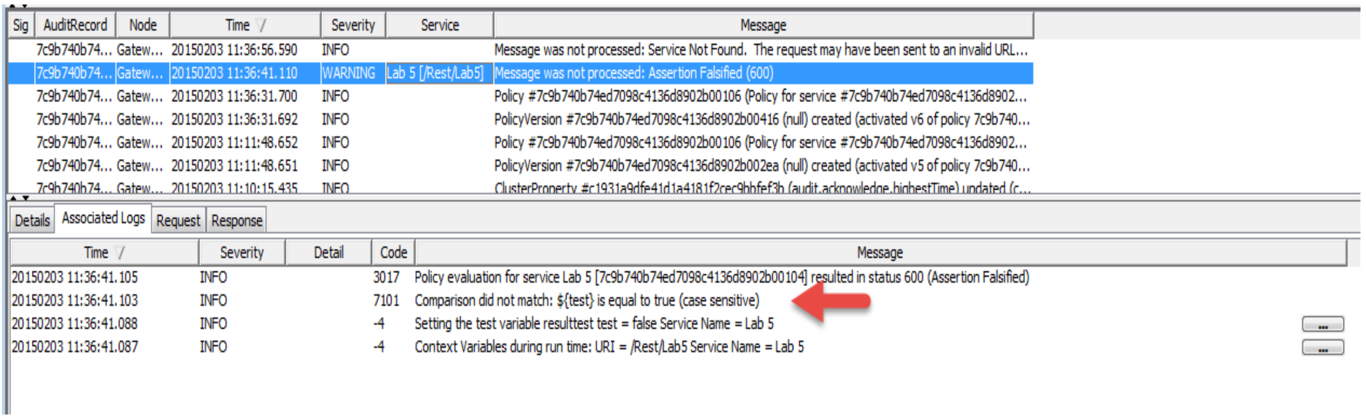
You should see "Assertion Falsified in your response.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <soapenv:Envelope>
  - <soapenv:Body>
    - <soapenv:Fault>
        <faultcode>soapenv:Server</faultcode>
        <faultstring>Policy Falsified</faultstring>
        <faultactor>http://192.168.137.11:8080/Rest/Lab5</faultactor>
      - <detail>
          <l7:policyResult status="Assertion Falsified"/>
        </detail>
      </soapenv:Fault>
    </soapenv:Body>
  </soapenv:Envelope>
```

In the Gateway Audit Events, this is what will appear:



Sig	AuditRecord	Node	Time	Severity	Service	Message
	7c9b740b74...	Gatew...	20150203 11:36:56.590	INFO		Message was not processed: Service Not Found. The request may have been sent to an invalid URL...
	7c9b740b74...	Gatew...	20150203 11:36:41.110	WARNING	Lab 5 [/Rest/Lab5]	Message was not processed: Assertion Falsified (600)
	7c9b740b74...	Gatew...	20150203 11:36:31.700	INFO		Policy #7c9b740b74ed7098c4136d8902b00106 (Policy for service #7c9b740b74ed7098c4136d8902...
	7c9b740b74...	Gatew...	20150203 11:36:31.692	INFO		PolicyVersion #7c9b740b74ed7098c4136d8902b00416 (null) created (activated v6 of policy 7c9b740...
	7c9b740b74...	Gatew...	20150203 11:11:48.652	INFO		Policy #7c9b740b74ed7098c4136d8902b00106 (Policy for service #7c9b740b74ed7098c4136d8902...
	7c9b740b74...	Gatew...	20150203 11:11:48.651	INFO		PolicyVersion #7c9b740b74ed7098c4136d8902b002ea (null) created (activated v5 of policy 7c9b740...
	7c9b740b74...	Gatew...	20150203 11:10:15.435	INFO		ClusterProperty #c1931a9dfe41d1a4181f2cec9bbfef3b (audit.acknowledge.highestTime) updated (c...

Time	Severity	Detail	Code	Message
20150203 11:36:41.105	INFO		3017	Policy evaluation for service Lab 5 [7c9b740b74ed7098c4136d8902b00104] resulted in status 600 (Assertion Falsified)
20150203 11:36:41.103	INFO		7101	Comparison did not match: ${test} is equal to true (case sensitive)
20150203 11:36:41.088	INFO		-4	Setting the test variable resulttest test = false Service Name = Lab 5
20150203 11:36:41.087	INFO		-4	Context Variables during run time: URI = /Rest/Lab5 Service Name = Lab 5

LAB 5D: POLICY BRANCHING

THE SITUATION

Dave needs to do some policy branching to enable the 'if … than' statements. He needs to determine within a policy how to only allow a certain user to access a backend service. All other users get denied or receive an error.

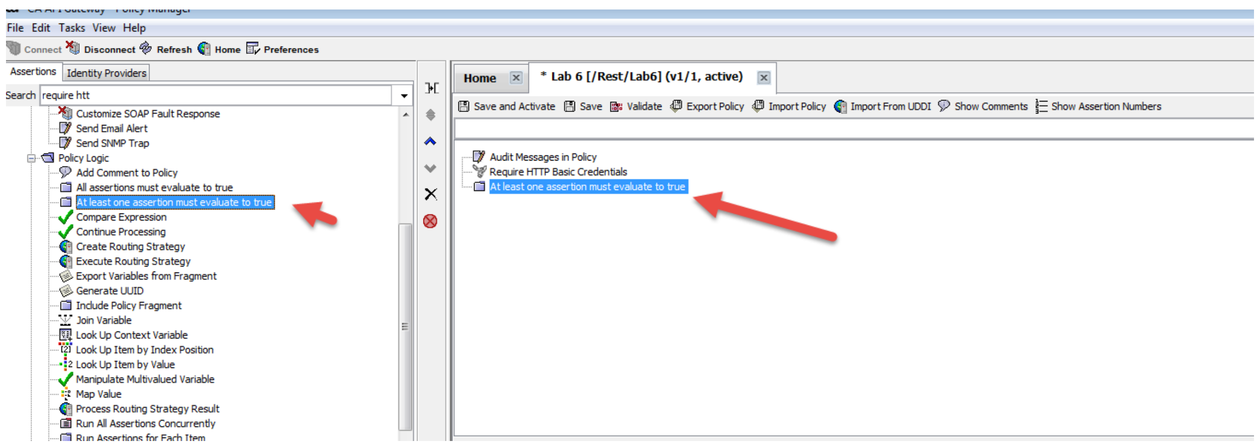
THE SOLUTION

Dave must enable policy branching logic and needs your help!

Step 1 – Create a new Web API service or copy an existing one.

Step 2 – Find the assertion "require HTTP Basic Credentials" within Access Control folder. Place this into the policy Development window. Drag and drop the Audit message in Policy assertion to the top of the policy. This ensures that the "required" credentials are present and in the right formate.

Step 3 – Search in the Assertion palette for "at least one assertion". Drag the "at least one assertion must evaluate to true" into the policy. This is the start of branching exercise.



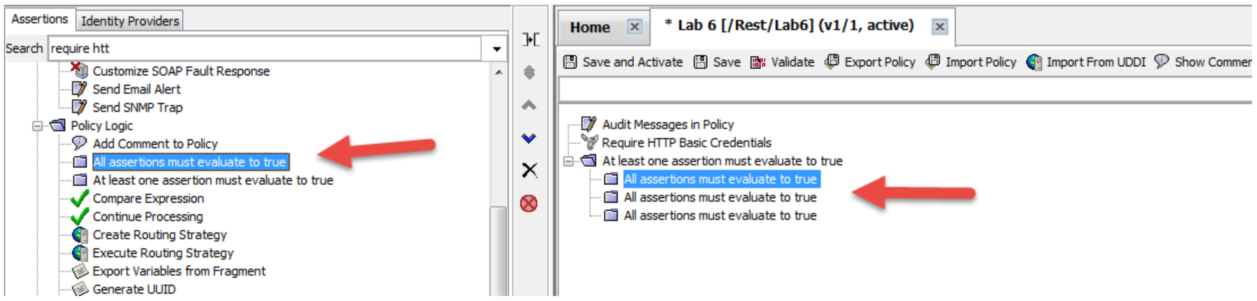
Search: require htt

- Customize SOAP Fault Response
- Send Email Alert
- Send SNMP Trap
- Policy Logic
  - Add Comment to Policy
  - All assertions must evaluate to true
  - At least one assertion must evaluate to true
  - Compare Expression
  - Continue Processing
  - Create Routing Strategy
  - Execute Routing Strategy
  - Export Variables from Fragment
  - Generate UUID
  - Include Policy Fragment
  - Join Variable
  - Look Up Context Variable
  - Look Up Item by Index Position
  - Look Up Item by Value
  - Manipulate Multivalued Variable
  - Map Value
  - Process Routing Strategy Result
  - Run All Assertions Concurrently
  - Run Assertions for Each Item

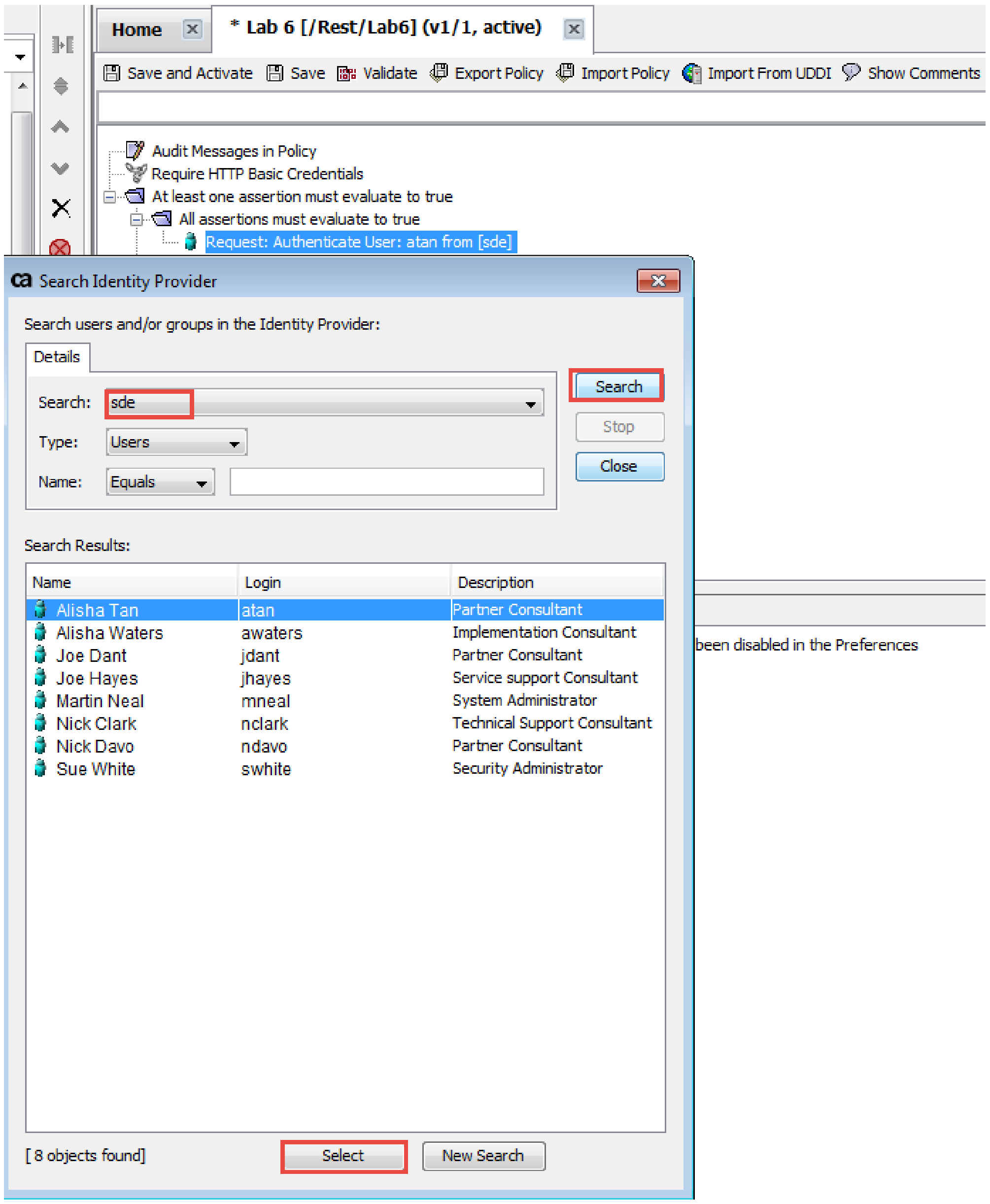
* Lab 6 [/Rest/Lab6] (v1/1, active)

- Audit Messages in Policy
- Require HTTP Basic Credentials
- At least one assertion must evaluate to true

Step 4 – Within the At least one assertion branch place 3 “all assertions must evaluate to true”. Easy lookup, it’s just above the at least one assertion. Drag and drop the assertion into policy on top of the at least one assertion in the policy, so it should look like this:


Assertions
Identity Providers
Search
require htt
Customize SOAP Fault Response
Send Email Alert
Send SNMP Trap
Policy Logic
Add Comment to Policy
All assertions must evaluate to true
At least one assertion must evaluate to true
Compare Expression
Continue Processing
Create Routing Strategy
Execute Routing Strategy
Export Variables from Fragment
Generate UUID
Home
* Lab 6 [/Rest/Lab6] (v1/1, active)
Save and Activate
Save
Validate
Export Policy
Import Policy
Import From UDDI
Show Commer
Audit Messages in Policy
Require HTTP Basic Credentials
At least one assertion must evaluate to true
All assertions must evaluate to true
All assertions must evaluate to true
All assertions must evaluate to true

Step 5: Within the top two “all assertions must evaluate to true”, drag and drop “Authenticate User or Group” into policy:



Audit Messages in Policy
Require HTTP Basic Credentials
At least one assertion must evaluate to true
All assertions must evaluate to true
Request: Authenticate User: atan from [sde]

Search Identity Provider

Search users and/or groups in the Identity Provider:

Details

Search: sde
Type: Users
Name: Equals

Search Results:

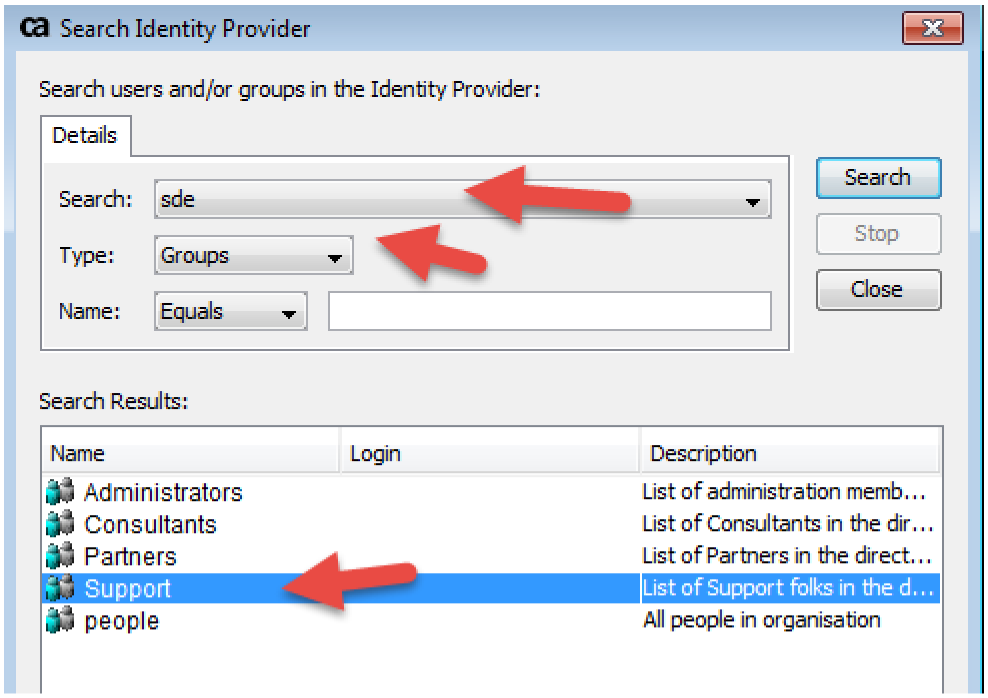
Name	Login	Description
Alisha Tan	atan	Partner Consultant
Alisha Waters	awaters	Implementation Consultant
Joe Dant	jdant	Partner Consultant
Joe Hayes	jhayes	Service support Consultant
Martin Neal	mneal	System Administrator
Nick Clark	nclark	Technical Support Consultant
Nick Davo	ndavo	Partner Consultant
Sue White	swhite	Security Administrator

[8 objects found]

been disabled in the Preferences

- Select the following user for the first authentication = Alisha Tan
- 2nd user for the authentication = Joe Dant

Step 6 – Drag and drop the “Authenticate User or Group” assertion into the 3rd All Assertion. Selecting the Type as Groups:


Search Identity Provider

Search users and/or groups in the Identity Provider:

Details

Search: sde

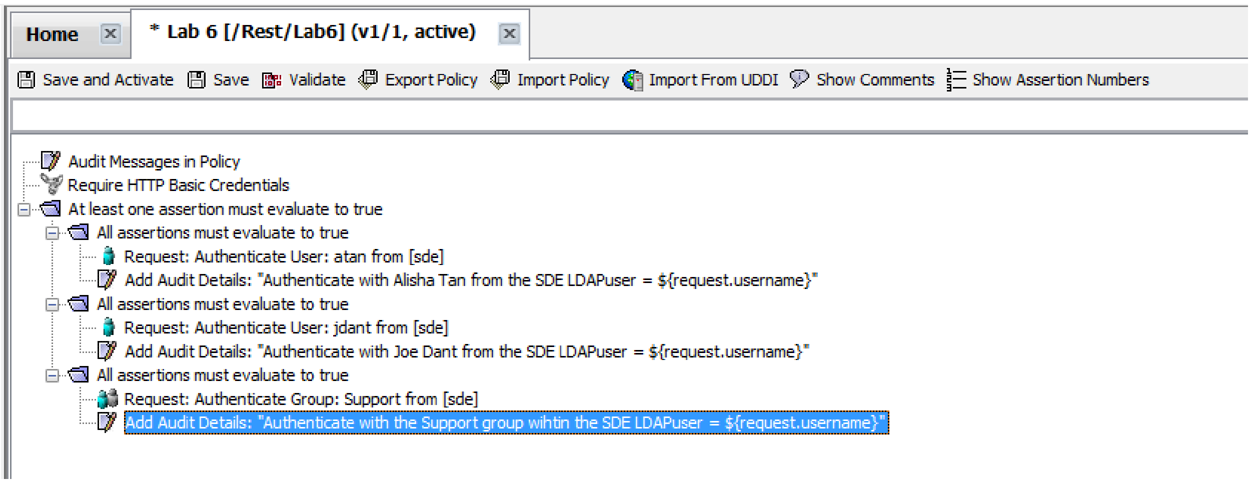
Type: Groups

Name: Equals

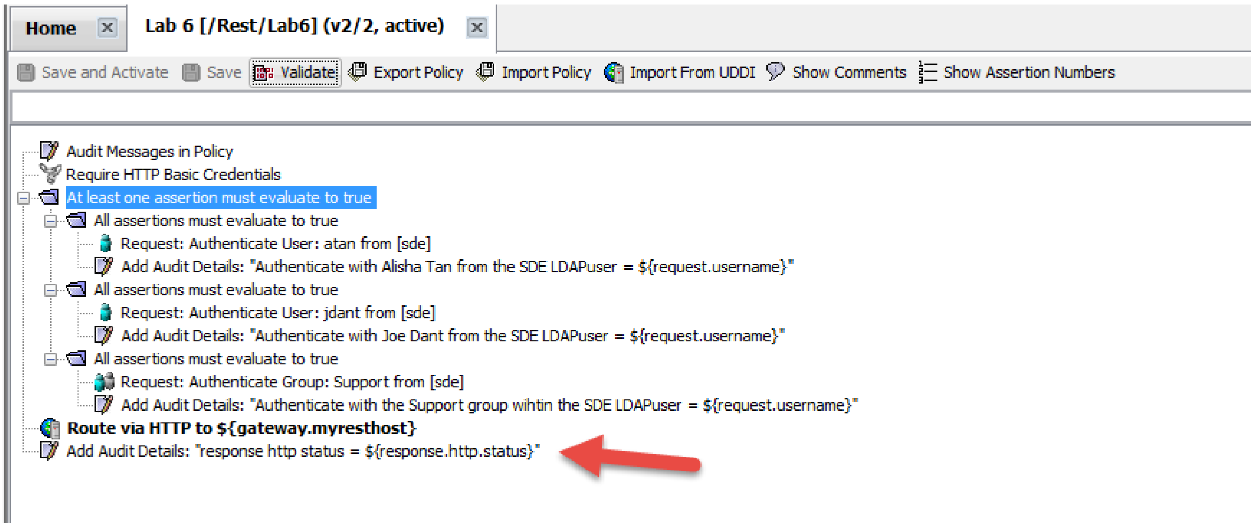
Search Results:

Name	Login	Description
Administrators		List of administration memb…
Consultants		List of Consultants in the dir…
Partners		List of Partners in the direct…
Support		List of Support folks in the d…
people		All people in organisation

- Select the Support group. This is what it should look like at this stage:


```
Audit Messages in Policy
Require HTTP Basic Credentials
At least one assertion must evaluate to true
    All assertions must evaluate to true
        Request: Authenticate User: atan from [sde]
        Add Audit Details: "Authenticate with Alisha Tan from the SDE LDAPuser = ${request.username}"
    All assertions must evaluate to true
        Request: Authenticate User: jdant from [sde]
        Add Audit Details: "Authenticate with Joe Dant from the SDE LDAPuser = ${request.username}"
    All assertions must evaluate to true
        Request: Authenticate Group: Support from [sde]
        Add Audit Details: "Authenticate with the Support group wihtin the SDE LDAPuser = ${request.username}"
```

Step 7 – The last 2 assertions of this policy added are related to HTTP/S Routing. Make sure the route is "**not**" within the nested logic and is the last assertion of the policy at this point. The last assertion is placing another Add Audit detail assertion to the very bottom, looking at the resulting status of the HTTP route to the back end.

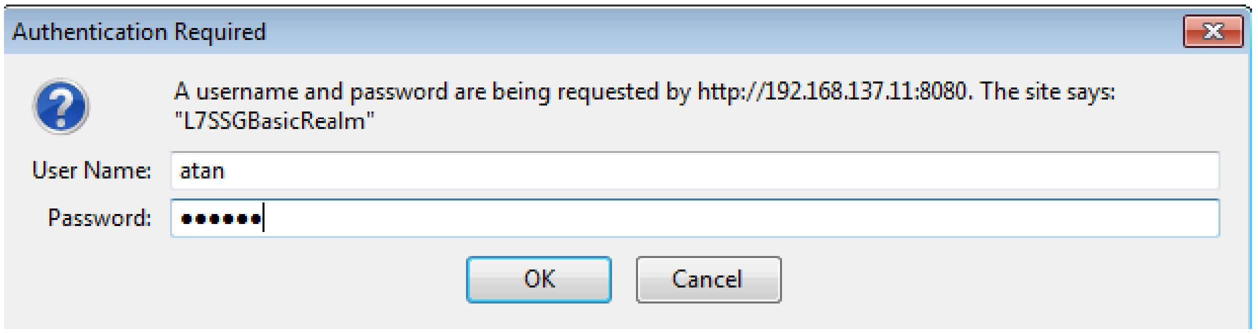


```
Audit Messages in Policy
Require HTTP Basic Credentials
At least one assertion must evaluate to true
    All assertions must evaluate to true
        Request: Authenticate User: atan from [sde]
        Add Audit Details: "Authenticate with Alisha Tan from the SDE LDAPuser = ${request.username}"
    All assertions must evaluate to true
        Request: Authenticate User: jdant from [sde]
        Add Audit Details: "Authenticate with Joe Dant from the SDE LDAPuser = ${request.username}"
    All assertions must evaluate to true
        Request: Authenticate Group: Support from [sde]
        Add Audit Details: "Authenticate with the Support group wihtin the SDE LDAPuser = ${request.username}"
Route via HTTP to ${gateway.myresthost}
Add Audit Details: "response http status = ${response.http.status}"
```

Step 8 – Save and Activate. Send the request using the browser window that was open from before. When you send the request through you should have a pop up window allowing you to input user credentials.

All passwords associated to the users in the SDE LDAP are ***7layer***. The user in order to pass this service, use one of the 2 users (atan,jdant) and to test the group account, use jhayes.

Note: - You will need to clean up the cache of your browser each time you run through this.



A username and password are being requested by http://192.168.137.11:8080. The site says: "L7SSGBasicRealm"

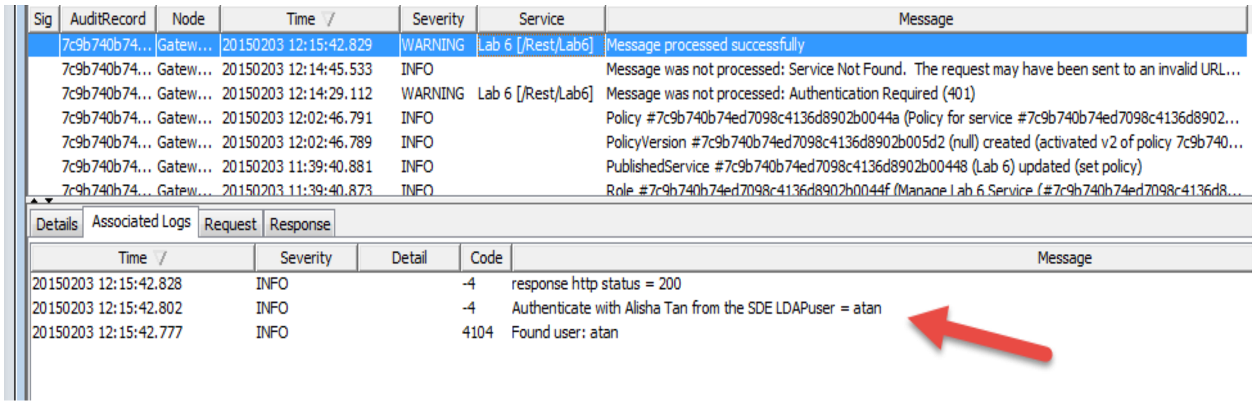
User Name: atan

Password: ••••••

Type in the user Name = atan

Password = 7layer

Step 9 – Check the Gateway Audits to see how the service ran through.



Sig	AuditRecord	Node	Time	Severity	Service	Message
	7c9b740b74...	Gatew...	20150203 12:15:42.829	WARNING	Lab 6 [/Rest/Lab6]	Message processed successfully
	7c9b740b74...	Gatew...	20150203 12:14:45.533	INFO		Message was not processed: Service Not Found. The request may have been sent to an invalid URL...
	7c9b740b74...	Gatew...	20150203 12:14:29.112	WARNING	Lab 6 [/Rest/Lab6]	Message was not processed: Authentication Required (401)
	7c9b740b74...	Gatew...	20150203 12:02:46.791	INFO		Policy #7c9b740b74ed7098c4136d8902b0044a (Policy for service #7c9b740b74ed7098c4136d8902...
	7c9b740b74...	Gatew...	20150203 12:02:46.789	INFO		PolicyVersion #7c9b740b74ed7098c4136d8902b005d2 (null) created (activated v2 of policy 7c9b740...
	7c9b740b74...	Gatew...	20150203 11:39:40.881	INFO		PublishedService #7c9b740b74ed7098c4136d8902b00448 (Lab 6) updated (set policy)
	7c9b740b74...	Gatew...	20150203 11:39:40.873	INFO		Role #7c9b740b74ed7098c4136d8902b0044f (Manage Lab 6 Service (#7c9b740b74ed7098c4136d8...

Time	Severity	Detail	Code	Message
20150203 12:15:42.828	INFO		-4	response http status = 200
20150203 12:15:42.802	INFO		-4	Authenticate with Alisha Tan from the SDE LDAPuser = atan
20150203 12:15:42.777	INFO		4104	Found user: atan

MODULE 6: FRAGMENTS & GLOBAL POLICIES

LAB 6A: CREATE A POLICY FRAGMENT

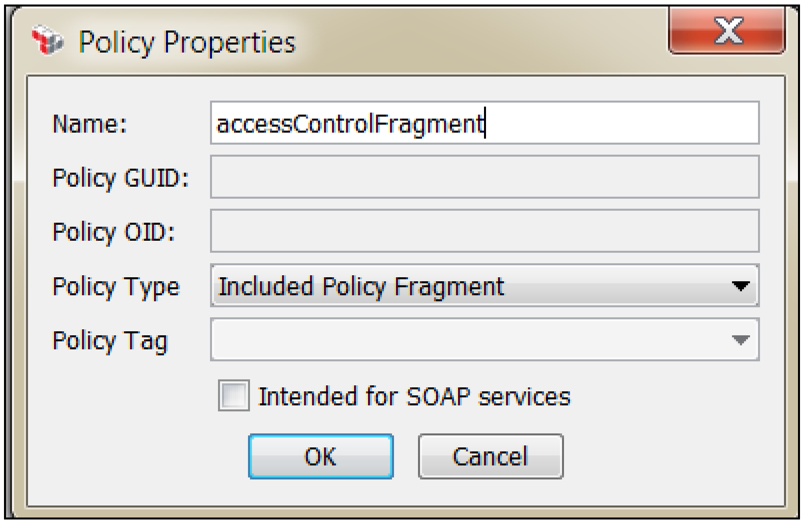
THE SITUATION

Dave needs to start looking at modularizing his code because he has found out from Architect Andy that they will be deploying many more services in the near future.

THE SOLUTION

Implement Policy Fragments in the policy so that other services can utilize the same logic without repeating development.

Step 1: Create an Included Policy Fragment



Name: accessControlFragment
Policy GUID:
Policy OID:
Policy Type: Included Policy Fragment
Policy Tag:
☐ Intended for SOAP services

Step 2: Copy Authentication Logic to **accessControlFragment**

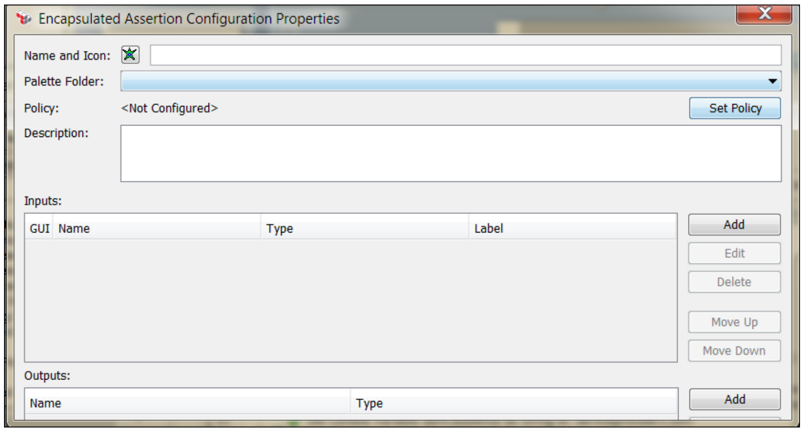
Step 3: Delete or disable previous logic in Master Policy and use “Include Policy Fragment” assertion to reference the new logic from the policy fragment created.

Step 4: Test with SOAPUI to ensure new fragment executes the same logic without error. Note that you cannot edit this within the master policy because it is a common piece of code used in many services.

LAB 6B: CONVERT POLICY FRAGMENT TO AN ENCAPSULATED ASSERTION

Step 1 - Select Tasks -> Manage Encapsulated Assertions option

Step 2 - Select Create and fill in fields in pop-up window


Encapsulated Assertion Configuration Properties

Name and Icon:

Palette Folder:

Policy: <Not Configured>

Description:

Inputs:

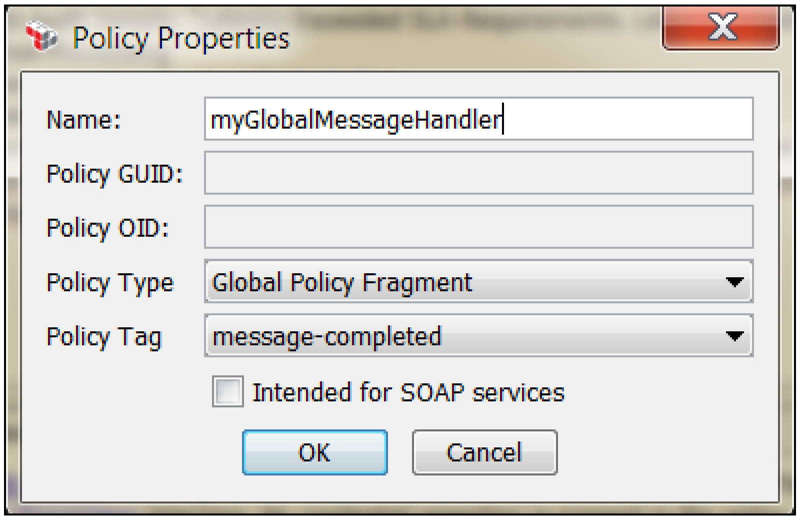
GUI	Name	Type	Label

Outputs:

Name	Type

LAB 6C: CREATE A GLOBAL POLICY

Step 1 – Create a Global Policy from the TASKS->Create Policy


Policy Properties

Name: myGlobalMessageHandler

Policy GUID:

Policy OID:

Policy Type: Global Policy Fragment

Policy Tag: message-completed

☐ Intended for SOAP services

Step 2 – Add and Audit Detail Assertion with a message in it

Step 3 – Send requests from SOAPUI to 2 different services and note that the message is logged for both.

MODULE 7: ACCESS CONTROL

LAB 7A: ACCESS CONTROL VIA INTERNAL IDENTITY

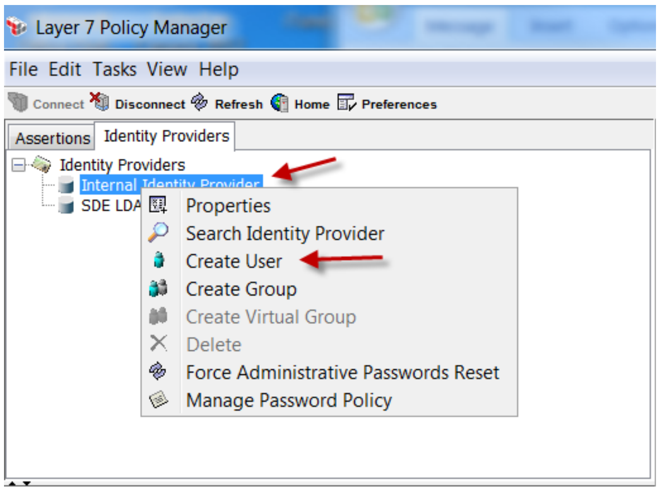
THE SITUATION

Operations Oscar controls who can have access to the Gateway. He has some new-hires that he needs to grant access to.

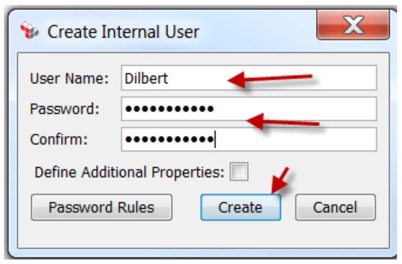
THE SOLUTION

Learn how to create new users within the Internal Identity Provider.

Step 1 – When logged into the gateway, navigate to the "Identity Providers" tab, beside the Assertions tab. Under the Identity Providers root level of the window, there will be 2 providers, one from your SDE LDAP that we configured earlier and the Internal Identity Provider. Right click on the "Internal Identity Provider", click on "Create User":



Step 2 – Type Dave in the User Name field

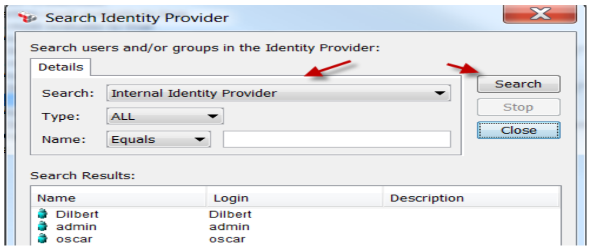

Create Internal User

User Name: Dilbert
Password: •••••••••••
Confirm: •••••••••••
Define Additional Properties: ☐

In the Password field use the following password (based on our stig requirements). L7Secure$0@ Click Create.

Create another user for Operations Oscar. (eg. Oscar).

Step 3 – To view the users within your Internal Identity Provider, you can right click on the Internal Identity Providers



Search Identity Provider

Search users and/or groups in the Identity Provider:

Details

Search: Internal Identity Provider

Type: ALL

Name: Equals

Search

Stop

Close

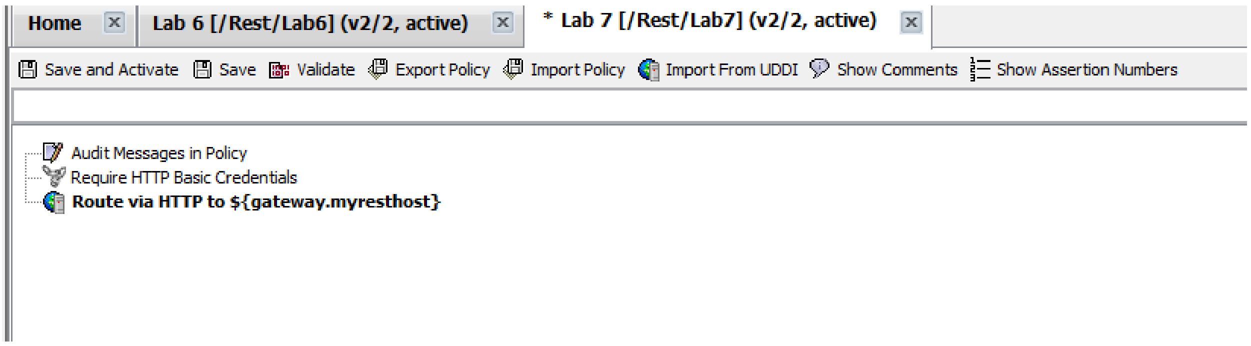
Search Results:

Name	Login	Description
Dilbert	Dilbert	
admin	admin	
oscar	oscar	

Step 4 – Create a new Soap Service with the same wsdl as all of the others. Layer the Service with the following assertions:

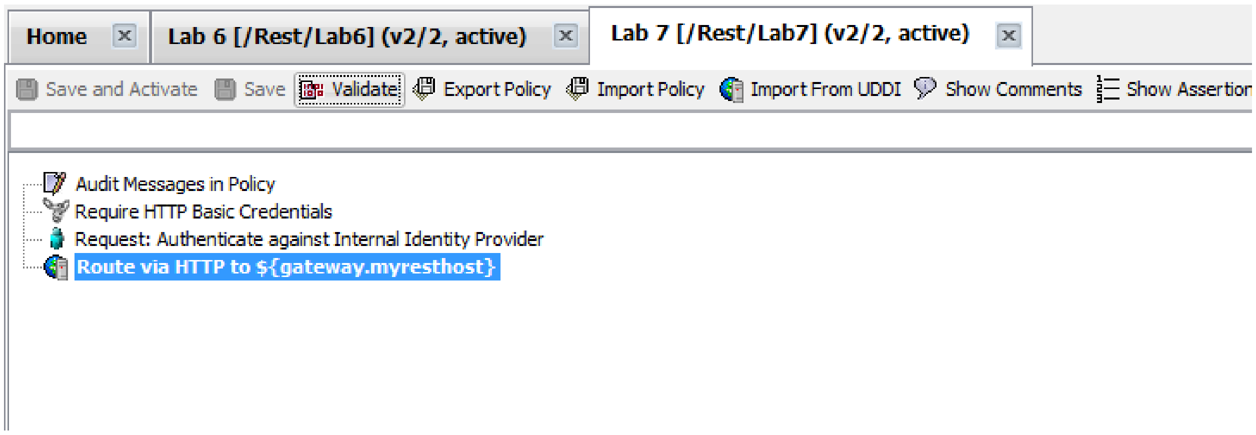
Audits = Audit Messages in Policy

Requirement of credentials = Require HTTP Basic Credentials



Audit Messages in Policy
Require HTTP Basic Credentials
Route via HTTP to ${gateway.myresthost}

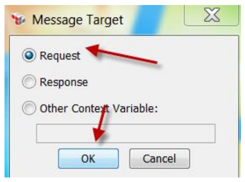
Step 5 – For this lab you will be authenticating against your internal identity provider. Drag and Drop the “Authenticate Against Identity Provider” assertion to above the “Route via HTTP” assertion within policy.



Audit Messages in Policy
Require HTTP Basic Credentials
Request: Authenticate against Internal Identity Provider
Route via HTTP to ${gateway.myresthost}

*** Important – The Authentication assertion must read “Request” at the front of the policy assertion. If you moved the assertion into the service below the route, the first part of the assertion will read “Response:

Authenticate against…”.   You will also need to right click on the assertion and *“Select Target Message”*. Change to Request for the target message. ***



Step 6 – Save and Activate

Step 7 – Go back to your Browser and run the request to /Rest/Lab7. Change the username from one of the ldap sde usernames to oscar. Send this request through.

LAB 7B: RESTRICT ACCESS TO SSL OR TLS TRANSPORT

THE SITUATION

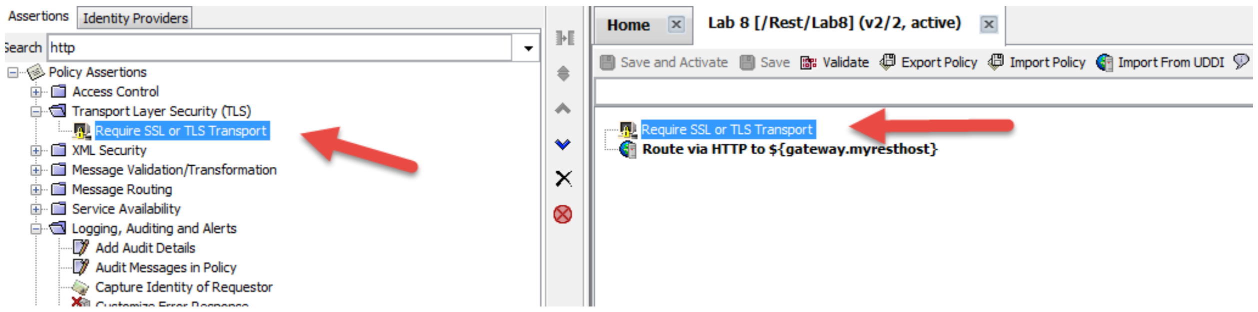
Security has decided that the services require SSL to prevent a hijack of personal information across the wire.

THE SOLUTION

Implement the assertion “Require SSL or TLS Transport”.

Step 1 – Open up an existing service in the policy manager

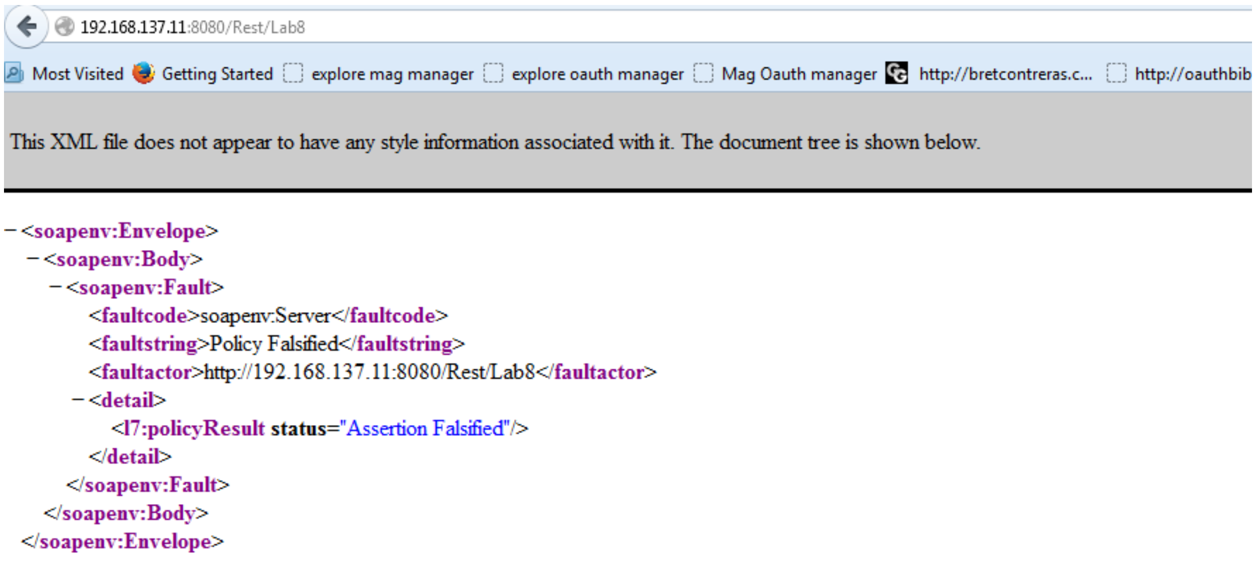
Step 2 – Drag and Drop the “Require SSL or TLS Transport” assertion to the top of your policy, above the “Route via HTTP”:



Require SSL or TLS Transport
Route via HTTP to ${gateway.myresthost}

Step 3 – Save and Activate.

Step 4 – Send a request to the following gateway service URL on port 8080 (non-SSL).

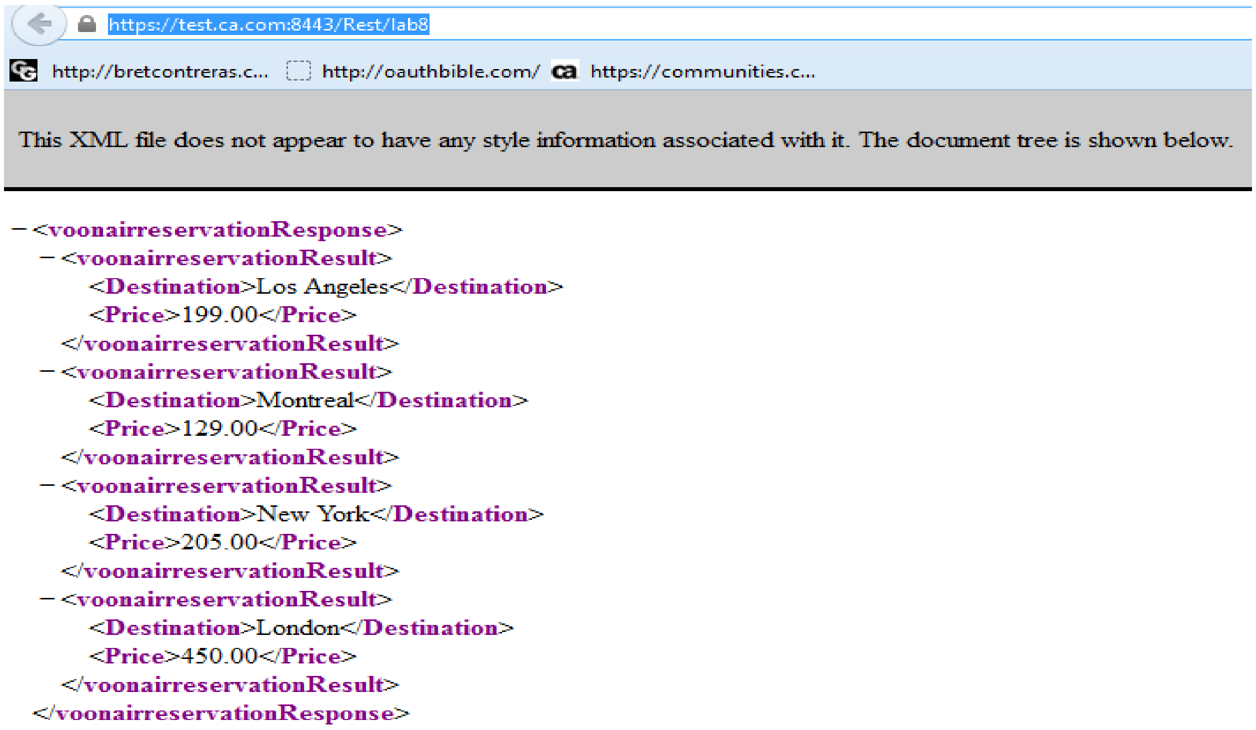


This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
− <soapenv:Envelope>
  − <soapenv:Body>
    − <soapenv:Fault>
        <faultcode>soapenv:Server</faultcode>
        <faultstring>Policy Falsified</faultstring>
        <faultactor>http://192.168.137.11:8080/Rest/Lab8</faultactor>
      − <detail>
          <l7:policyResult status="Assertion Falsified"/>
        </detail>
      </soapenv:Fault>
    </soapenv:Body>
  </soapenv:Envelope>
```

You should see the policy fail, because we're not meeting the need of SSL transmission.

Step 5 –When you send the service to the same URL using the necessary SSL transmission of https and port 8443 you should see a successful result.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
− <voonairreservationResponse>
  − <voonairreservationResult>
      <Destination>Los Angeles</Destination>
      <Price>199.00</Price>
    </voonairreservationResult>
  − <voonairreservationResult>
      <Destination>Montreal</Destination>
      <Price>129.00</Price>
    </voonairreservationResult>
  − <voonairreservationResult>
      <Destination>New York</Destination>
      <Price>205.00</Price>
    </voonairreservationResult>
  − <voonairreservationResult>
      <Destination>London</Destination>
      <Price>450.00</Price>
    </voonairreservationResult>
  </voonairreservationResponse>
```

MODULE 8: MESSAGE SECURITY

LAB 8A: EXPLORE XPATH VALIDATION

THE SITUATION

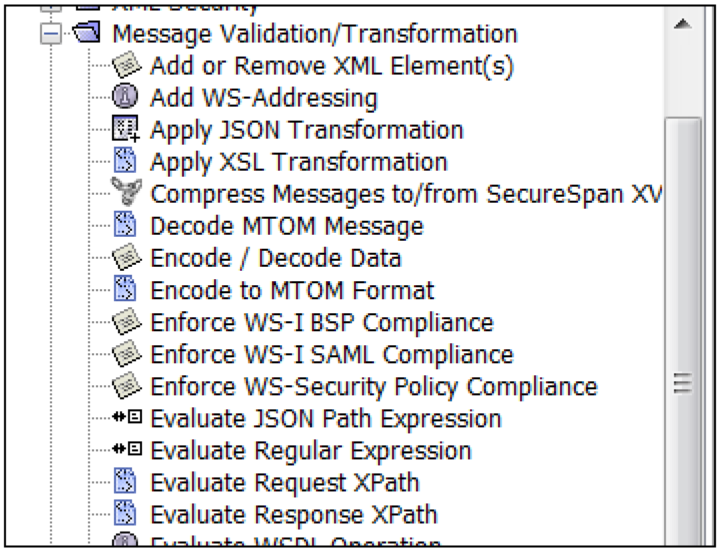
Dave needs to extract a value from a SOAP message to make sure that it is below a specific value.

THE SOLUTION

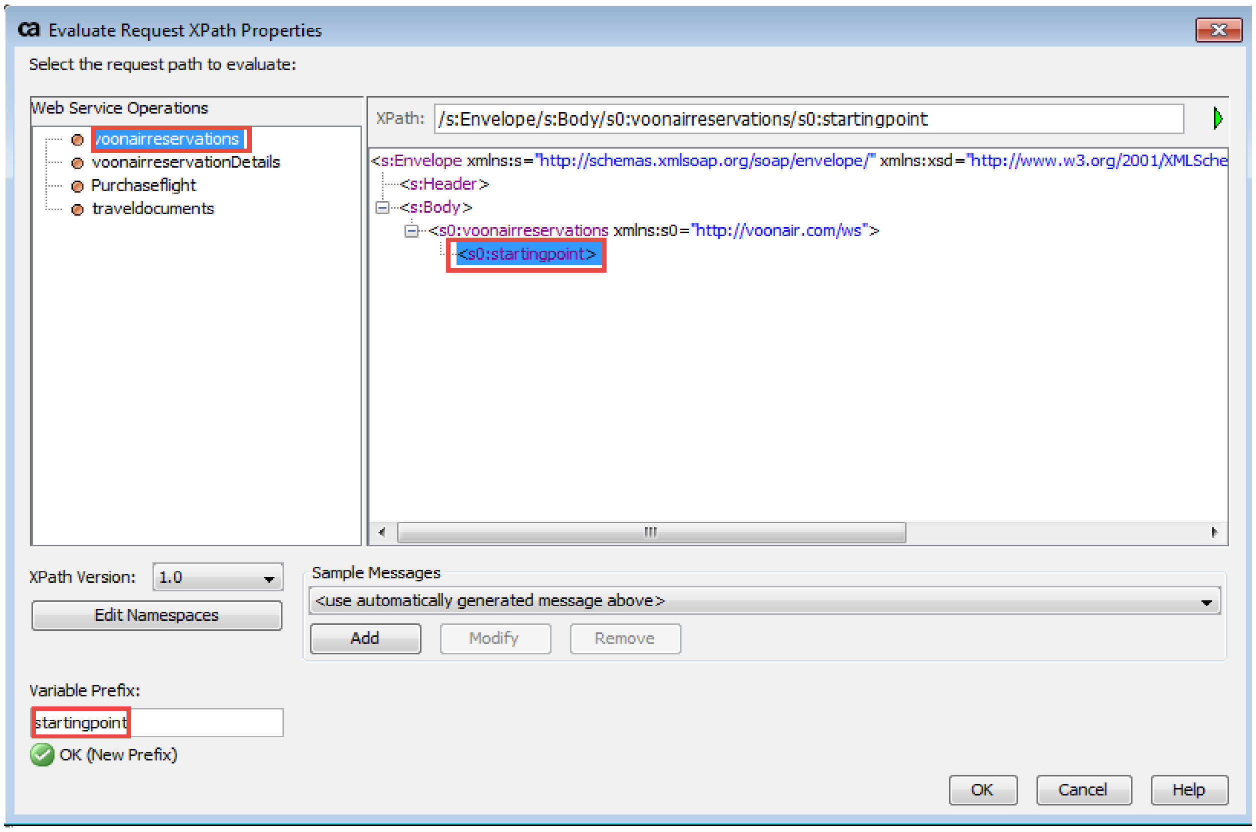
Dave needs to learn how to extract data out of services with the Evaluate XPath Request/Response

Step 1 – Build a new Soap service call it **lab9**

Step 2 - Drag and Drop the "Evaluate Request XPath" into your policy before the Route HTTP/S Assertion


Message Validation/Transformation
Add or Remove XML Element(s)
Add WS-Addressing
Apply JSON Transformation
Apply XSL Transformation
Compress Messages to/from SecureSpan XV
Decode MTOM Message
Encode / Decode Data
Encode to MTOM Format
Enforce WS-I BSP Compliance
Enforce WS-I SAML Compliance
Enforce WS-Security Policy Compliance
Evaluate JSON Path Expression
Evaluate Regular Expression
Evaluate Request XPath
Evaluate Response XPath

Step 3 – Click on the "***voonairreservations***" Operation and highlight the ***startingpoint*** element. Place startingpoint in the variable prefix box.



XPath: /s:Envelope/s:Body/s0:voonairreservations/s0:startingpoint

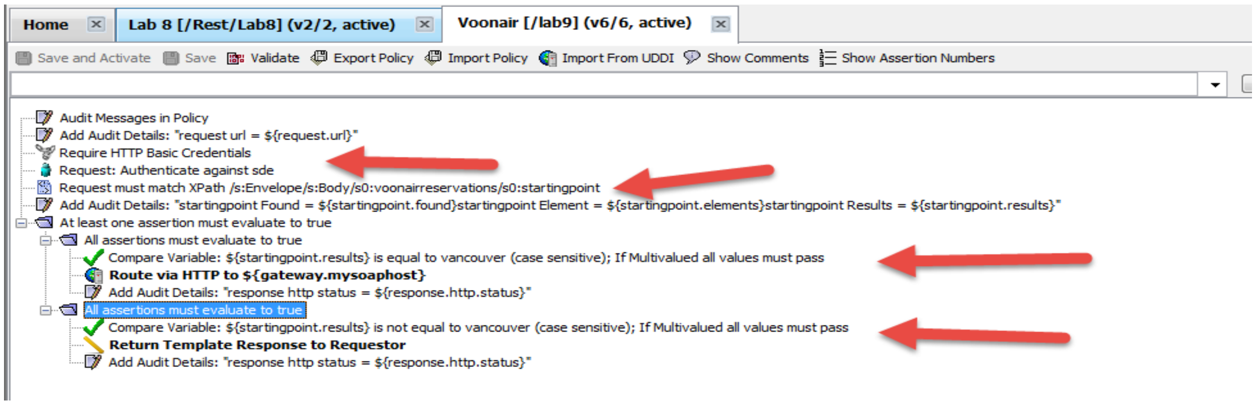
```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSche
  <s:Header>
  <s:Body>
    <s0:voonairreservations xmlns:s0="http://voonair.com/ws">
      <s0:startingpoint>
```

XPath Version: 1.0

Variable Prefix: startingpoint

Step 4 – Use “F1” for help and search for the “XPath Request” (e.g. **startingpoint.results)** you’ll find a list of context variables that get populated with this assertion execution.

Step 5 – Put Logic in that checks the value of the “startingpoint” attribute and use the “Compare Expression” to make sure it is not equal to Vancouver. Incorporate logic to handle an error condition.

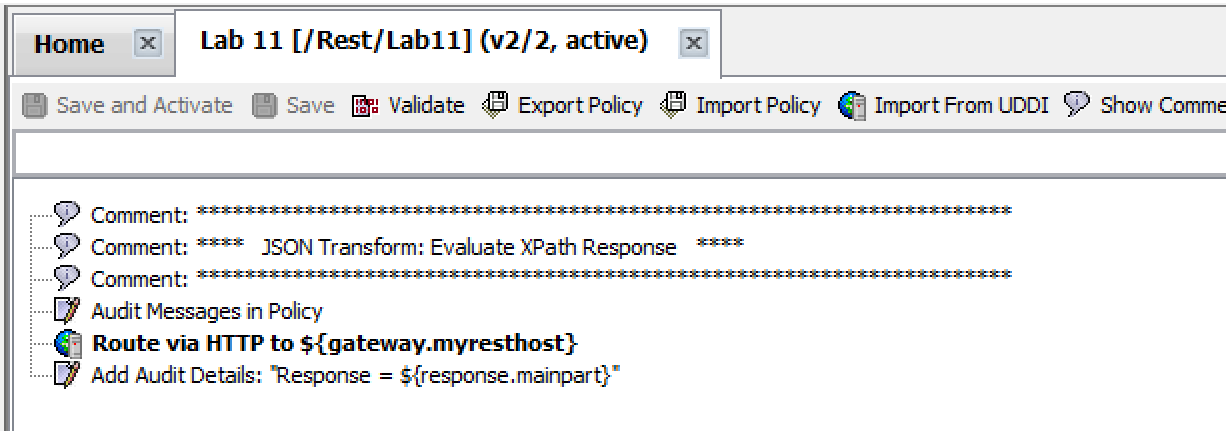


```
Audit Messages in Policy
Add Audit Details: "request url = ${request.url}"
Require HTTP Basic Credentials
Request: Authenticate against sde
Request must match XPath /s:Envelope/s:Body/s0:voonairreservations/s0:startingpoint
Add Audit Details: "startingpoint Found = ${startingpoint.found}startingpoint Element = ${startingpoint.elements}startingpoint Results = ${startingpoint.results}"
At least one assertion must evaluate to true
  All assertions must evaluate to true
    Compare Variable: ${startingpoint.results} is equal to vancouver (case sensitive); If Multivalued all values must pass
    Route via HTTP to ${gateway.mysoaphost}
    Add Audit Details: "response http status = ${response.http.status}"
  All assertions must evaluate to true
    Compare Variable: ${startingpoint.results} is not equal to vancouver (case sensitive); If Multivalued all values must pass
    Return Template Response to Requestor
    Add Audit Details: "response http status = ${response.http.status}"
```

LAB 8B – JSON MESSAGE TRANSFORMATIONS

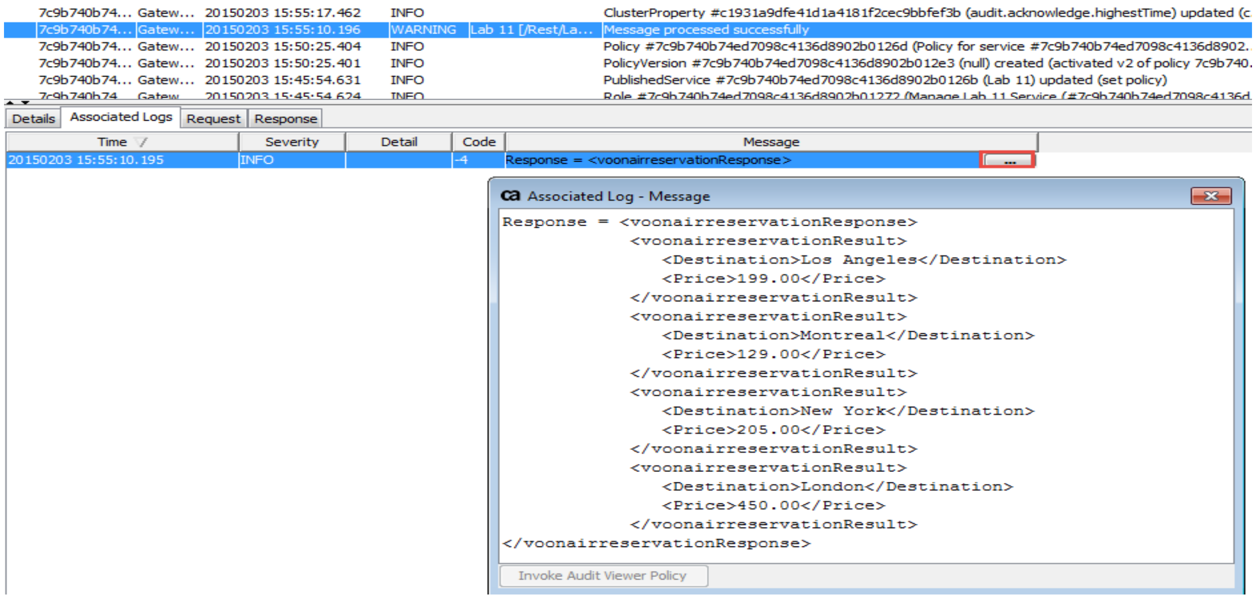
Step 1: Create a new Policy – Publish Web API – New Service = /Rest/Lab11

Step 2: In order to figure out the response from the backend and how it's formed we need to setup our policy with just a routing statement and then place an "Add Audit Detail" assertion highlighting ${response.mainpart}. Like so:



```
Comment: ***********************************************************************
Comment: ****  JSON Transform: Evaluate XPath Response  ****
Comment: ***********************************************************************
Audit Messages in Policy
Route via HTTP to ${gateway.myresthost}
Add Audit Details: "Response = ${response.mainpart}"
```

Run a request through to the URL and then go into the audits to gain access to the response from the back end:



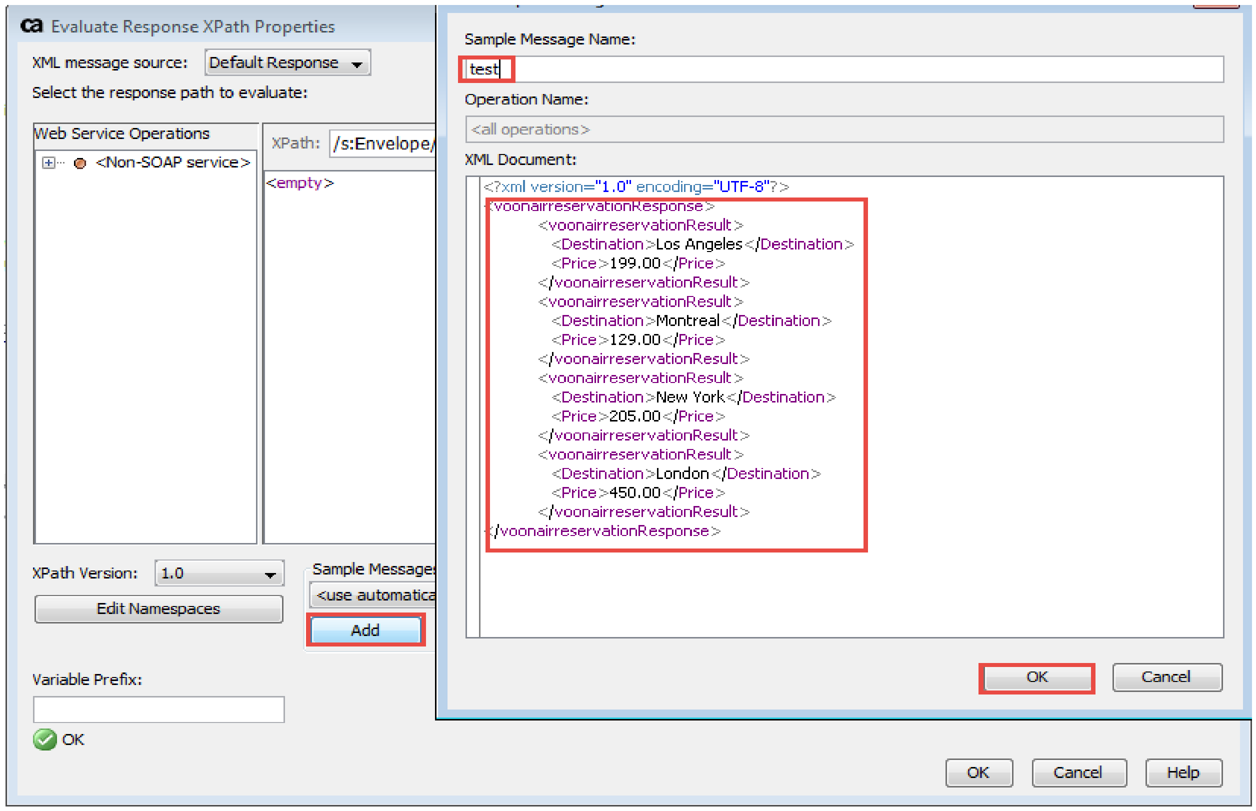
```
7c9b740b74... Gatew... 20150203 15:55:17.462  INFO                           ClusterProperty #c1931a9dfe41d1a4181f2cec9bbfef3b (audit.acknowledge.highestTime) updated (c
7c9b740b74... Gatew... 20150203 15:55:10.196  WARNING  Lab 11 [/Rest/La...  Message processed successfully
7c9b740b74... Gatew... 20150203 15:50:25.404  INFO                           Policy #7c9b740b74ed7098c4136d8902b0126d (Policy for service #7c9b740b74ed7098c4136d8902.
7c9b740b74... Gatew... 20150203 15:50:25.401  INFO                           PolicyVersion #7c9b740b74ed7098c4136d8902b012e3 (null) created (activated v2 of policy 7c9b740.
7c9b740b74... Gatew... 20150203 15:45:54.631  INFO                           PublishedService #7c9b740b74ed7098c4136d8902b0126b (Lab 11) updated (set policy)
7c9b740b74... Gatew... 20150203 15:45:54.624  INFO                           Role #7c9b740b74ed7098c4136d8902b01272 (Manage Lab 11 Service (#7c9b740b74ed7098c4136d
```

Time	Severity	Detail	Code	Message
20150203 15:55:10.195	INFO		-4	Response = <voonairreservationResponse>

```
Response = <voonairreservationResponse>
            <voonairreservationResult>
               <Destination>Los Angeles</Destination>
               <Price>199.00</Price>
            </voonairreservationResult>
            <voonairreservationResult>
               <Destination>Montreal</Destination>
               <Price>129.00</Price>
            </voonairreservationResult>
            <voonairreservationResult>
               <Destination>New York</Destination>
               <Price>205.00</Price>
            </voonairreservationResult>
            <voonairreservationResult>
               <Destination>London</Destination>
               <Price>450.00</Price>
            </voonairreservationResult>
</voonairreservationResponse>
```

Copy out the response message from the audits.

Step 3: Go back to your policy development window and Drag and Drop the "Evaluate response Xpath" assertion and place it below the route.



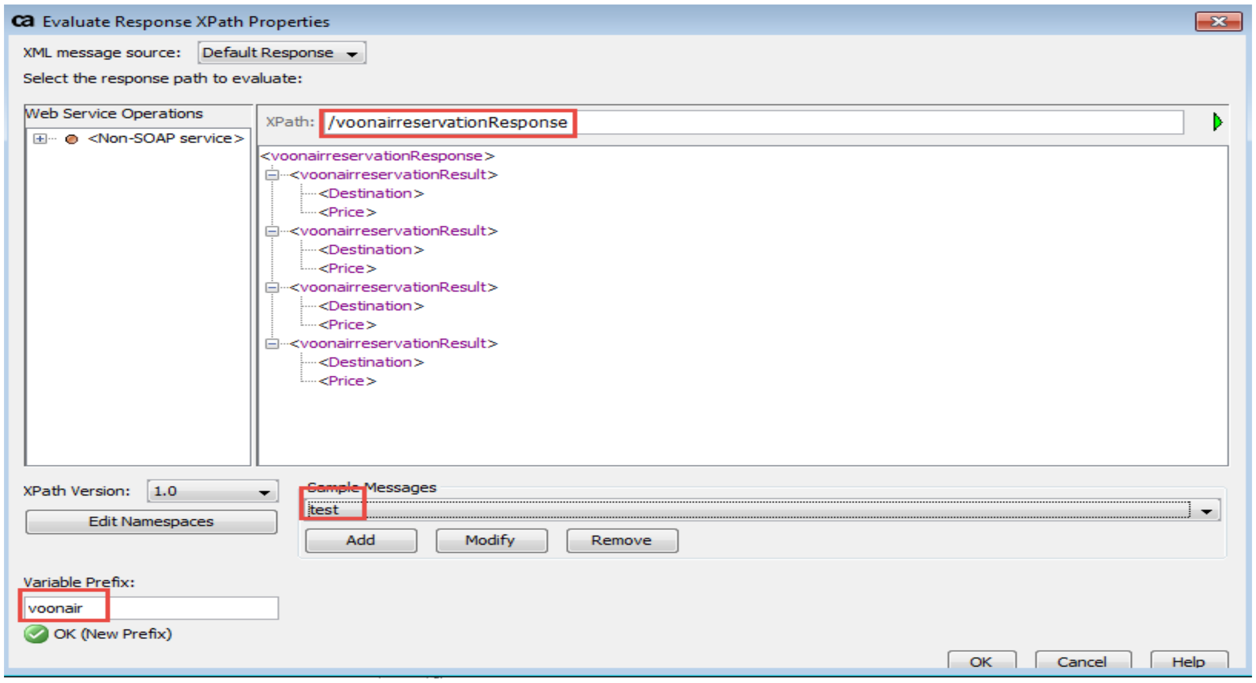
```
<?xml version="1.0" encoding="UTF-8"?>
<voonairreservationResponse>
    <voonairreservationResult>
      <Destination>Los Angeles</Destination>
      <Price>199.00</Price>
    </voonairreservationResult>
    <voonairreservationResult>
      <Destination>Montreal</Destination>
      <Price>129.00</Price>
    </voonairreservationResult>
    <voonairreservationResult>
      <Destination>New York</Destination>
      <Price>205.00</Price>
    </voonairreservationResult>
    <voonairreservationResult>
      <Destination>London</Destination>
      <Price>450.00</Price>
    </voonairreservationResult>
</voonairreservationResponse>
```

Click Add (Sample Messages area)

Sample message window appears; paste the response message from the step previous into the XML Document window. Click Ok.

Select the “voonairreservationResponse” opening element.

Variable Prefix should be = voonair


Evaluate Response XPath Properties

XML message source: Default Response

Select the response path to evaluate:

Web Service Operations

<Non-SOAP service>

XPath: /voonairreservationResponse

```
<voonairreservationResponse>
  <voonairreservationResult>
    <Destination>
    <Price>
  <voonairreservationResult>
    <Destination>
    <Price>
  <voonairreservationResult>
    <Destination>
    <Price>
  <voonairreservationResult>
    <Destination>
    <Price>
```

XPath Version: 1.0

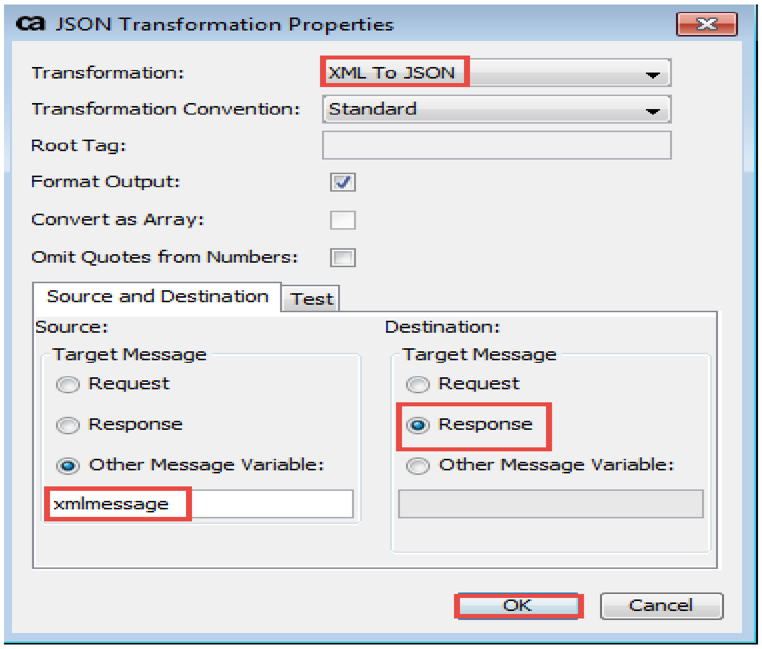
Sample Messages: test

Variable Prefix: voonair

OK (New Prefix)

Step 4: Set a message variable (xmlmessage) with the elements variable (voonair.elements).

Step 5: Drag and Drop the "Apply the JSON Transformation" assertion, below the message variable set above.


JSON Transformation Properties

Transformation: XML To JSON

Transformation Convention: Standard

Root Tag:

Format Output: ✓

Convert as Array:

Omit Quotes from Numbers:

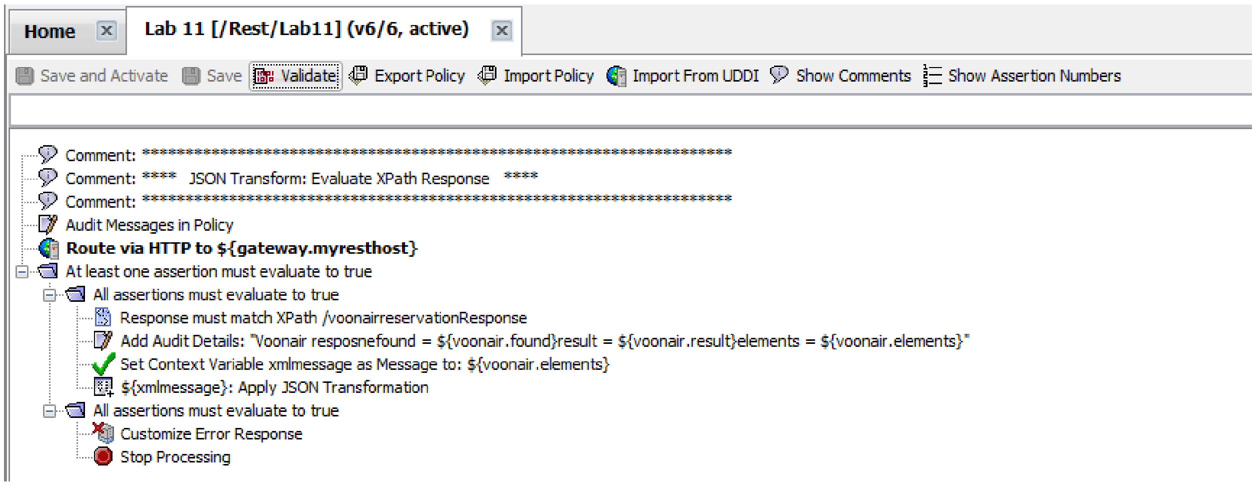
Source and Destination | Test

Source: Target Message: Request [✓ Other Message Variable:] Response — xmlmessage

Destination: Target Message: Request [✓ Response] Other Message Variable:

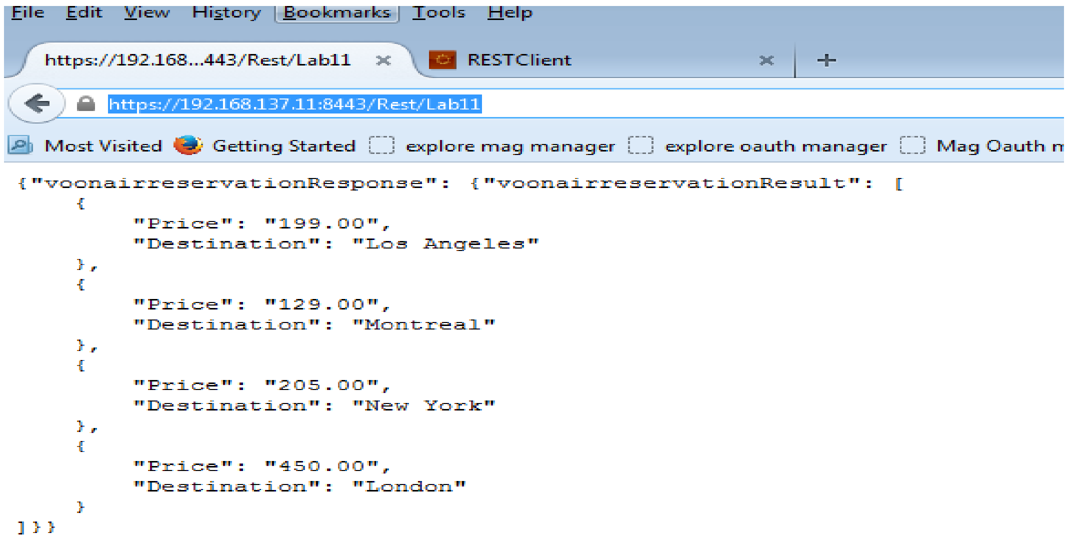
Make sure the target is "xmlmessage" and that you selected the Transformation = "XML to JSON"

Step 7: Set up some sort of failure response as well to get invoked if the xpath fails.



```
Comment: **************************************************************************
Comment: ****  JSON Transform: Evaluate XPath Response  ****
Comment: **************************************************************************
Audit Messages in Policy
Route via HTTP to ${gateway.myresthost}
At least one assertion must evaluate to true
    All assertions must evaluate to true
        Response must match XPath /voonairreservationResponse
        Add Audit Details: "Voonair resposnefound = ${voonair.found}result = ${voonair.result}elements = ${voonair.elements}"
        Set Context Variable xmlmessage as Message to: ${voonair.elements}
        ${xmlmessage}: Apply JSON Transformation
    All assertions must evaluate to true
        Customize Error Response
        Stop Processing
```

When you test from a browser the response should look like this:


File Edit View History Bookmarks Tools Help
https://192.168...443/Rest/Lab11
RESTClient
https://192.168.137.11:8443/Rest/Lab11
Most Visited
Getting Started
explore mag manager
explore oauth manager
Mag Oauth m

```
{"voonairreservationResponse": {"voonairreservationResult": [
    {
        "Price": "199.00",
        "Destination": "Los Angeles"
    },
    {
        "Price": "129.00",
        "Destination": "Montreal"
    },
    {
        "Price": "205.00",
        "Destination": "New York"
    },
    {
        "Price": "450.00",
        "Destination": "London"
    }
]}}
```

MODULE 9: THREAT PROTECTION

LAB 9A: PROTECT AGAINST CODE INJECTION

THE SITUATION

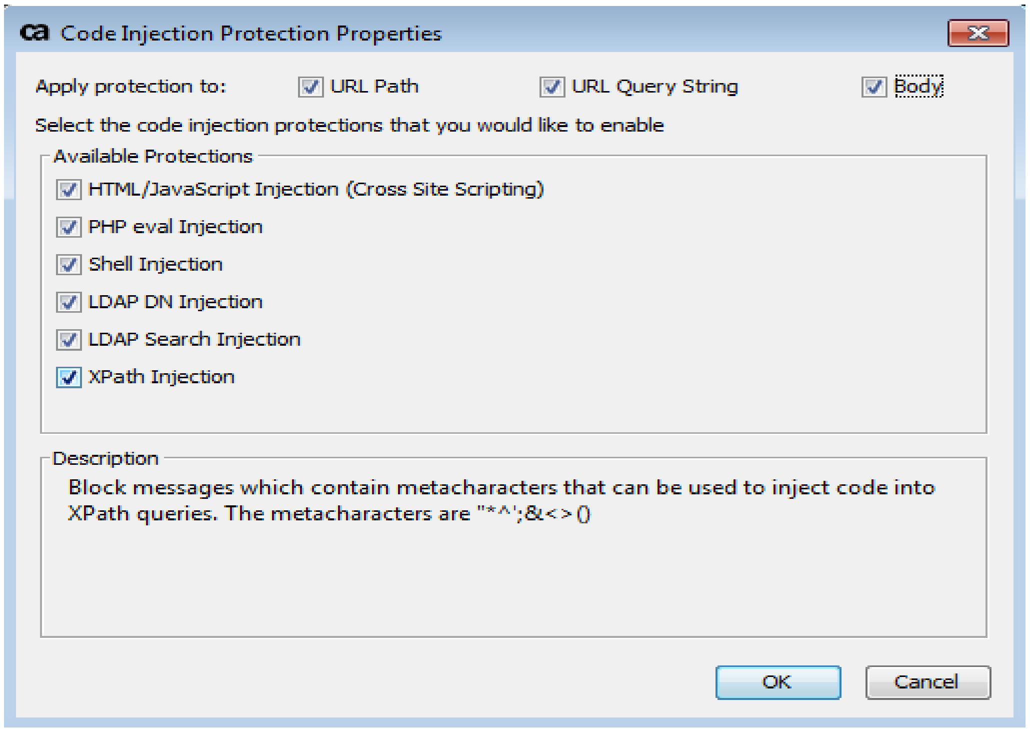
Dave has a request from Architect Andy to secure their web services and prevent bad data from getting to the backend.

THE SOLUTION

Dave must learn how to enforce certain IT Security requirements

Step 1 – Create a new Soap Service. Service URL = lab10

Step 2 – Drag and Drop the "Protect Against Code Injection" assertion into your policy, should be the first line in your policy. Select all in the "Apply protection to" (URL Path, URL Query String, Body) and Select all "Available Protections".


Code Injection Protection Properties

Apply protection to: [✓] URL Path [✓] URL Query String [✓] Body

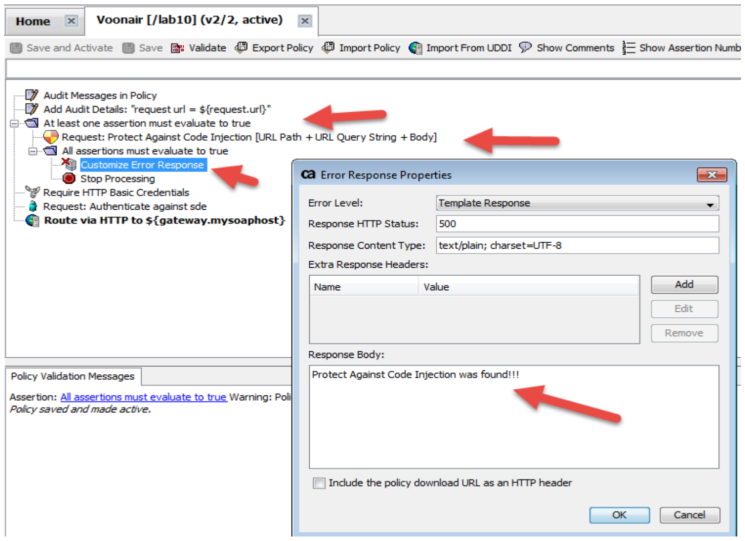
Select the code injection protections that you would like to enable

Available Protections
- [✓] HTML/JavaScript Injection (Cross Site Scripting)
- [✓] PHP eval Injection
- [✓] Shell Injection
- [✓] LDAP DN Injection
- [✓] LDAP Search Injection
- [✓] XPath Injection

Description

Block messages which contain metacharacters that can be used to inject code into XPath queries. The metacharacters are "*^';&<>()

Step 3 – Build out Policy like the following to handle the error that's associated to the "protect Against Code Injection". Place the "At least one assertion" and move the code injection under it. Drag and Drop the "All assertion must evaluate to true" assertion with a Customize error response.



```
Audit Messages in Policy
Add Audit Details: "request url = ${request.url}"
At least one assertion must evaluate to true
    Request: Protect Against Code Injection [URL Path + URL Query String + Body]
    All assertions must evaluate to true
        Customize Error Response
        Stop Processing
Require HTTP Basic Credentials
Request: Authenticate against sde
Route via HTTP to ${gateway.mysoaphost}
```

Assertion: All assertions must evaluate to true Warning: Poli
Policy saved and made active.

Error Response Properties

Error Level:	Template Response
Response HTTP Status:	500
Response Content Type:	text/plain; charset=UTF-8

Extra Response Headers:

Name	Value

Response Body:

```
Protect Against Code Injection was found!!!
```

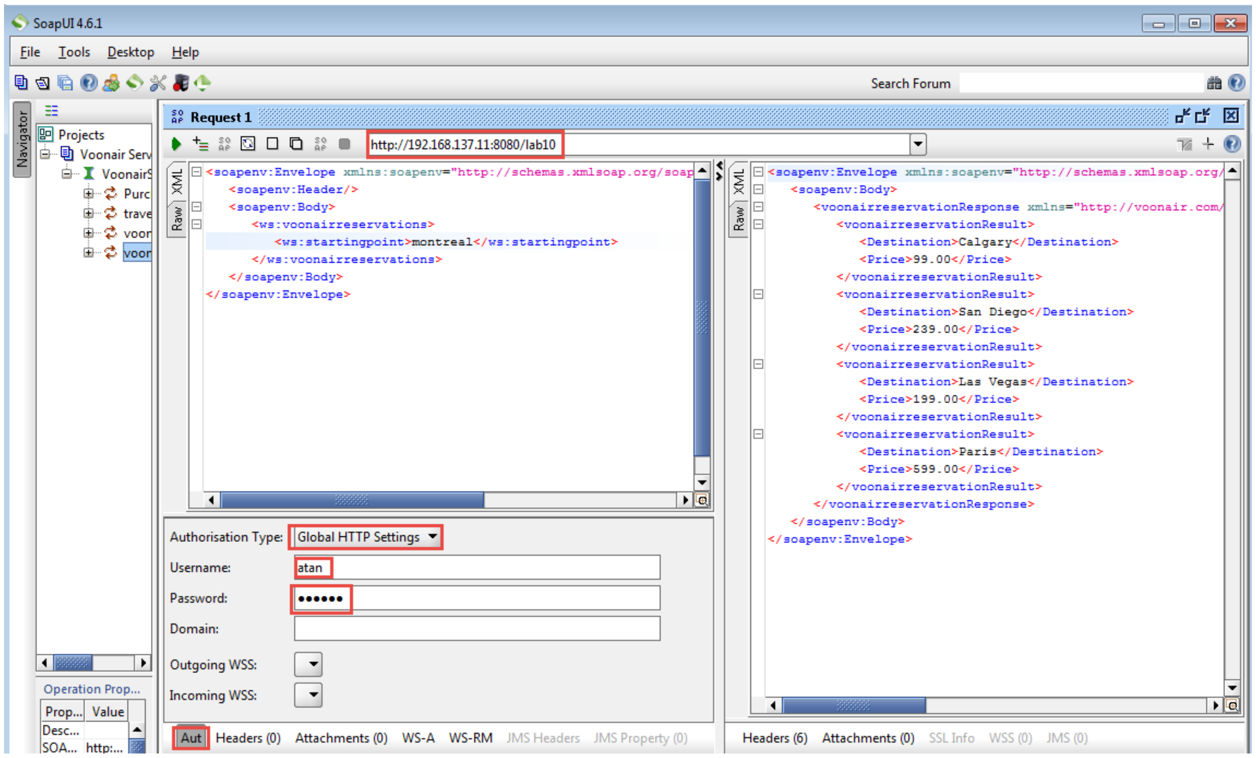
Include the policy download URL as an HTTP header

Add a stop processing assertion below the error response.

Step 4 – Place a require http basic and authentication to the sde assertion. And a route via http to the "mysoaphost".

"Save and Activate".

Step 5 – Go back your soapui project and add authentication to the request, like so:



http://192.168.137.11:8080/lab10

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap
   <soapenv:Header/>
   <soapenv:Body>
      <ws:voonairreservations>
         <ws:startingpoint>montreal</ws:startingpoint>
      </ws:voonairreservations>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
   <soapenv:Body>
      <voonairreservationResponse xmlns="http://voonair.com/
         <voonairreservationResult>
            <Destination>Calgary</Destination>
            <Price>99.00</Price>
         </voonairreservationResult>
         <voonairreservationResult>
            <Destination>San Diego</Destination>
            <Price>239.00</Price>
         </voonairreservationResult>
         <voonairreservationResult>
            <Destination>Las Vegas</Destination>
            <Price>199.00</Price>
         </voonairreservationResult>
         <voonairreservationResult>
            <Destination>Paris</Destination>
            <Price>599.00</Price>
         </voonairreservationResult>
      </voonairreservationResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

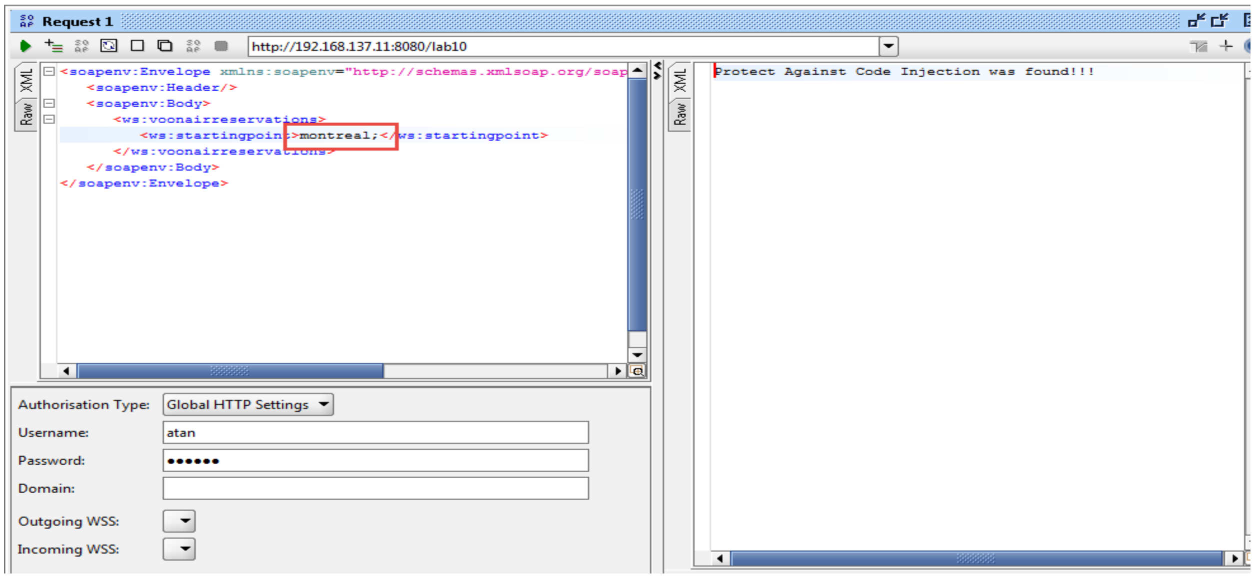
Authorisation Type:	Global HTTP Settings
Username:	atan
Password:	••••••
Domain:	
Outgoing WSS:	
Incoming WSS:	

Username = atan

Password = 7layer

If you test the service like so, this should pass and you should get a response from the backend service.

Step 6 – Now let’s invoke a failure – Place a known “code injection” into the request.



http://192.168.137.11:8080/lab10

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap
   <soapenv:Header/>
   <soapenv:Body>
      <ws:voonairreservations>
         <ws:startingpoint>montreal;</ws:startingpoint>
      </ws:voonairreservations>
   </soapenv:Body>
</soapenv:Envelope>
```

```
Protect Against Code Injection was found!!!
```

Authorisation Type:	Global HTTP Settings
Username:	atan
Password:	••••••
Domain:	
Outgoing WSS:	
Incoming WSS:	

In the startingpoint element type montreal; in the field provided. You should get the response you placed in your customized error response.

***** The stop processing forces the policy to fail. You need to place this in the all assertion after the “customize error response”. The customized error response assertion will only get invoked once there is a failure below it. *****

LAB 9B: INCLUDE POLICY FRAGMENT

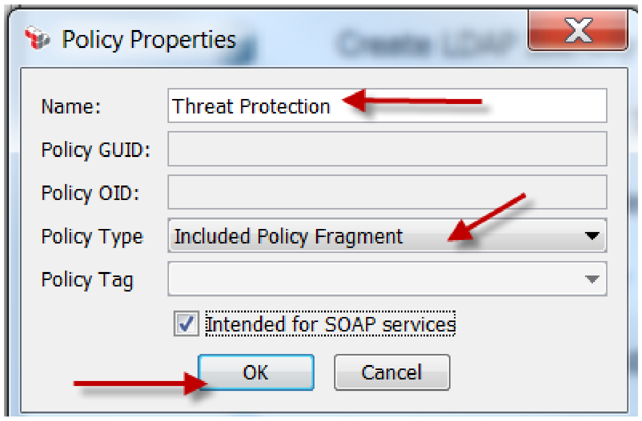
THE SITUATION

Security has asked to apply threat protection services against the services

THE SOLUTION

Create a Policy Fragment and include these threat protection assertions.

Step 1 – Create an "Include Policy Fragment". Select Tasks / Create Policy.



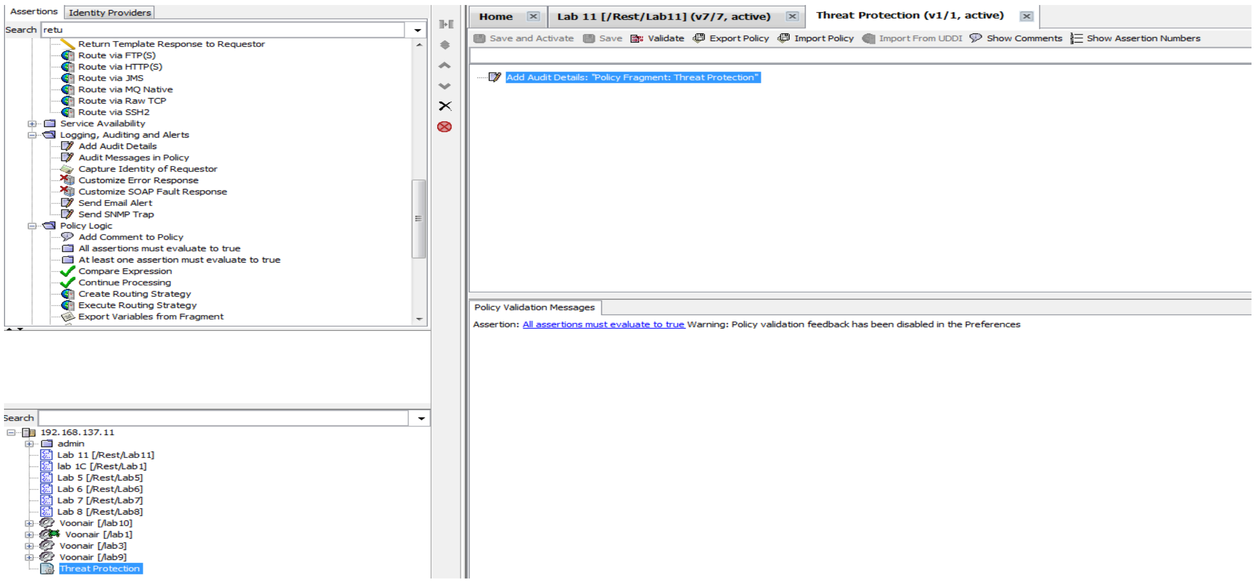
Name: Threat Protection
Policy GUID:
Policy OID:
Policy Type: Included Policy Fragment
Policy Tag:
☑ Intended for SOAP services

Update the Name Field with – Threat Protection

Select Policy Type = "Include Policy fragment"

Select Ok.

Step 2 – The Policy Fragment is created on the route of the policy and services area. And the first line of the policy fragment is with an Add Audit Detail assertion with the name of the policy fragment.



Assertions | Identity Providers

Search: retu

- Return Template Response to Requestor
- Route via FTP(S)
- Route via HTTP(S)
- Route via JMS
- Route via MQ Native
- Route via Raw TCP
- Route via SSH2
- Service Availability
- Logging, Auditing and Alerts
  - Add Audit Details
  - Audit Messages in Policy
  - Capture Identity of Requestor
  - Customize Error Response
  - Customize SOAP Fault Response
  - Send Email Alert
  - Send SNMP Trap
- Policy Logic
  - Add Comment to Policy
  - All assertions must evaluate to true
  - At least one assertion must evaluate to true
  - Compare Expression
  - Continue Processing
  - Create Routing Strategy
  - Execute Routing Strategy
  - Export Variables from Fragment

Search

- 192.168.137.11
  - admin
  - Lab 11 [/Rest/Lab11]
  - lab 1C [/Rest/Lab1]
  - Lab 5 [/Rest/Lab5]
  - Lab 6 [/Rest/Lab6]
  - Lab 7 [/Rest/Lab7]
  - Lab 8 [/Rest/Lab8]
  - Voonair [/lab10]
  - Voonair [/lab1]
  - Voonair [/lab3]
  - Voonair [/lab9]
  - Threat Protection

Home | Lab 11 [/Rest/Lab11] (v7/7, active) | Threat Protection (v1/1, active)

Save and Activate | Save | Validate | Export Policy | Import Policy | Import From UDDI | Show Comments | Show Assertion Numbers

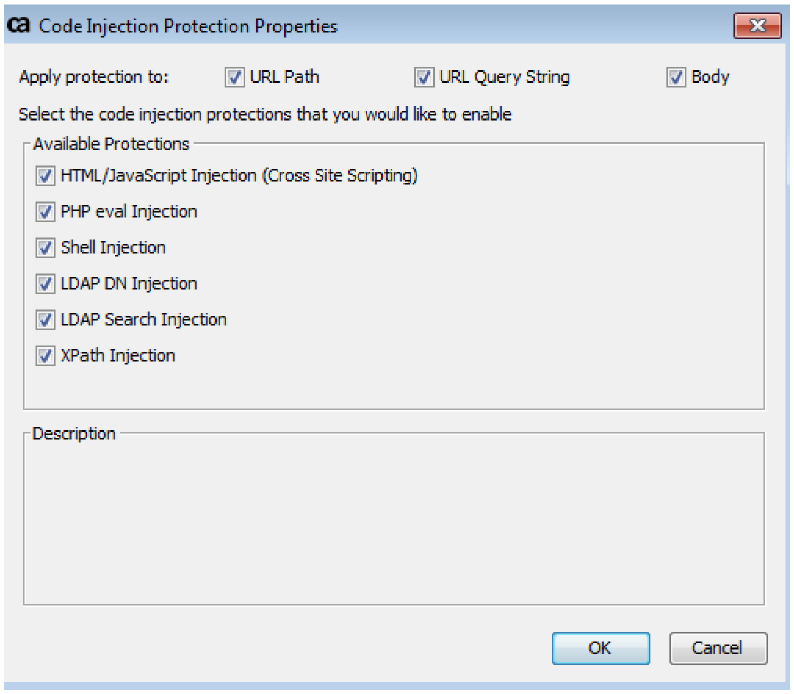
Add Audit Details: "Policy Fragment: Threat Protection"

Policy Validation Messages

Assertion: All assertions must evaluate to true Warning: Policy validation feedback has been disabled in the Preferences

Step 3 - Drag and Drop different threat protection assertions from the Threat Protection folder:

- Protect Against Code Injection
- Protect Against Document Structure Threats
- Protect Against SQL Attacks



Code Injection Protection Properties

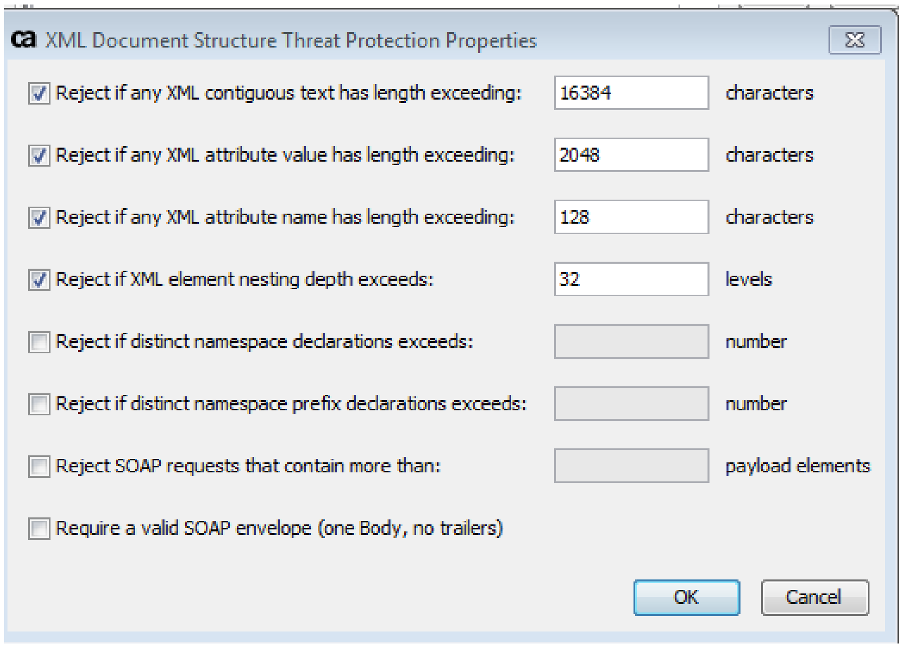
Apply protection to: [✓] URL Path [✓] URL Query String [✓] Body

Select the code injection protections that you would like to enable

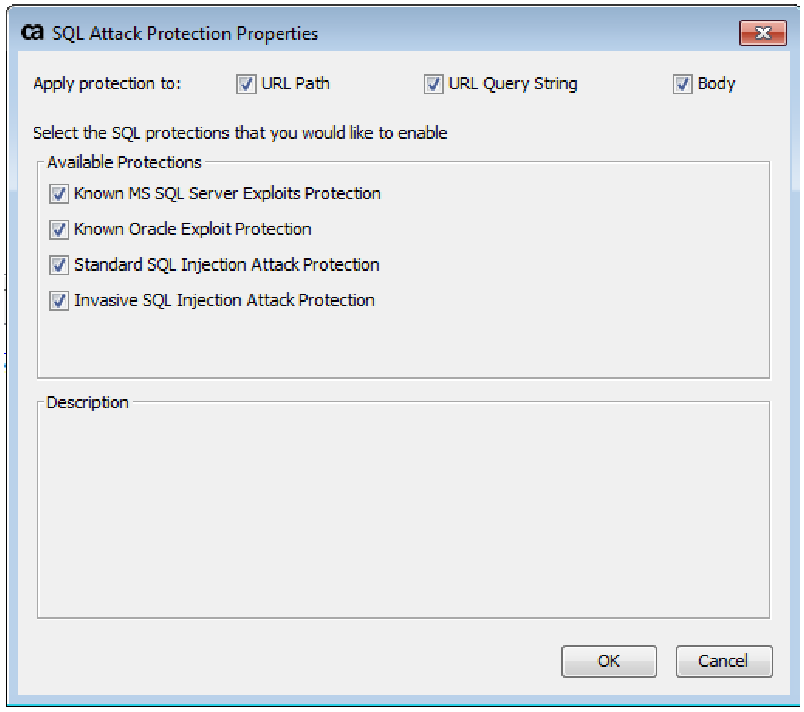
Available Protections

- [✓] HTML/JavaScript Injection (Cross Site Scripting)
- [✓] PHP eval Injection
- [✓] Shell Injection
- [✓] LDAP DN Injection
- [✓] LDAP Search Injection
- [✓] XPath Injection

Description



- [✓] Reject if any XML contiguous text has length exceeding: 16384 characters
- [✓] Reject if any XML attribute value has length exceeding: 2048 characters
- [✓] Reject if any XML attribute name has length exceeding: 128 characters
- [✓] Reject if XML element nesting depth exceeds: 32 levels
- [ ] Reject if distinct namespace declarations exceeds: number
- [ ] Reject if distinct namespace prefix declarations exceeds: number
- [ ] Reject SOAP requests that contain more than: payload elements
- [ ] Require a valid SOAP envelope (one Body, no trailers)

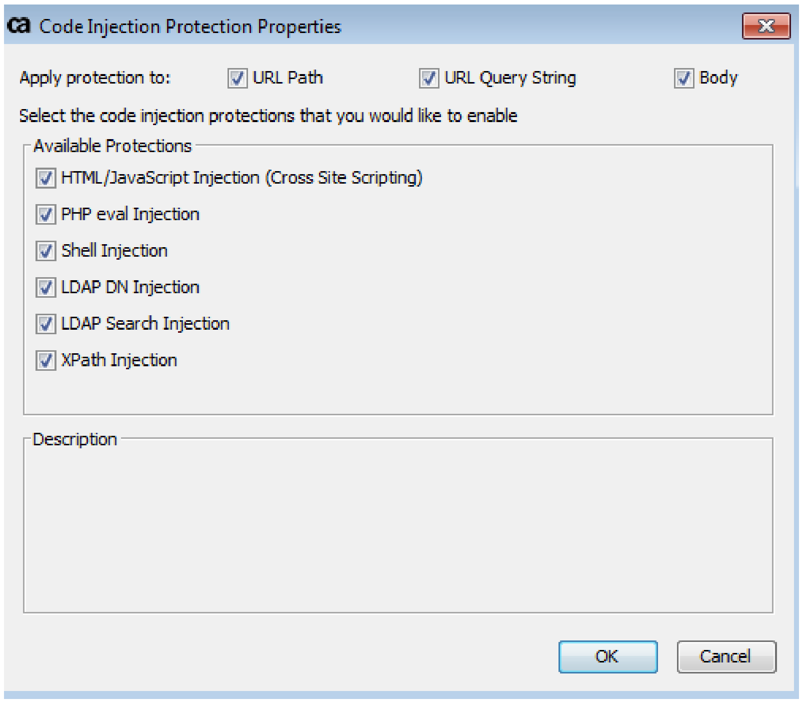


Apply protection to: [✓] URL Path [✓] URL Query String [✓] Body

Select the SQL protections that you would like to enable

Available Protections

- [✓] Known MS SQL Server Exploits Protection
- [✓] Known Oracle Exploit Protection
- [✓] Standard SQL Injection Attack Protection
- [✓] Invasive SQL Injection Attack Protection



Code Injection Protection Properties

Apply protection to: [✓] URL Path [✓] URL Query String [✓] Body

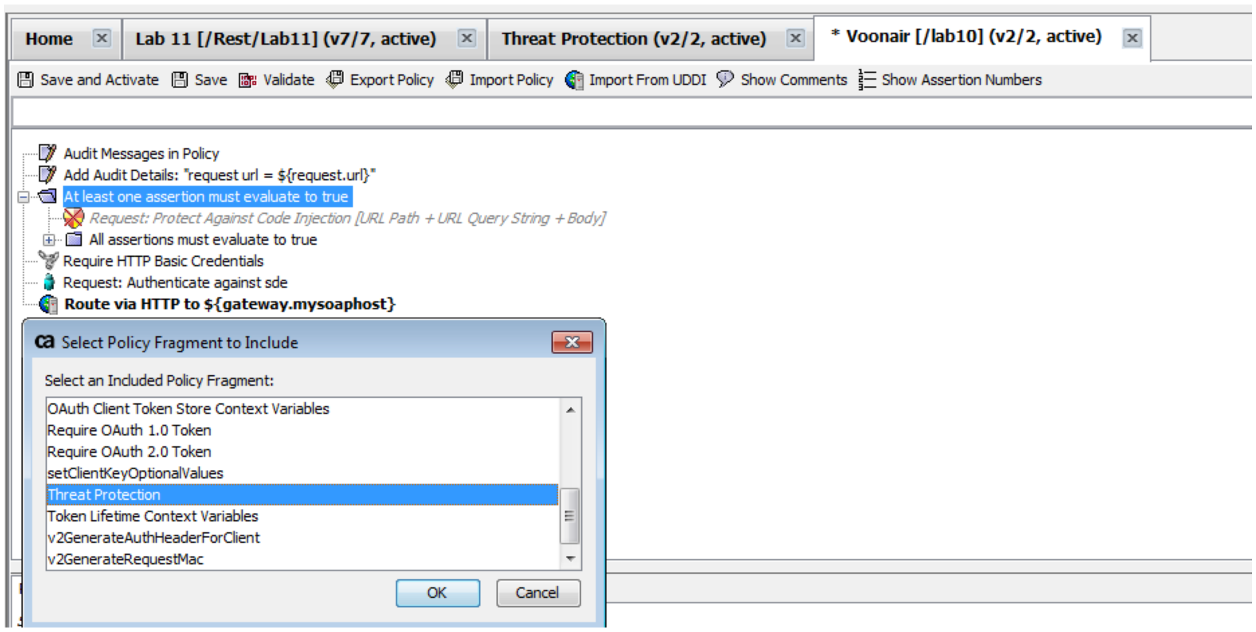
Select the code injection protections that you would like to enable

Available Protections
- [✓] HTML/JavaScript Injection (Cross Site Scripting)
- [✓] PHP eval Injection
- [✓] Shell Injection
- [✓] LDAP DN Injection
- [✓] LDAP Search Injection
- [✓] XPath Injection

Description

Step 4 – Save and Activate.

Step 5 – Add the "Include Policy Fragment" assertion into a policy. Place this policy fragment at the top of the master policy where we would like this enforcement done before executing anything else in our policy.



```
Audit Messages in Policy
Add Audit Details: "request url = ${request.url}"
At least one assertion must evaluate to true
    Request: Protect Against Code Injection [URL Path + URL Query String + Body]
    All assertions must evaluate to true
Require HTTP Basic Credentials
Request: Authenticate against sde
Route via HTTP to ${gateway.mysoaphost}
```

Select Policy Fragment to Include

Select an Included Policy Fragment:

- OAuth Client Token Store Context Variables
- Require OAuth 1.0 Token
- Require OAuth 2.0 Token
- setClientKeyOptionalValues
- Threat Protection
- Token Lifetime Context Variables
- v2GenerateAuthHeaderForClient
- v2GenerateRequestMac

Disable the "Protect Against Code Injection" assertion that we placed in the policy earlier.

Update the customized error response with “Failed because of a possible Threat”

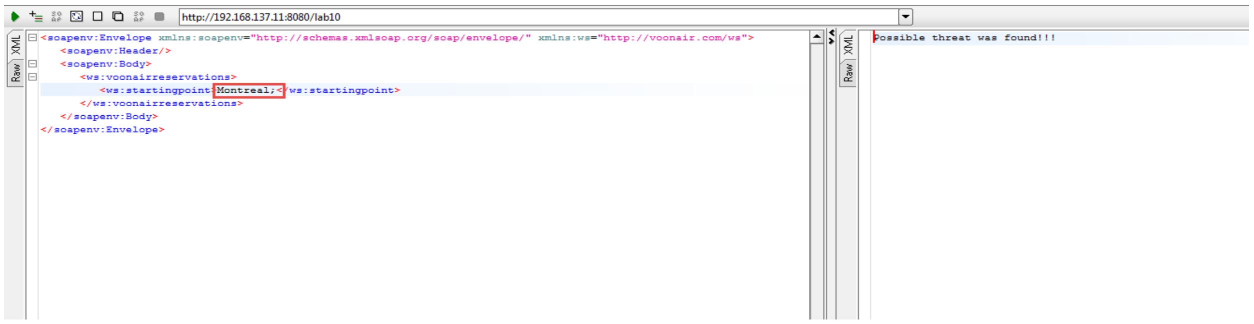
“**Save and Activate**” the policy.

Step 6 – Send a request to the service and note it is executed in-line of the master policy.

Step 7 – Let’s cause the failures now:

Fail with a Code Injection:

Use the “;” (ie. Montreal;) within the startingpoint element again to see if it fails again.

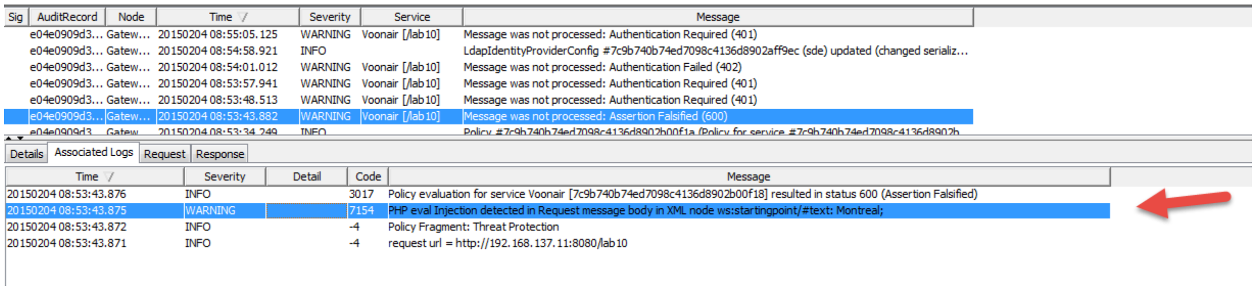


```
http://192.168.137.11:8080/lab10
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://voonair.com/ws">
   <soapenv:Header/>
   <soapenv:Body>
      <ws:voonairreservations>
         <ws:startingpoint>Montreal;</ws:startingpoint>
      </ws:voonairreservations>
   </soapenv:Body>
</soapenv:Envelope>
```

```
Possible threat was found!!!
```

If you go into the Audits, you should see the following:

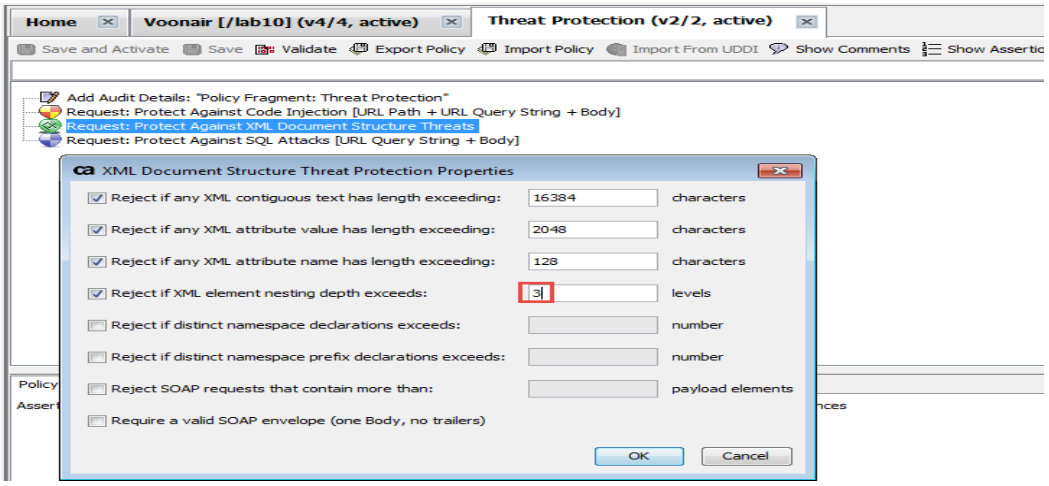


Sig	AuditRecord	Node	Time	Severity	Service	Message
	e04e0909d3...	Gatew...	20150204 08:55:05.125	WARNING	Voonair [/lab10]	Message was not processed: Authentication Required (401)
	e04e0909d3...	Gatew...	20150204 08:54:58.921	INFO		LdapIdentityProviderConfig #7c9b740b74ed7098c4136d8902aff9ec (sde) updated (changed serializ...
	e04e0909d3...	Gatew...	20150204 08:54:01.012	WARNING	Voonair [/lab10]	Message was not processed: Authentication Failed (402)
	e04e0909d3...	Gatew...	20150204 08:53:57.941	WARNING	Voonair [/lab10]	Message was not processed: Authentication Required (401)
	e04e0909d3...	Gatew...	20150204 08:53:48.513	WARNING	Voonair [/lab10]	Message was not processed: Authentication Required (401)
	e04e0909d3...	Gatew...	20150204 08:53:43.882	WARNING	Voonair [/lab10]	Message was not processed: Assertion Falsified (600)
	e04e0909d3	Gatew	20150204 08:53:34.249	INFO		Policy #7c9b740b74ed7098c4136d8902b00f1a (Policy for service #7c9b740b74ed7098c4136d8902b

Time	Severity	Detail	Code	Message
20150204 08:53:43.876	INFO		3017	Policy evaluation for service Voonair [7c9b740b74ed7098c4136d8902b00f18] resulted in status 600 (Assertion Falsified)
20150204 08:53:43.875	WARNING		7154	PHP eval Injection detected in Request message body in XML node ws:startingpoint/#text: Montreal;
20150204 08:53:43.872	INFO		-4	Policy Fragment: Threat Protection
20150204 08:53:43.871	INFO		-4	request url = http://192.168.137.11:8080/lab10

Fail with an XML document structure threat:

Edit the Threat Protection and change the “Reject if XML element nesting depth exceeds:” from 32 to 3.   Click OK and “**Save and Activate**” the policy fragment:



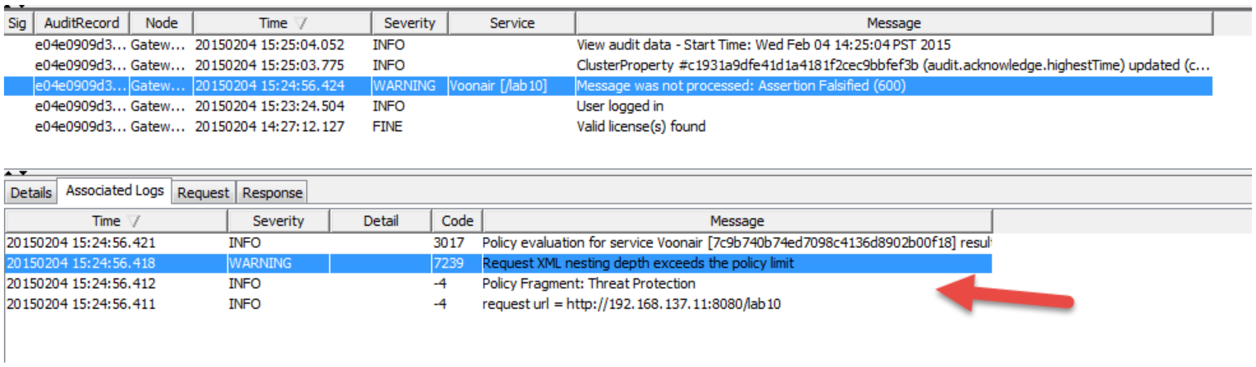
Add Audit Details: "Policy Fragment: Threat Protection"
Request: Protect Against Code Injection [URL Path + URL Query String + Body]
Request: Protect Against XML Document Structure Threats
Request: Protect Against SQL Attacks [URL Query String + Body]

XML Document Structure Threat Protection Properties

[✓] Reject if any XML contiguous text has length exceeding:	16384	characters
[✓] Reject if any XML attribute value has length exceeding:	2048	characters
[✓] Reject if any XML attribute name has length exceeding:	128	characters
[✓] Reject if XML element nesting depth exceeds:	3	levels
[ ] Reject if distinct namespace declarations exceeds:		number
[ ] Reject if distinct namespace prefix declarations exceeds:		number
[ ] Reject SOAP requests that contain more than:		payload elements
[ ] Require a valid SOAP envelope (one Body, no trailers)		

OK Cancel

Run a standard request through the gateway, without the (;), this should give you another type of error, same response:



Sig	AuditRecord	Node	Time	Severity	Service	Message
	e04e0909d3...	Gatew...	20150204 15:25:04.052	INFO		View audit data - Start Time: Wed Feb 04 14:25:04 PST 2015
	e04e0909d3...	Gatew...	20150204 15:25:03.775	INFO		ClusterProperty #c1931a9dfe41d1a4181f2cec9bbfef3b (audit.acknowledge.highestTime) updated (c...
	e04e0909d3...	Gatew...	20150204 15:24:56.424	WARNING	Voonair [/lab10]	Message was not processed: Assertion Falsified (600)
	e04e0909d3...	Gatew...	20150204 15:23:24.504	INFO		User logged in
	e04e0909d3...	Gatew...	20150204 14:27:12.127	FINE		Valid license(s) found

Details | Associated Logs | Request | Response

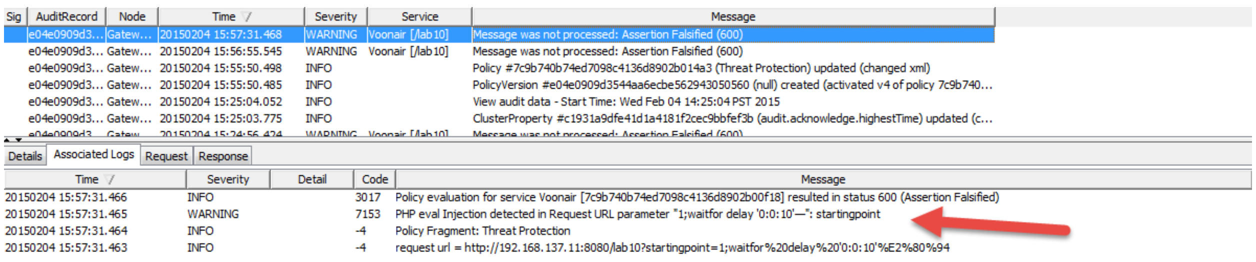
Time	Severity	Detail	Code	Message
20150204 15:24:56.421	INFO		3017	Policy evaluation for service Voonair [7c9b740b74ed7098c4136d8902b00f18] resul
20150204 15:24:56.418	WARNING		7239	Request XML nesting depth exceeds the policy limit
20150204 15:24:56.412	INFO		-4	Policy Fragment: Threat Protection
20150204 15:24:56.411	INFO		-4	request url = http://192.168.137.11:8080/lab10

Change the XML Document structure back to 32 from 3. Save and Activate the Policy Fragment.

Fail with a SQL Attack threat:

In the URL for your test case type the following to invoke a possible threat.

http://Gateway_URL:8080/lab10?startingpoint=1;waitfor delay '0:0:10'—



Sig	AuditRecord	Node	Time	Severity	Service	Message
	e04e0909d3...	Gatew...	20150204 15:57:31.468	WARNING	Voonair [/lab10]	Message was not processed: Assertion Falsified (600)
	e04e0909d3...	Gatew...	20150204 15:56:55.545	WARNING	Voonair [/lab10]	Message was not processed: Assertion Falsified (600)
	e04e0909d3...	Gatew...	20150204 15:55:50.498	INFO		Policy #7c9b740b74ed7098c4136d8902b014a3 (Threat Protection) updated (changed xml)
	e04e0909d3...	Gatew...	20150204 15:55:50.485	INFO		PolicyVersion #e04e0909d3544aa6ecbe562943050560 (null) created (activated v4 of policy 7c9b740...
	e04e0909d3...	Gatew...	20150204 15:25:04.052	INFO		View audit data - Start Time: Wed Feb 04 14:25:04 PST 2015
	e04e0909d3...	Gatew...	20150204 15:25:03.775	INFO		ClusterProperty #c1931a9dfe41d1a4181f2cec9bbfef3b (audit.acknowledge.highestTime) updated (c...
	e04e0909d3	Gatew...	20150204 15:24:56.424	WARNING	Voonair [/lab10]	Message was not processed: Assertion Falsified (600)

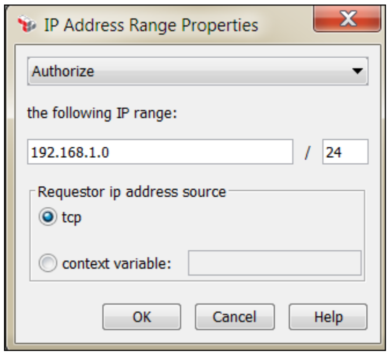
Details | Associated Logs | Request | Response

Time	Severity	Detail	Code	Message
20150204 15:57:31.466	INFO		3017	Policy evaluation for service Voonair [7c9b740b74ed7098c4136d8902b00f18] resulted in status 600 (Assertion Falsified)
20150204 15:57:31.465	WARNING		7153	PHP eval Injection detected in Request URL parameter "1;waitfor delay '0:0:10'—": startingpoint
20150204 15:57:31.464	INFO		-4	Policy Fragment: Threat Protection
20150204 15:57:31.463	INFO		-4	request url = http://192.168.137.11:8080/lab10?startingpoint=1;waitfor%20delay%20'0:0:10'%E2%80%94

LAB 9C: RESTRICT SERVICE AVAILABILITY

Grant access to service 24x7 for a known company IP address but ONLY on weekends and evenings for everyone else.

SOLUTION:



Authorize

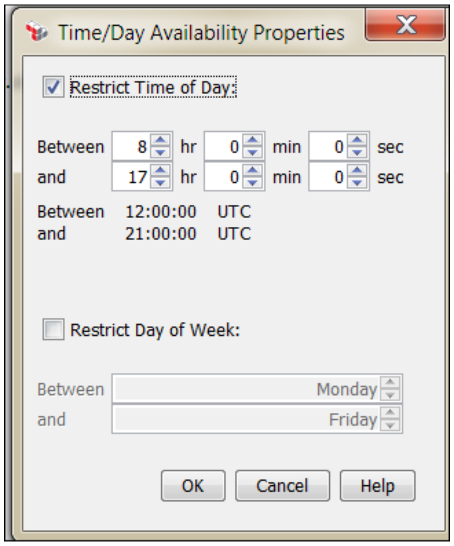
the following IP range:

192.168.1.0 / 24

Requestor ip address source

[✓ tcp]

context variable:



[✓ Restrict Time of Day:]

Between	8	hr	0	min	0	sec
and	17	hr	0	min	0	sec

Between 12:00:00 UTC
and 21:00:00 UTC

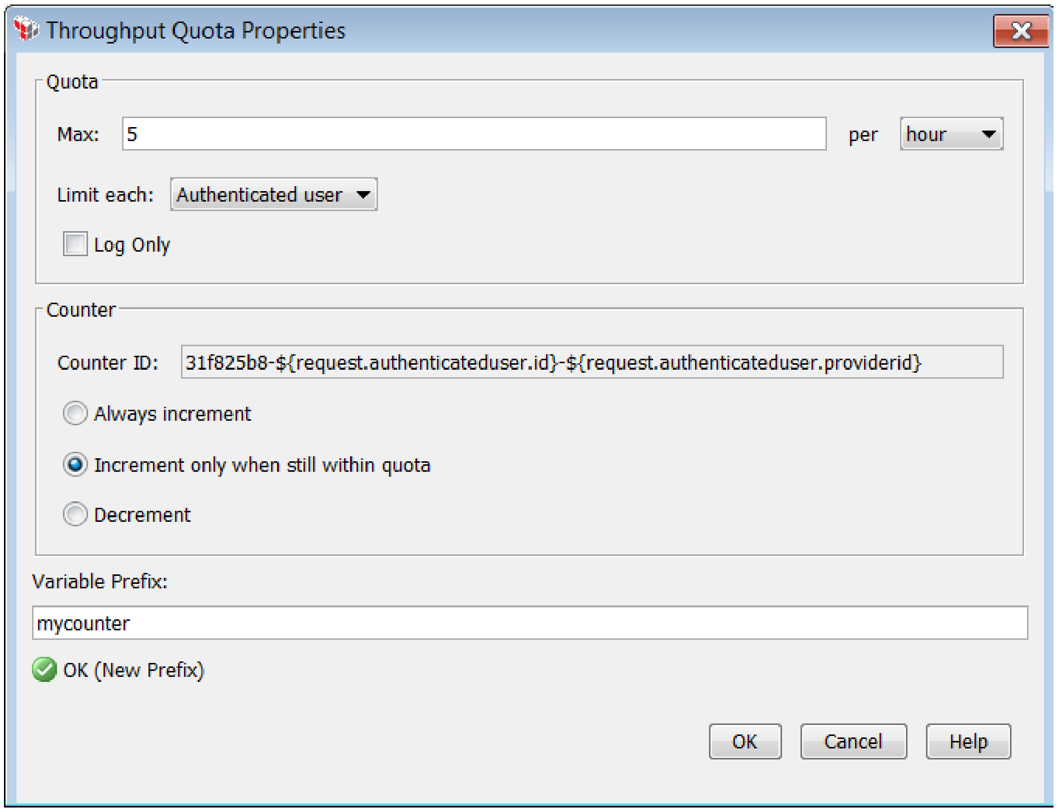
Restrict Day of Week:

Between Monday
and Friday

LAB 9D: LIMIT THROUGHPUT

SITUATION

Create a policy that limits access to a service to only 5x per hour per authenticated user. When the maximum is met, create a return template response.


Throughput Quota Properties

Quota

Max: 5 per hour

Limit each: Authenticated user

[ ] Log Only

Counter

Counter ID: 31f825b8-${request.authenticateduser.id}-${request.authenticateduser.providerid}

Always increment [✓ Increment only when still within quota] Decrement

Variable Prefix: mycounter

OK (New Prefix)

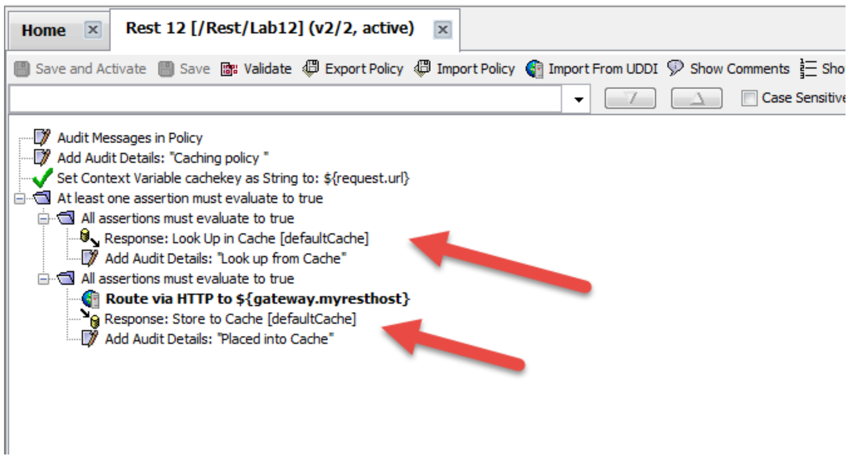
LAB 9E: SERVICE AVAILABILITY CACHE RESPONSE

Using cache where possible is extremely valuable from a performance and perhaps cost saving basis. Many companies use this for storing static or public data, but careful that you use it wisely and make sure the cache keys are unique for user data where necessary.

SOLUTION:

Step 1 – Create a Publish Web API Service, Name it /Rest/Lab12

Step 2 – Use the lookup in cache and the store to cache feature.



Audit Messages in Policy
Add Audit Details: "Caching policy "
Set Context Variable cachekey as String to: ${request.url}
At least one assertion must evaluate to true
All assertions must evaluate to true
Response: Look Up in Cache [defaultCache]
Add Audit Details: "Look up from Cache"
All assertions must evaluate to true
Route via HTTP to ${gateway.myresthost}
Response: Store to Cache [defaultCache]
Add Audit Details: "Placed into Cache"

Section 3: Gateway Operations

AUDITING AND LOGGING

LAB 1A: AUDITING

THE SITUATION

You need to create an audit sink policy.

Step 1 – From Tasks > Manage Log/Audit Sinks > Manage Audit Sinks

Step 2 – Import the sample policy: auditsinkpolicy-layer7-example.xml

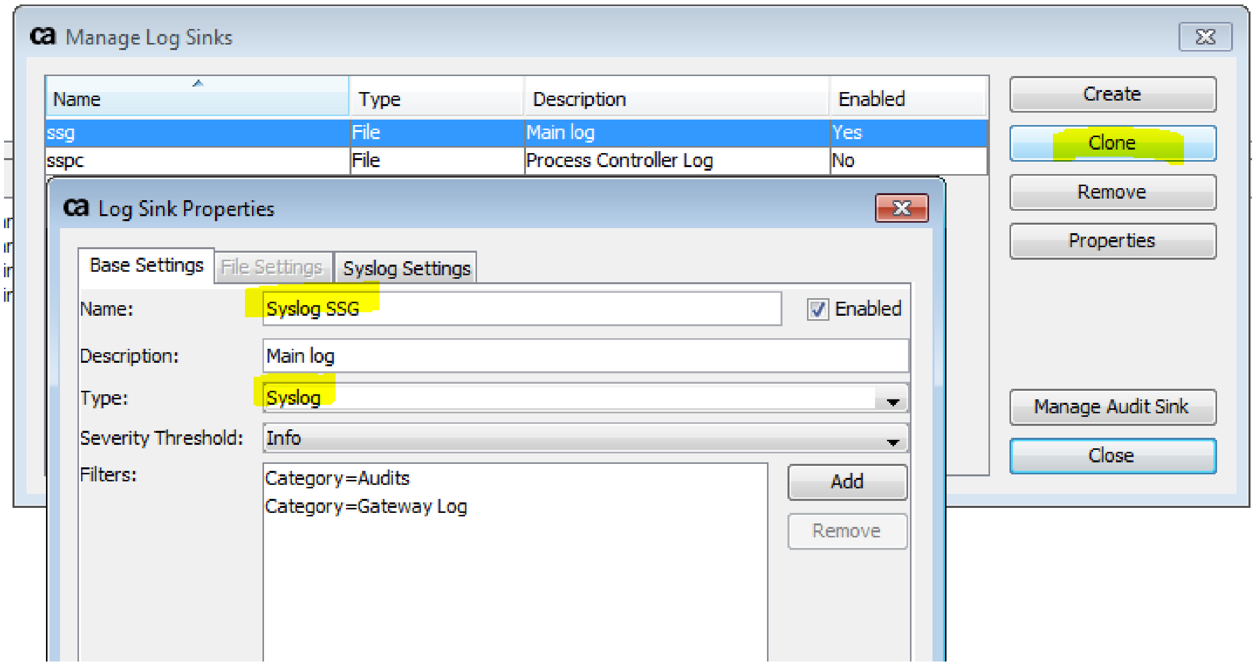
LAB 1B: LOGGING

THE SITUATION

Operations Oscar needs to setup the syslog server so that he can off-box the logs.

Step 1 – From Tasks > Manage Log/Audit Sinks

Step 2 – Highlight ssg log and click clone.

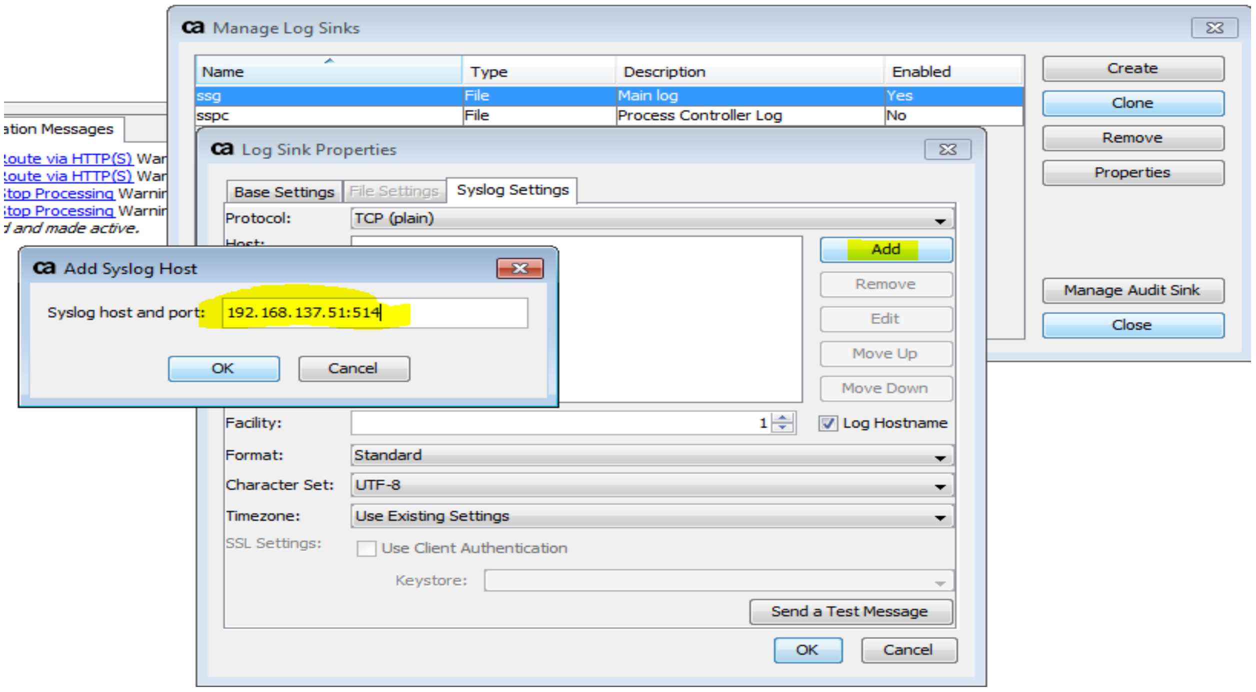


Name	Type	Description	Enabled
ssg	File	Main log	Yes
sspc	File	Process Controller Log	No

Name: Syslog SSG ✓ Enabled
Description: Main log
Type: Syslog
Severity Threshold: Info
Filters: Category=Audits
Category=Gateway Log

Step 3 – Place a proper name for the syslog ssg log sink. Select Syslog within the down the Type.

Step 4 – Click the Syslog Settings tab.



Name	Type	Description	Enabled
ssg	File	Main log	Yes
sspc	File	Process Controller Log	No

Protocol: TCP (plain)
Host:
Syslog host and port: 192.168.137.51:514
Facility: 1 ✓ Log Hostname
Format: Standard
Character Set: UTF-8
Timezone: Use Existing Settings
SSL Settings: Use Client Authentication
Keystore:

Step 5 – Click Add and type the location of the syslog server. Click ok and then close at the manage log sinks wizard.

Step 6 – Go onto the syslog server and view the gateways log files. Navigate to the following directory:

Cd /var/log/syslog-ng/{gateway name}

Tail –f messages

SYSTEM CONFIGURATION

LAB 4: CREATE A LISTENER PORT

THE SITUATION

By default we have 4 different listener ports. The listen ports open up an actual port that listens on the Gateway for any traffic that comes from the travel agency. When it receives a request from that travel agency, it will act accordingly. A listen port is a TCP port that 'listens' for incoming messages that are then passed to the Gateway message processor. This lab will walk you through how to configure the proper port for the Policy Manager.

Step 1 – Select Tasks – Manage Listen Ports

Step 2 – Click Create

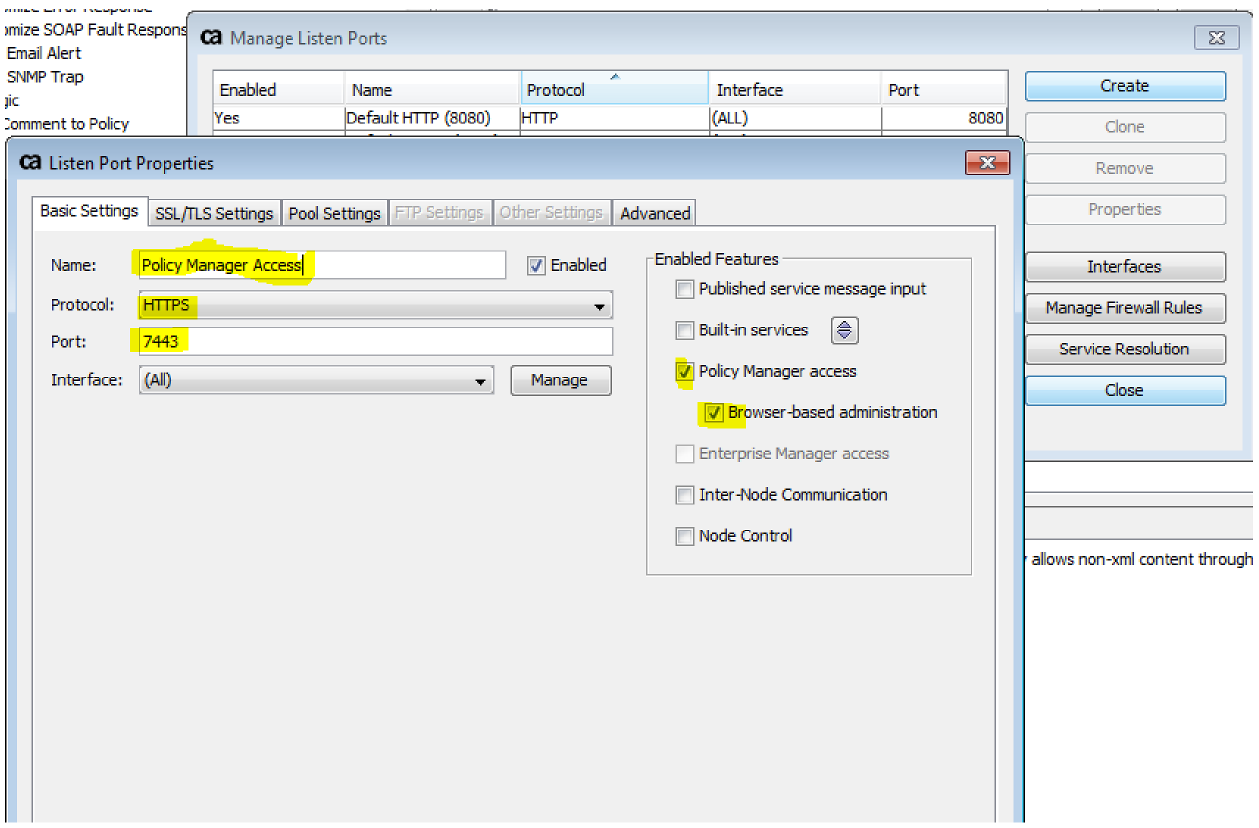
Step 3 – Edit the following properties within the wizard:

Name = Policy Manager Access

Protocols = https

Port = 7443

Enabled Features – Select "Policy Manager Access" and "Browser-based administration"

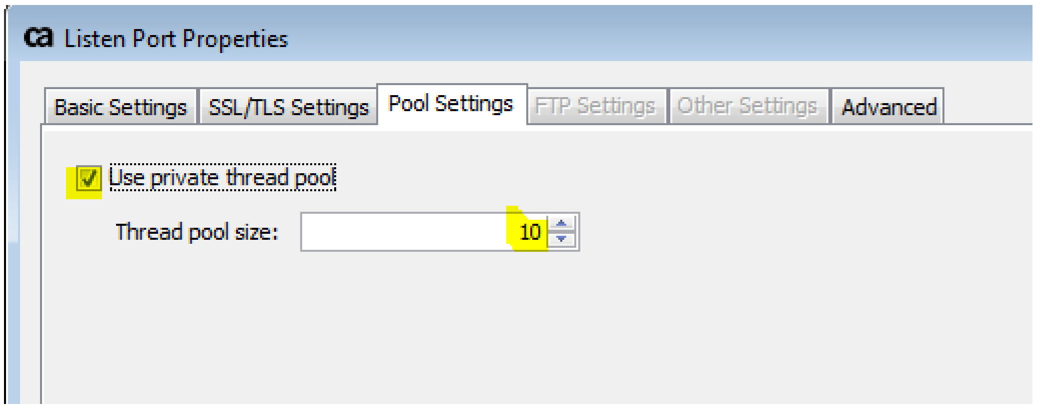


Enabled	Name	Protocol	Interface	Port
Yes	Default HTTP (8080)	HTTP	(ALL)	8080

Name: Policy Manager Access [✓ Enabled]
Protocol: HTTPS
Port: 7443
Interface: (All)

Enabled Features
- [ ] Published service message input
- [ ] Built-in services
- [✓] Policy Manager access
  - [✓] Browser-based administration
- [ ] Enterprise Manager access
- [ ] Inter-Node Communication
- [ ] Node Control

Step 4 – Click on Pool Settings tab and select "Use private thread pool" – guarantee 10 threads for accessing the policy manager.



[✓] Use private thread pool
Thread pool size: 10

Step 5 – Click OK and Close.

Step 6 – Disconnect and re-connect to the new port. When you log into the policy manager make sure the URL has port 7443 at the end of it. For example = gateway.ca.com:7443