# The things used for the Demo

1. WSO2 Enterprise Service Bus (ESB)
2. WSO2 Identity Server (IS)
3. A Web Service ( SimpleStockQuoteService of WSO2 ESB Samples )

# The demo setup

1. Uses the SimpleStockQuoteService (SOAP) as the backend
2. Uses WSO2 ESB to mediate it, and expose in the form of a REST API
3. Uses WSO2 IS as an OAuth server to authenticate the REST API

# Steps

* Start SimpleStockQuoteService | http://{host}:9000/services

<https://docs.wso2.com/display/ESB481/Sample+0%3A+Introduction+to+ESB>

* Set port offsets the products as follow

ESB port offset = 10

IS port offset = 20

* Start the ESB and IS
* Open the provided projects with WSO2 Developer Studio
  + ***AllyBank-CompositeApplication***
  + *AllyBank-ESBProject*
  + *AllyBank-RegistryProject*
  + *CustomMediatorProject*
* Export the Composite Archive (*AllyBank-CompositeApplication*) from the WSO2 Developer Studio (save as \*.car)
* Deploy the CAR file, on WSO2 ESB

<http://wso2.com/library/articles/2011/09/create-deploy-car-file-standalone-wso2-server-wso2-stratos/>

* Log into IS management console and create a service provider and an Inbound OAuth authentication

<https://docs.wso2.com/display/IS500/Configuring+Inbound+Authentication+for+a+Service+Provider>

* Get client\_key and client\_secret
* Use token API of IS to generate Access Token

curl -v -k -X POST --user client\_key:client\_secret -H "Content-Type: application/x-www-form-urlencoded;charset=UTF-8" -d 'grant\_type=password&username=admin&password=admin' https://localhost:9463/oauth2/token

… or use the provided SOAP-UI/ReadyAPI project providing the client\_key:client\_secret combination encoded in base64 to provide with basic auth header.

Authorization **Basic** {encodeBase64(client\_key**:**client\_secret)}

*Use something like* [*https://www.base64encode.org/*](https://www.base64encode.org/)

* A response needs to be delivered similar to,

**{**

**"token\_type": "bearer",**

**"expires\_in": 3299,**

**"refresh\_token": "8e63353e4f9b4468c95a12d9311093",**

**"access\_token": "e7dba3dc394fd2a077ee9854bb135"**

**}**

* Invoke the StockAPI using the generated access\_token

Authorization **Bearer** {access\_token}

* A response needs to be delivered similar to,

**{"getQuoteResponse": {"return": {**

**"@type": "ax21:GetQuoteResponse",**

**"change": 4.027511698373331,**

**"earnings": 13.259567133412038,**

**"high": 152.02893879212672,**

**"last": 146.7808331484884,**

**"lastTradeTimestamp": "Thu May 14 02:33:30 EDT 2015",**

**"low": 152.13151943771885,**

**"marketCap": 5.5819323194555E7,**

**"name": "IBM Company",**

**"open": 152.40419357640985,**

**"peRatio": 25.310205964012297,**

**"percentageChange": 2.401643393800928,**

**"prevClose": 167.69815655267809,**

**"symbol": "IBM",**

**"volume": 18470**

**}}}**

* Mediation flow - described

<?xml version="1.0" encoding="UTF-8"?>  
<api xmlns="http://ws.apache.org/ns/synapse" name="StockAPIX" context="/stockapix">  
 <resource methods="GET" uri-template="/{**symbol**}">  
 <inSequence>  
 <!-- OAuth based validation -->  
 <sequence key="AuthSequence" />  
 <!-- logs incoming message -->  
 <log level="full" description="log message" />  
 <!-- extracts 'symbol' value form URL (resource) and preserves it within mediation context (synapse scope) -->  
 <property name="symbol" expression="get-property('uri.var.symbol')" scope="default" type="STRING" description="preserve &quot;symbol&quot;" />  
 <!-- logs the preserved 'symbol' value -->  
 <log level="custom" description="log symbol">  
 <property name="Symbol is " expression="get-property('symbol')" />  
 </log>  
 <!-- builds a brand new SOAP message payload (using the extracted symbol value)-->  
 <sequence key="BuildPayloadSequence" />  
 <!-- Injects a new header called 'Action' -->  
 <header name="Action" scope="default" value="getQuote" />  
 <!-- delivers the new SOAP message to a backend service -->  
 <send>  
 <endpoint key="conf:endpoints/StockQuoteEndpoint.xml" />  
 </send>  
 </inSequence>  
 <outSequence>  
 <!-- reads response from the backend, and sets a new property called 'riskFactor' within mediation context -->  
 <class name="com.demo.mediators.SampleClassMediator" />  
 <!-- logs 'riskFactor' peroperty value -->  
 <log level="custom">  
 <property name="RISK" expression="get-property('riskFactor')" />  
 </log>  
 <!-- logs the response -->  
 <log level="full" description="log message" />  
 <!-- converts XML response into JSON -->  
 <property name="messageType" value="application/json" scope="axis2" type="STRING" description="application/json" />  
 <!-- sends response to the client -->  
 <send />  
 </outSequence>  
 <faultSequence>  
 <log level="custom" description="log message">  
 <property name="ALERT" value="ERROR OCCURED" />  
 </log>  
 </faultSequence>  
 </resource>  
</api>

<?xml version="1.0" encoding="UTF-8"?>

<api xmlns="http://ws.apache.org/ns/synapse" name="StockAPIX" context="/stockapix">

<resource methods="GET" uri-template="/{symbol}">

<inSequence>

<!-- OAuth based validation -->

<sequence key="AuthSequence"/>

<!-- logs incoming message -->

<log level="full" description="log message"/>

<!-- extracts 'symbol' value form URL (resource) and preserves it within mediation context (synapse scope) -->

<property name="symbol" expression="get-property('uri.var.symbol')" scope="default" type="STRING" description="preserve &quot;symbol&quot;"/>

<!-- logs the preserved 'symbol' value -->

<log level="custom" description="log symbol">

<property name="Symbol is " expression="get-property('symbol')"/>

</log>

<!-- builds a brand new SOAP message payload (using the extracted symbol value)-->

<sequence key="BuildPayloadSequence"/>

<!-- Injects a new header called 'Action' -->

<header name="Action" scope="default" value="getQuote"/>

<!-- delivers the new SOAP message to a backend service -->

<send>

<endpoint key="conf:endpoints/StockQuoteEndpoint.xml"/>

</send>

</inSequence>

<outSequence>

<!-- reads response from the backend, and sets a new property called 'riskFactor' within mediation context -->

<class name="com.demo.mediators.SampleClassMediator"/>

<!-- logs 'riskFactor' peroperty value -->

<log level="custom">

<property name="RISK" expression="get-property('riskFactor')"/>

</log>

<!-- logs the response -->

<log level="full" description="log message"/>

<!-- converts XML response into JSON -->

<property name="messageType" value="application/json" scope="axis2" type="STRING" description="application/json"/>

<!-- sends response to the client -->

<send/>

</outSequence>

<faultSequence>

<log level="custom" description="log message">

<property name="ALERT" value="ERROR OCCURED"/>

</log>

</faultSequence>

</resource>

</api>

