**Practical Exercise: Protocol Switching**

Training Objective

Configure an integration solution that will translate messages between JMS and HTTP

protocols.

Prerequisites

You will need JAVA installed locally to be able to run the back-end service that is used for this practical exercise.

High-level Steps

* Create an integration project.
* Create a proxy service.
* Configure a JMS message broker as messages will be received on the JMS protocol.
* Test your configuration.

Detailed Instructions

Switching from JMS to HTTP(S)

This example demonstrates how the Micro Integrator receives a messages over the JMS transport and forwards it over an HTTP/S transport. In this sample, the client sends a request message to the proxy service exposed in JMS. The Micro Integrator forwards this message to the HTTP endpoint and returns the reply back to the client through a JMS temporary queue.

Synapse configuration

Following are the integration artifacts (proxy service) that we can used to implement this scenario. See the instructions on how to [build and run](https://apim.docs.wso2.com/en/latest/integrate/examples/protocol-switching/switching_from_jms_to_http/#build-and-run) this example.

<**proxy** xmlns="http://ws.apache.org/ns/synapse" name="JMStoHTTPStockQuoteProxy" transports="jms">

<**target**>

<**inSequence**>

<**property** action="set" name="OUT\_ONLY" value="true"/>

<**send**>

<**endpoint**>

<**address** uri="http://localhost:9000/services/SimpleStockQuoteService"/>

</**endpoint**>

</**send**>

</**inSequence**>

</**target**>

<**parameter** name="transport.jms.ContentType">

<**rules**>

<**jmsProperty**>contentType</**jmsProperty**>

<**default**>text/xml</**default**>

</**rules**>

</**parameter**>

<**parameter** name="transport.jms.Destination">Queue1</**parameter**>

<**parameter** name="transport.jms.ConnectionFactory">myQueueListener</**parameter**>

</**proxy**>

Build and Run

Create the artifacts:

1. [Set up WSO2 Integration Studio](https://apim.docs.wso2.com/en/4.1.0/integrate/develop/installing-wso2-integration-studio).
2. [Create an integration project](https://apim.docs.wso2.com/en/4.1.0/integrate/develop/create-integration-project) with an **ESB Configs** module and an **Composite Exporter**.
3. Create the [proxy service](https://apim.docs.wso2.com/en/4.1.0/integrate/develop/creating-artifacts/creating-a-proxy-service) with the configurations given above.
4. [Deploy the artifacts](https://apim.docs.wso2.com/en/4.1.0/integrate/develop/deploy-artifacts) in your Micro Integrator.
5. Start the selected message broker and create a queue with name **Queue1**.
6. [Configure MI with the selected message broker](https://apim.docs.wso2.com/en/4.1.0/install-and-setup/setup/mi-setup/brokers/configure-with-activemq) and start the Micro-Integrator.

Set up the back-end service:

1. Download the [back-end service](https://github.com/wso2-docs/WSO2_EI/blob/master/Back-End-Service/axis2Server.zip)
2. Extract the downloaded zip file.
3. Open a terminal, navigate to the axis2Server/bin/ directory inside the extracted folder.
4. Execute the following command to start the axis2server with the SimpleStockQuote back-end service:

On MacOS/Linux/CentOS

sh axis2server.sh

On Windows

Publish the following XML message to the Queue1.

<**soapenv:Envelope** xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

<**soapenv:Header**/>

<**soapenv:Body**>

<**m0:placeOrder** xmlns:m0="http://services.samples">

<**m0:order**>

<**m0:price**>172.23182849731984</**m0:price**>

<**m0:quantity**>18398</**m0:quantity**>

<**m0:symbol**>IBM</**m0:symbol**>

</**m0:order**>

</**m0:placeOrder**>

</**soapenv:Body**>

</**soapenv:Envelope**>