

Lab: Applying Security to a REST API

# Training Objective

Learn how to securely expose a REST API through the Micro Integrator.

# High-Level Steps

* Create a REST API by engaging the default Basic Auth handler.
* Conﬁgure an LDAP user store.
* Start the back-end service.
* Invoke the REST API securely.
* Try out other options for securing REST APIs.

# Detailed Instructions

# Securing REST APIs

In most of the real-world use cases of REST, when a consumer attempts to access a privileged resource, access will be denied unless the consumer's credentials are provided in an Authorization header. By default, the Micro Integrator validates the credentials of the consumer (that is provided in the Authorization header) against the credentials of users that are registered in the [user store connected to the server](https://apim.docs.wso2.com/en/latest/install-and-setup/setup/mi-setup/user_stores/setting_up_a_userstore/).

**Info**

The Micro Integrator uses a **Basic Auth handler** for this purpose. If required, you can use a custom basic auth handler or other security implementations. Find out more about [applying security to REST APIs](https://apim.docs.wso2.com/en/latest/integrate/develop/advanced-development/applying-security-to-an-api).

## **Synapse configuration**

Following is a sample REST API configuration 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/rest_api_examples/securing-rest-apis/#build-and-run) this example.

**Note**

The basic auth handler is engaged in the API as follows:

<**handlers**>

<**handler** class="org.wso2.micro.integrator.security.handler.RESTBasicAuthHandler"/>

</**handlers**>

See the REST API given below for an example of how the default basic auth handler is used.

<**api** xmlns="http://ws.apache.org/ns/synapse" name="StockQuoteAPI" context="/stockquote">

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

<**inSequence**>

<**payloadFactory** media-type="xml">

<**format**>

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

<**m0:request**>

<**m0:symbol**>$1</**m0:symbol**>

</**m0:request**>

</**m0:getQuote**>

</**format**>

<**args**>

<**arg** evaluator="xml" expression="get-property('uri.var.symbol')"/>

</**args**>

</**payloadFactory**>

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

<**send**>

<**endpoint**>

<**address** uri="http://localhost:9000/services/SimpleStockQuoteService" format="soap11"/>

</**endpoint**>

</**send**>

</**inSequence**>

<**outSequence**>

<**send**/>

</**outSequence**>

<**faultSequence**/>

</**resource**>

<**handlers**>

<**handler** class="org.wso2.micro.integrator.security.handler.RESTBasicAuthHandler"/>

</**handlers**>

</**api**>

## **Build and run**

Create the artifacts:

1. [Set up WSO2 Integration Studio](https://apim.docs.wso2.com/en/latest/integrate/develop/installing-wso2-integration-studio).
2. [Create an integration project](https://apim.docs.wso2.com/en/latest/integrate/develop/create-integration-project) with an **ESB Configs** module and an **Composite Exporter**.
3. [Create the rest API](https://apim.docs.wso2.com/en/latest/integrate/develop/creating-artifacts/creating-an-api) with the configurations given above.
4. [Deploy the artifacts](https://apim.docs.wso2.com/en/latest/integrate/develop/deploy-artifacts) in your Micro Integrator.

[Configure an external user store](https://apim.docs.wso2.com/en/latest/install-and-setup/setup/mi-setup/user_stores/setting_up_a_userstore).

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

Test the API:

1. First, invoke the service using the following service URL without providing any user credentials: http://127.0.0.1:8290/stockquote/view/IBM

**Info**

You can invoke the service using Postman or Curl.

curl -v http://127.0.0.1:8290/stockquote/view/IBM

Note that you will receive the following error because the username and password are not passed and the service cannot be authenticated: 401 Unauthorized

1. Now, invoke the service again by providing the credentials of a user that is registered in the user store that is hosted.

**Info**

Note that the credentials (YWRtaW46YWRtaW4=) given in the authorization header (Authorization: Basic YWRtaW46YWRtaW4=) are the Base64-encoded username and password in the following format: username:password.

The request is passed to the back-end service and you will receive a response similar to what is shown below:

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

<**soapenv:Body**>

<**ns:getQuoteResponse** xmlns:ns="http://services.samples">

<**ns:return** xmlns:ax21="http://services.samples/xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ax21:GetQuoteResponse">

<**ax21:change**>-2.6989539095024164</**ax21:change**>

<**ax21:earnings**>12.851852793420885</**ax21:earnings**>

<**ax21:high**>-166.81703170012037</**ax21:high**>

<**ax21:last**>170.03627716039932</**ax21:last**>

<**ax21:lastTradeTimestamp**>Mon Jul 30 15:10:56 IST 2018</**ax21:lastTradeTimestamp**>

<**ax21:low**>178.02122263133768</**ax21:low**>

<**ax21:marketCap**>-7306984.135450081</**ax21:marketCap**>

<**ax21:name**>IBM Company</**ax21:name**>

<**ax21:open**>-165.86249647643422</**ax21:open**>

<**ax21:peRatio**>23.443106773044992</**ax21:peRatio**>

<**ax21:percentageChange**>1.5959734616866617</**ax21:percentageChange**>

<**ax21:prevClose**>-169.11019978052138</**ax21:prevClose**>

<**ax21:symbol**>IBM</**ax21:symbol**>

<**ax21:volume**>9897</**ax21:volume**>

</**ns:return**>

</**ns:getQuoteResponse**>

</**soapenv:Body**>

</**soapenv:Envelope**>

# Applying Security to an API

## **Using a Basic Auth handler**

A Basic Authentication handler is enabled in the Micro Integrator by default. See the example on [securing an API with basic auth](https://apim.docs.wso2.com/en/latest/integrate/examples/rest_api_examples/securing-rest-apis).

## **Using a custom basic auth handler**

If required, you can implement a custom basic auth handler (instead of the default handler explained above). The following example of a primitive security handler serves as a template that can be used to write your own security handler to secure an API.

### **Prerequisites**

**Before you begin**, be sure to [configure a user store](https://apim.docs.wso2.com/en/latest/install-and-setup/setup/mi-setup/user_stores/setting_up_a_userstore/) for the Micro Integrator and add the required users and roles.

### **Creating the custom handler**

The custom Basic Auth handler in this sample simply verifies whether the request uses username: admin and password: admin. Following is the code for this handler:

**package** org.wso2.rest;

**import** org.apache.commons.codec.binary.Base64;

**import** org.apache.synapse.MessageContext;

**import** org.apache.synapse.core.axis2.Axis2MessageContext;

**import** org.apache.synapse.core.axis2.Axis2Sender;

**import** org.apache.synapse.rest.Handler;

**import** java.util.Map;

**public** **class** BasicAuthHandler **implements** Handler {

**public** **void** addProperty(String s, Object o) {

//To change body of implemented methods use File | Settings | File Templates.

}

**public** Map getProperties() {

**return** **null**; //To change body of implemented methods use File | Settings | File Templates.

}

**public** **boolean** handleRequest(MessageContext messageContext) {

org.apache.axis2.context.MessageContext axis2MessageContext

= ((Axis2MessageContext) messageContext).getAxis2MessageContext();

Object headers = axis2MessageContext.getProperty(

org.apache.axis2.context.MessageContext.TRANSPORT\_HEADERS);

**if** (headers != **null** && headers **instanceof** Map) {

Map headersMap = (Map) headers;

**if** (headersMap.get("Authorization") == **null**) {

headersMap.clear();

axis2MessageContext.setProperty("HTTP\_SC", "401");

headersMap.put("WWW-Authenticate", "Basic realm=\"WSO2 ESB\"");

axis2MessageContext.setProperty("NO\_ENTITY\_BODY", **new** Boolean("true"));

messageContext.setProperty("RESPONSE", "true");

messageContext.setTo(**null**);

Axis2Sender.sendBack(messageContext);

**return** **false**;

} **else** {

String authHeader = (String) headersMap.get("Authorization");

**if** (processSecurity(credentials)) {

**return** **true**;

} **else** {

headersMap.clear();

axis2MessageContext.setProperty("HTTP\_SC", "403");

axis2MessageContext.setProperty("NO\_ENTITY\_BODY", **new** Boolean("true"));

messageContext.setProperty("RESPONSE", "true");

messageContext.setTo(**null**);

Axis2Sender.sendBack(messageContext);

**return** **false**;

}

}

}

**return** **false**;

}

**public** **boolean** handleResponse(MessageContext messageContext) {

**return** **true**;

}

**public** **boolean** processSecurity(String credentials) {

String decodedCredentials = **new** String(**new** Base64().decode(credentials.getBytes()));

String usernName = decodedCredentials.split(":")[0];

String password = decodedCredentials.split(":")[1];

**if** ("admin".equals(username) && "admin".equals(password)) {

**return** **true**;

} **else** {

**return** **false**;

}

}

}

You can build the project (mvn clean install) for this handler by accessing its source from here: https://github.com/wso2/product-esb/tree/v5.0.0/modules/samples/integration-scenarios/starbucks\_sample/BasicAuth-handler

**Note**

When building the sample using the source ensure you update pom.xml with the online repository. To do this, add the following section before <dependencies> tag in pom.xml :

<**repositories**>

<**repository**>

<**id**>wso2-nexus</**id**>

<**name**>WSO2 internal Repository</**name**>

<**url**>http://maven.wso2.org/nexus/content/groups/wso2-public/</**url**>

<**releases**>

<**enabled**>true</**enabled**>

<**updatePolicy**>daily</**updatePolicy**>

<**checksumPolicy**>ignore</**checksumPolicy**>

</**releases**>

</**repository**>

<**repository**>

<**id**>wso2-maven2-repository</**id**>

<**name**>WSO2 Maven2 Repository</**name**>

<**url**>http://dist.wso2.org/maven2</**url**>

<**snapshots**>

<**enabled**>false</**enabled**>

</**snapshots**>

<**releases**>

<**enabled**>true</**enabled**>

<**updatePolicy**>never</**updatePolicy**>

<**checksumPolicy**>fail</**checksumPolicy**>

</**releases**>

</**repository**>

</**repositories**>

Alternatively, you can download the JAR file from the following location, copy it to the MI\_HOME/lib directory, and restart the Micro Integrator: https://github.com/wso2/product-esb/blob/v5.0.0/modules/samples/integration-scenarios/starbucks\_sample/bin/WSO2-REST-BasicAuth-Handler-1.0-SNAPSHOT.jar

### **Creating the REST API**

Add the handler to the REST API:

<**api** xmlns="http://ws.apache.org/ns/synapse" name="StockQuoteAPI" context="/stockquote">

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

<**inSequence**>

<**payloadFactory** media-type="xml">

<**format**>

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

<**m0:request**>

<**m0:symbol**>$1</**m0:symbol**>

</**m0:request**>

</**m0:getQuote**>

</**format**>

<**args**>

<**arg** evaluator="xml" expression="get-property('uri.var.symbol')"/>

</**args**>

</**payloadFactory**>

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

<**send**>

<**endpoint**>

<**address** uri="http://localhost:9000/services/SimpleStockQuoteService" format="soap11"/>

</**endpoint**>

</**send**>

</**inSequence**>

<**outSequence**>

<**send**/>

</**outSequence**>

<**faultSequence**/>

</**resource**>

<**handlers**>

<**handler** class="org.wso2.rest.BasicAuthHandler"/>

</**handlers**>

</**api**>

You can now send a request to the secured API.

 [CC BY 4.0](http://creativecommons.org/licenses/by/4.0/)