




# WSO2 API Manager 4.1.0 Fundamentals - Integration Profile

Developing Unit Test Suite for Artifacts



WSO2 Training

CC by 4.0



## Module Objective

At the end of this module, attendees will be able to:

- Understand the basics of WSO2 Micro Integrator's Unit Test Framework.
- Understand how to implement unit tests for integration artifacts.





## WSO2 MI Unit Test Framework

- Build unit tests for integration artifacts.
- Build mock services to simulate mock backends.
- Test on either a local server or a remote server.



# Developing Unit Tests





## Supported Artifact Types

- Mediation Sequences
- Proxy services
- APIs
- Artifacts with Registry Resources
- Artifacts using Connectors



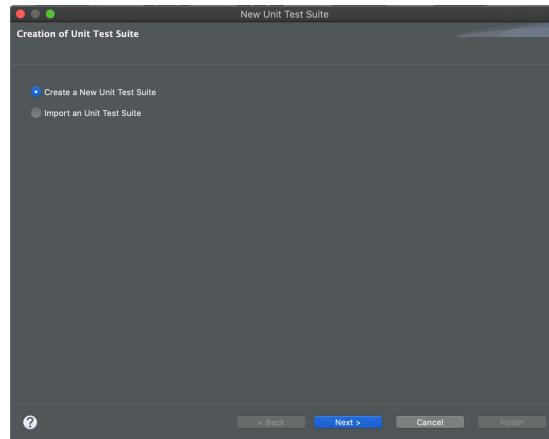
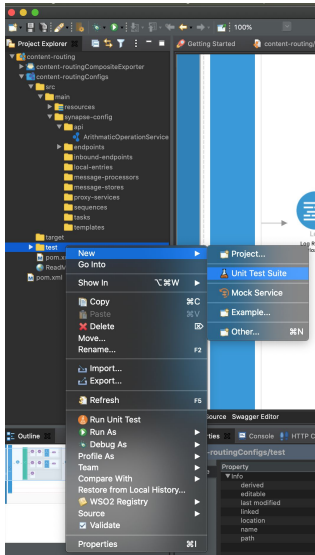


## Unit Test Suite

- Allows grouping of unit tests into a single suite.
- Either a sequence, API, or proxy service should be specified per test suite.
- Select the dependent artifacts for a given main artifact.
- Specify mock service (backend).



# Unit Test Suite



# Unit Test Suite

New Unit Test Suite

**Unit Test Suite**  
Create a new Unit Test Suite

Name of the Unit Test Suite:

Save Location:

Select an artifact file which want to test:

Test Artifacts	Location
<input type="checkbox"/> ContentBasedRouting/pr...	
<input checked="" type="checkbox"/> ArithmeticOperationS...	/ContentBasedRouting/src/main/synapse-config/proxy-services/...

New Unit Test Suite

**Supportive Artifacts**  
Select supportive artifacts which are used in the test artifact

Artifacts to include in the Unit Test Suite

Supportive Artifacts	Location
<input checked="" type="checkbox"/> ContentBasedRouting	
<input checked="" type="checkbox"/> NumberDivisionEP.xml	/ContentBasedRouting/src/main/synapse-config/endpoints/NumberDivision...
<input checked="" type="checkbox"/> NumberAdditionEP.xml	/ContentBasedRouting/src/main/synapse-config/endpoints/NumberAdditio...
<input type="checkbox"/> HelloWorld.xml	/HelloService/src/main/synapse-config/proxy-services/HelloWorld.xml
<input checked="" type="checkbox"/> ScatterGather	
<input type="checkbox"/> MissionEP2.xml	/ScatterGather/src/main/synapse-config/endpoints/MissionEP2.xml
<input type="checkbox"/> MissionEP3.xml	/ScatterGather/src/main/synapse-config/endpoints/MissionEP3.xml
<input type="checkbox"/> MissionService.xml	/ScatterGather/src/main/synapse-config/proxy-services/MissionService.xml
<input type="checkbox"/> MissionEP1.xml	/ScatterGather/src/main/synapse-config/endpoints/MissionEP1.xml



## Building a Unit Test Case

- Specify a unique name per test case.
- Input Payload: Should be XML, JSON or Plain Text.
- Input properties: Should be one of axis2, synapse, or transport-scope properties.
- For a sequence test case, you can specify any kind of property.
- For APIs or a proxy service test case, only transport-scope properties are accepted.



# Building a Unit Test Case

Add New Test Case

Add Input Payload, Properties and Assertion Details

Enter input and assertion details which you want to test

Test Case Name:

Operation\_Add

Input Payload and Properties

Input Payload:

<ArithmeticOperation>

<Operation>Add</Operation>

<Arg1>10</Arg1>

<Arg2>25</Arg2>

</ArithmeticOperation>

Input Properties:

Scope

Transport

Name

Content-Type

Value

text/xml

Assertions

Assertion Type	Assert Property
AssertEquals	\$body
AssertNotNull	\$body

Cancel

Update



## Building a Unit Test Case

- Request path:
  - Url mapping for the API resource.
  - If the path contains parameters, replace those with values.
- Request method: HTTP method of the resource.
- Assertions: You can match the results using assertions (AssertEquals or AssertNotNull).



## Building a Unit Test Case

**Unit Test - Assertions**  
Enter types of assertions which want to test

Assertion Type:

Actual Expression:

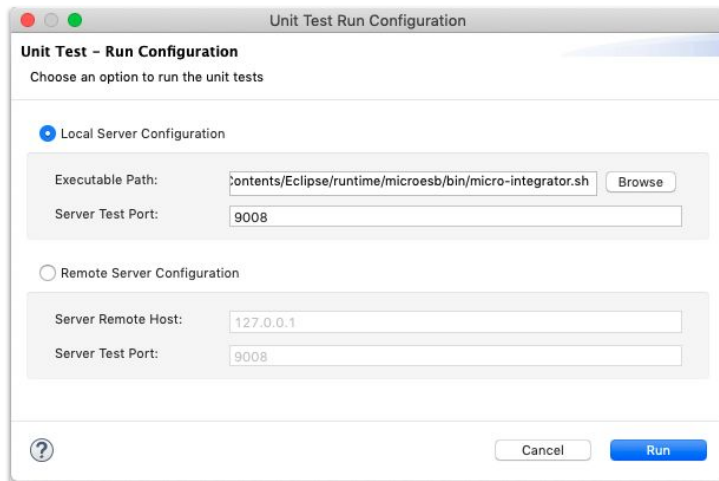
Expected Value: 

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:s="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
```

Error Message:

- Assertion Type:
  - **AssertEquals**
  - **AssertNotNull**
- Assertion Expression:
  - **\$body** : asserts the payload.
  - **\$ctx:""** : asserts synapse property.
  - **\$axis2:""** : asserts axis2. property
  - **\$trp:""** : asserts the transport property.
- Expected Value: The expected response.
- Error Message : To print upon assertion failure.

## Running Test Case



- **Local Server:**  
Run test suite using the embedded unit test server in the Micro Integrator.
- **Remote Server:**  
Run test suite using a remote unit test server in the Micro Integrator.

Be sure to start the Micro Integrator runtime with the **-DsynapseTest** parameter.

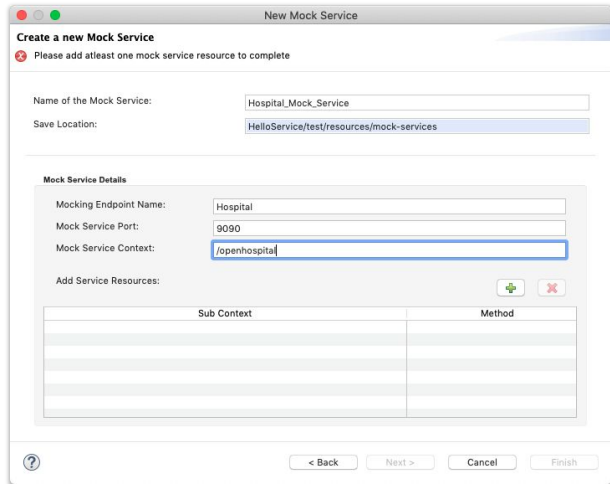
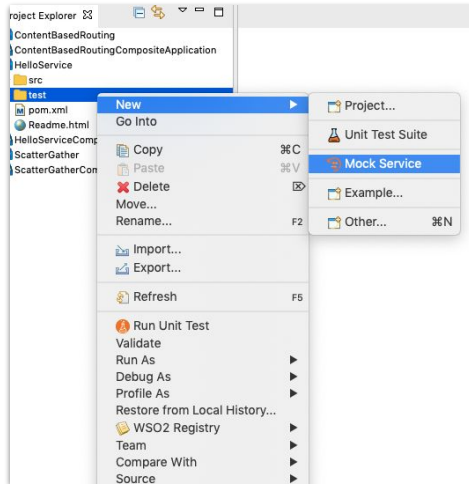


## Mock Services

- Mock services allow you to test your integration artifacts without using the actual back-end system.
- Useful when automating the tests as real back-end systems may not be accessible within the continuous integration system.



## Building a Mock Service



## Building a Mock Service

**Mock Service - Resource Details**  
Add sub resource with expected request and response data

Service Sub Context:

Service Method:

Expected Request to the Mock Service Resource

Note - Request must be matched with the entered request headers or payload to send the response

Expected Request Headers:

Header Name	Header Value
Content-Type	application/xml

Expected Request Payload:

< Back Next > Cancel Add

- Mock Service Context : Main url context for mock service. Starts with “/”. Example: /openhospital
- Service Sub Context: Sub url context under the main URL context for the resource.
- Service method: The HTTP method of the resource.
- Header Name: Expected header name.
- Header Value: Expected header value.
- Expected request payload: Expected request payload for the service.



## Building a Mock Service

**Add Resource for the Mock Service**

**Mock Service - Resource Details**  
Add sub resource with expected response data

Response Send Out from the Mock Service Resource

Response Status Code: 200 OK

Send out Response Headers:

Header Name	Header Value
Content-Type	application/json

Send out Response Payload:

```
[{"Doctor": "John"}]
```

< Back Next > Cancel Add

- Response Status Code: HTTP Status code for the response message.
- Header Name: Name for the given response header.
- Header Value: Value for the specified header.
- Send out response payload: Expected response payload.





**Let's try it out!**

**Creating a Unit Test Suite**

