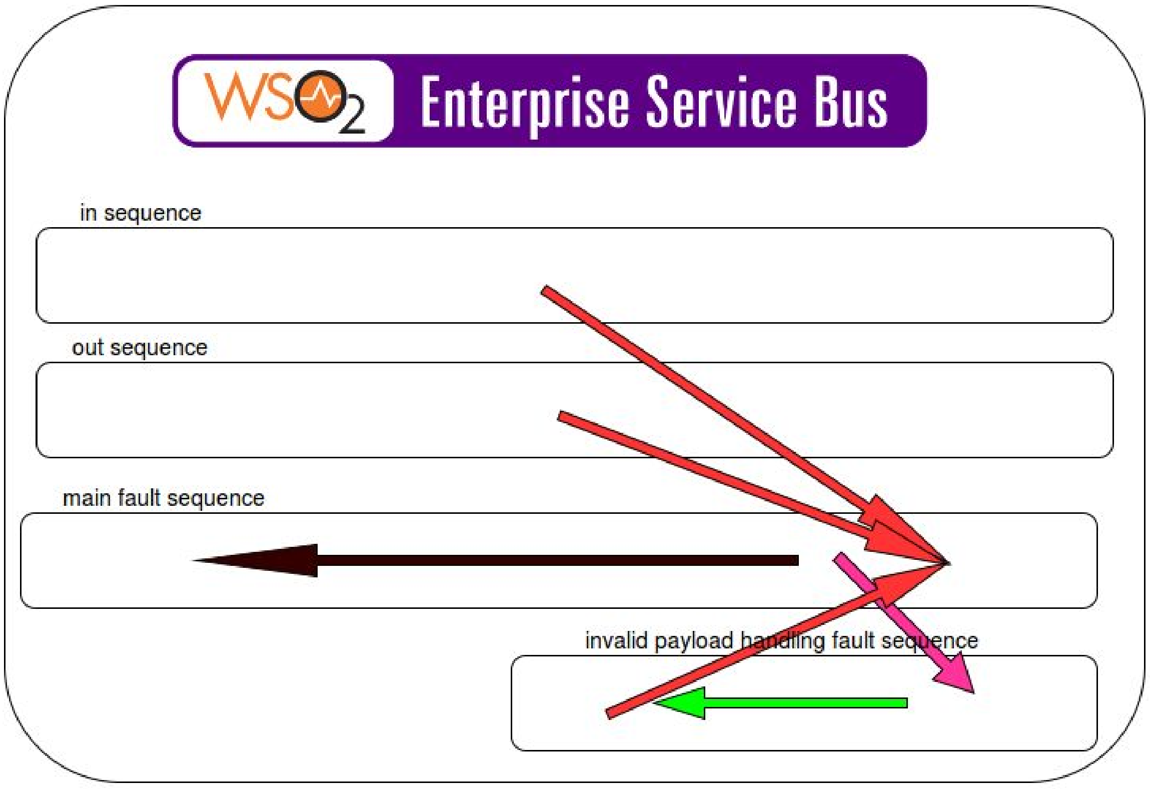
How to handle invalid JSON requests in WSO2 ESB

This document will take you on how to handle invalid JSON requests using a class mediator in wso2 esb. The main reason is that if the request is a invalid JSON when we try to build the message from the inputstream it will throw an exception and will forward to fault sequence. But again if there is a content aware mediator in the fault sequence it will try to build the invalid request again in the start of the sequence and it will fail. (This behaviour is not present in the current synapse. It has been undergone many improvements to handle this kind of scenarios)

So to handle above kind of scenario we could use a class mediator and design a flow that could handle it successfully.


WSO2 Enterprise Service Bus
in sequence
out sequence
main fault sequence
invalid payload handling fault sequence

As above diagram shows, when error occurs in the insequence or in the outsequence we would be forward to fault sequence. In this scenario if an **invalid JSON** request is sent to insequence and if we have a content aware mediator in it we would be forwarded to the fault sequence we defined as the “onError”/”faultsequence” in it.

But if we do processing these fault messages in the fault sequence it also will be a content aware sequence. Then it will try to build the message and will fail in this scenario. Lets define different error handling sequence as shown in the above diagram (invalid payload handling fault sequence) in the main fault sequence. So when main fault sequence fails it will be forward to this error handling sequence.

We will include our class mediator to remove this invalid JSON message and set a custom envelope saying Invalid JSON message has been received. Make sure you do not use any content aware mediator in this error sequence. Otherwise this sequence will also fails when trying to build it. So class mediator should override the isContentAware method to return false. We will change the message and forward it back to our main fault sequence so that we could handle all fault scenarios in there.

So following are the example sequences to implement above solution.

In sequence

```
<?xml version="1.0" encoding="UTF-8"?>
<sequence xmlns="http://ws.apache.org/ns/synapse"
          name="....insequencename..."
          onError="fault">
  <log level="full">
          <property name="MessageFlow"
          value="-------------------------------In sequence of the message flow---------------------------------"/>
  </log>
</sequence>
```

Main Fault sequence

```
<sequence xmlns="http://ws.apache.org/ns/synapse" name="fault" onError="faultSeq">
  <log level="custom">
    <property name="STATUS:" value="----------------fault invoked-----------------------" />
    <property name="STATUS" value="Executing default 'fault' sequence" />
    <property name="ERROR_CODE" expression="get-property('ERROR_CODE')" />
    <property name="ERROR_MESSAGE" expression="get-property('ERROR_MESSAGE')" />
  </log>
  <switch source="//fault/type........">
    <case regex="invalid......">................</case>
    <default>..............</default>
  </switch>
  <class name="org.wso2.carbon.apimgt.usage.publisher.APIMgtFaultHandler" />
  <property name="RESPONSE" value="true" />
  <header name="To" action="remove" />
  <property name="NO_ENTITY_BODY" scope="axis2" action="remove" />
   ……………………………………………………………………………….
  <property name="X-JWT-Assertion" scope="transport" action="remove" />
  <sequence key="_cors_request_handler_" />
<send />
</sequence>
```

Invalid Payload handling error sequence

```
<?xml version="1.0" encoding="UTF-8"?>
<sequence xmlns="http://ws.apache.org/ns/synapse" name="faultSeq">
<class name="org.wso2.carbon.samples.FaultErrorMediator"></class>
<property description="" name="ContentType" scope="axis2" type="STRING" value="application/xml"/>
  <sequence key="fault"/>
</sequence>
```

Class Mediator Implementation

And now what we need is the class mediator to remove this invalid payload and construct a new message with fault message to be handle in the main fault sequence.

Following is the sample class mediator class for this.

```
package org.wso2.carbon.samples;

import org.apache.axiom.om.OMAbstractFactory;
import org.apache.axiom.om.OMElement;
import org.apache.axiom.om.OMNamespace;
import org.apache.axiom.soap.SOAPEnvelope;
import org.apache.axis2.AxisFault;
import org.apache.synapse.MessageContext;
import org.apache.synapse.mediators.AbstractMediator;

import org.apache.axiom.soap.SOAPFactory;

public class  FaultErrorMediator extends AbstractMediator {

    @Override
    public boolean mediate(MessageContext mc) {
        //Construct the SOAP envelope
        SOAPFactory factory = OMAbstractFactory.getSOAP11Factory();
        SOAPEnvelope envelope = factory.getDefaultEnvelope();
        OMNamespace ns = factory.createOMNamespace("http://sample.namespace", "");
        OMElement faultContent = factory.createOMElement("Fault", ns);
        OMElement response = factory.createOMElement("response",ns);
        OMElement fault = factory.createOMElement("fault", ns);
        OMElement msg = factory.createOMElement("message",ns);
        OMElement status = factory.createOMElement("status",ns);

        msg.addChild(factory.createOMText("JSON Parsing Error"));
        status.addChild(factory.createOMText("01"));

        fault.addChild(msg);
        fault.addChild(status);

        response.addChild(fault);
        faultContent.addChild(response);
        envelope.getBody().addChild(faultContent);
        try {
              //Update the messageContext with new SOAP envelope
              mc.setEnvelope(envelope);
        } catch (AxisFault axisFault) {
              axisFault.printStackTrace();
        }
        return true;
    }

    @Override
        public boolean isContentAware() {
              return false;
        }
}
```

Build the class mediator and put the jar into [carbon-home]/repository/components/lib folder.

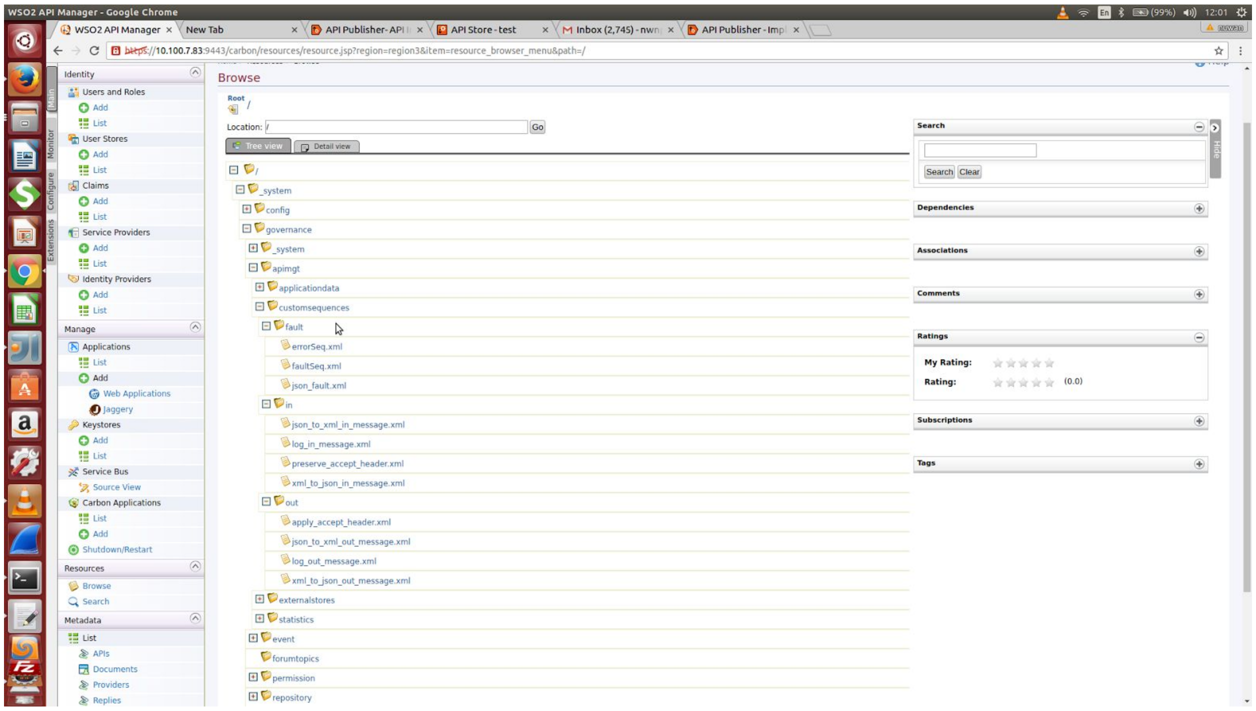
So this will construct a fault message. Set content type for xml and send it back to fault sequence.

We should be now able to handle the fault handling scenario with this configuration with the class mediator.

How To Handle Invalid JSON Fault Scenario in APIM

In the apim also the concept is the same. It is the same flow we should implement. To define the in sequence and fault sequence respect to our api following configurations be done.

First upload the created sequences in the apim registry.

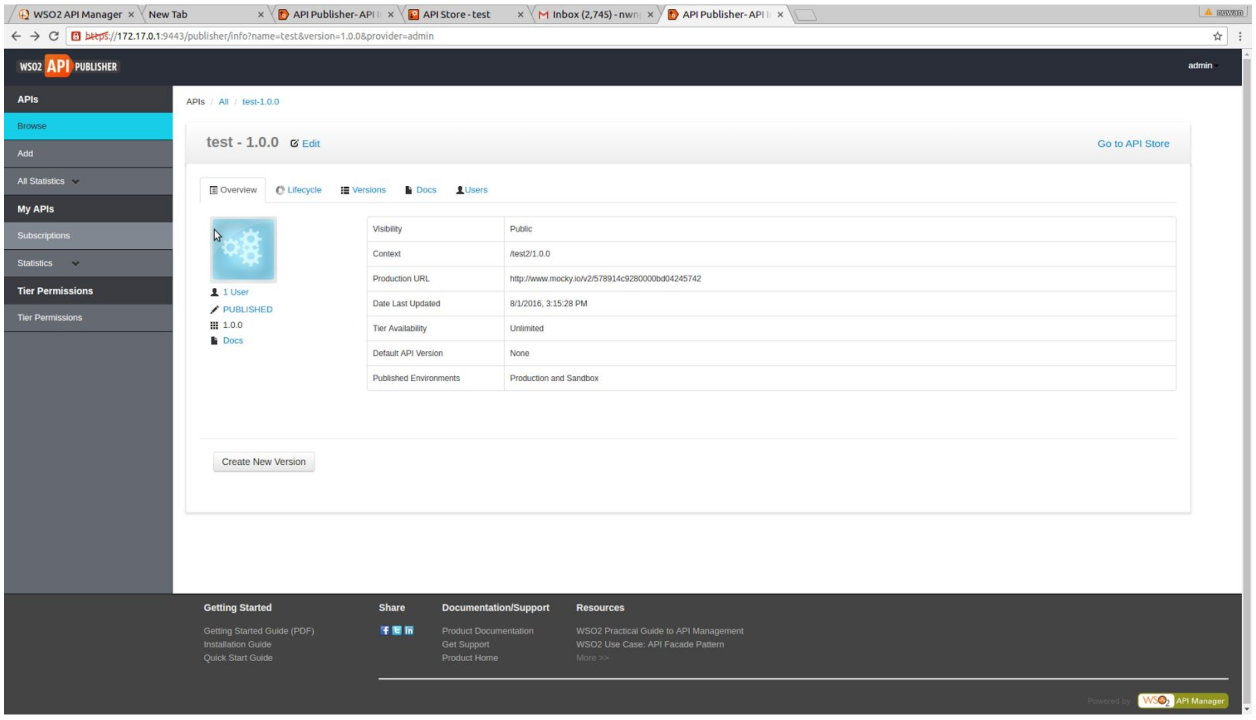


Browse

Location: /

```
/
  _system
    config
    governance
      _system
      apimgt
        applicationdata
        customsequences
          fault
            errorSeq.xml
            faultSeq.xml
            json_fault.xml
          in
            json_to_xml_in_message.xml
            log_in_message.xml
            preserve_accept_header.xml
            xml_to_json_in_message.xml
          out
            apply_accept_header.xml
            json_to_xml_out_message.xml
            log_out_message.xml
            xml_to_json_out_message.xml
        externalstores
        statistics
      event
      forumtopics
      permission
      repository
```

Then after that go to apim publisher and the created api.

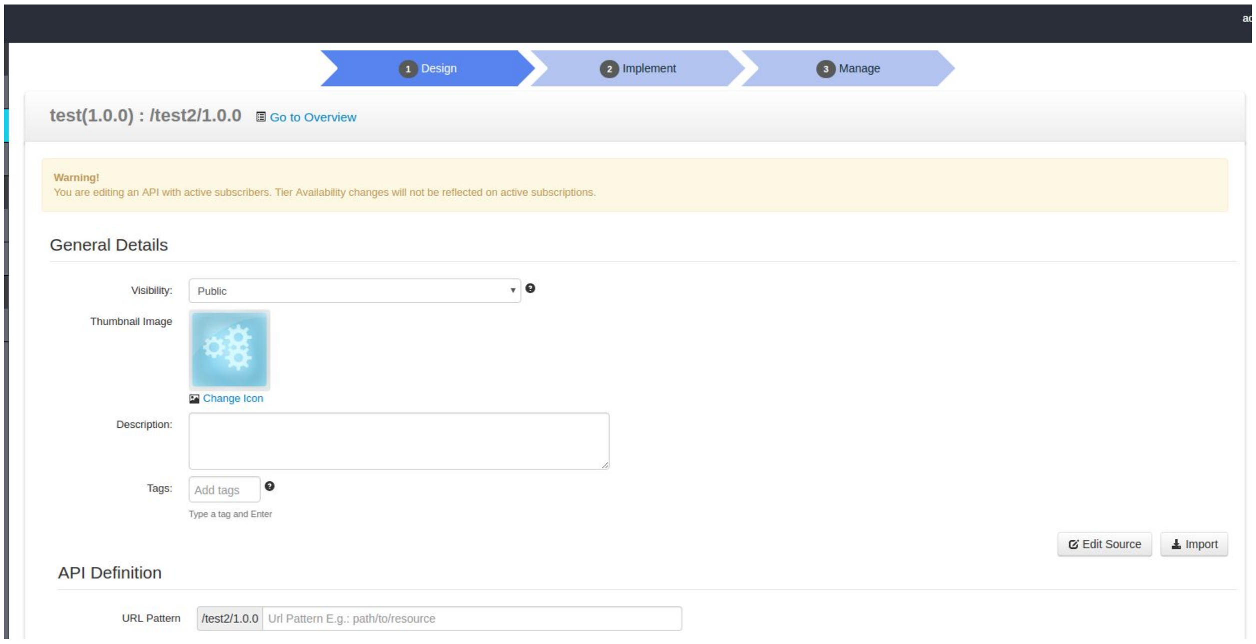


APIs / All / test-1.0.0

test - 1.0.0

Visibility	Public
Context	/test2/1.0.0
Production URL	http://www.mocky.io/v2/578914c9280000bd04245742
Date Last Updated	8/1/2016, 3:15:28 PM
Tier Availability	Unlimited
Default API Version	None
Published Environments	Production and Sandbox

Click edit and go to configuration area



1 Design | 2 Implement | 3 Manage

test(1.0.0) : /test2/1.0.0 Go to Overview

Warning!
You are editing an API with active subscribers. Tier Availability changes will not be reflected on active subscriptions.

General Details

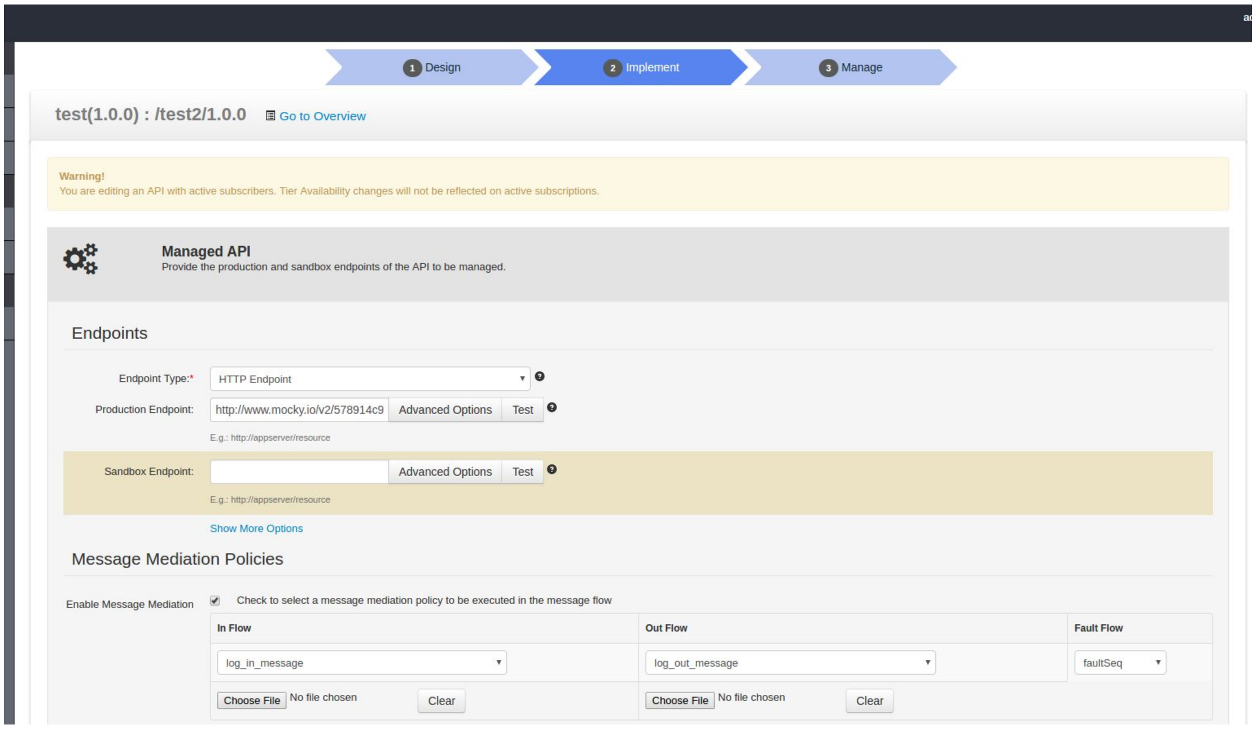
Visibility:	Public
Thumbnail Image	Change Icon
Description:	
Tags:	Add tags (Type a tag and Enter)

Edit Source | Import

API Definition

URL Pattern: /test2/1.0.0 Url Pattern E.g.: path/to/resource

Go to implement section



1 Design | 2 Implement | 3 Manage

test(1.0.0) : /test2/1.0.0 Go to Overview

Warning!
You are editing an API with active subscribers. Tier Availability changes will not be reflected on active subscriptions.

Managed API
Provide the production and sandbox endpoints of the API to be managed.

Endpoints

Endpoint Type:*	HTTP Endpoint
Production Endpoint:	http://www.mocky.io/v2/578914c9 (E.g.: http://appserver/resource)
Sandbox Endpoint:	(E.g.: http://appserver/resource)

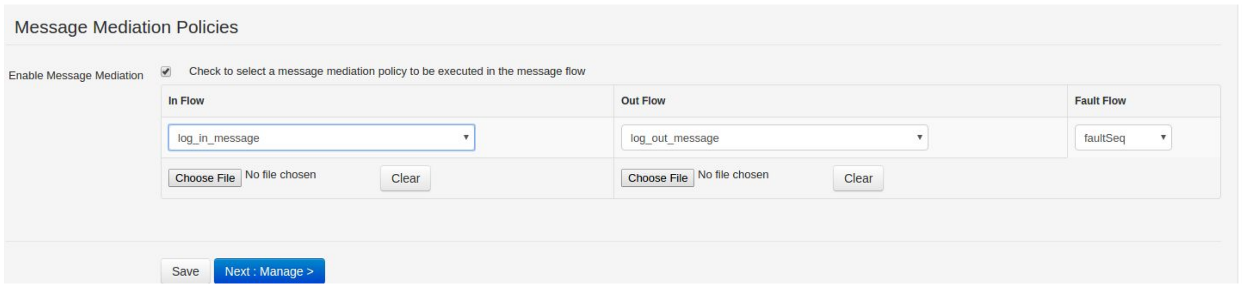
Show More Options

Message Mediation Policies

Enable Message Mediation: ✓ Check to select a message mediation policy to be executed in the message flow

In Flow	Out Flow	Fault Flow
log_in_message	log_out_message	faultSeq
Choose File No file chosen Clear	Choose File No file chosen Clear	

You will get the Message mediation policies in here and if you have uploaded the created sequences to registry as mention before they will be listed in here. So select the respective sequences (main fault sequence as the Fault Flow , in sequence with the on error pointed to main fault sequence) in In/Out and Fault flows.


Message Mediation Policies

Enable Message Mediation ✓ Check to select a message mediation policy to be executed in the message flow

In Flow	Out Flow	Fault Flow
log_in_message	log_out_message	faultSeq
Choose File No file chosen Clear	Choose File No file chosen Clear	

Version of the document : 1.1.0
Authors : nuwanp@wso2.com