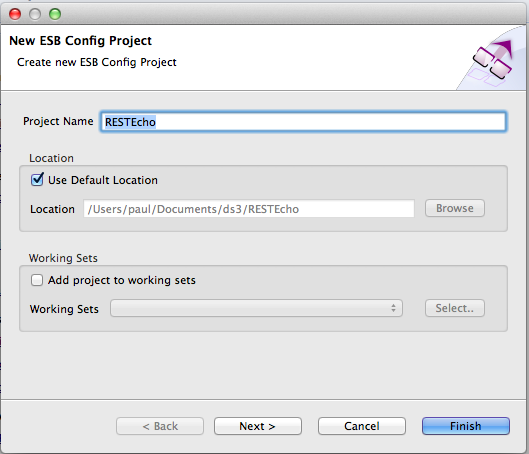
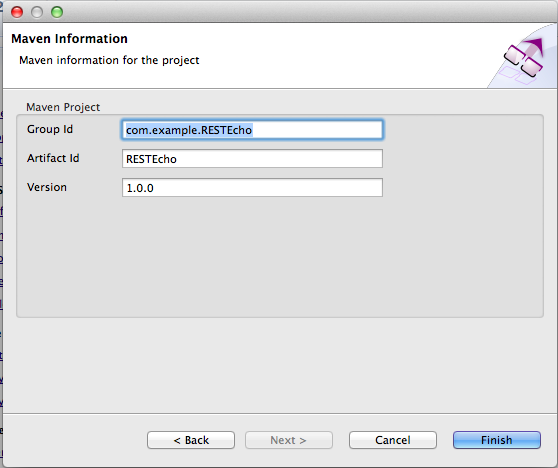
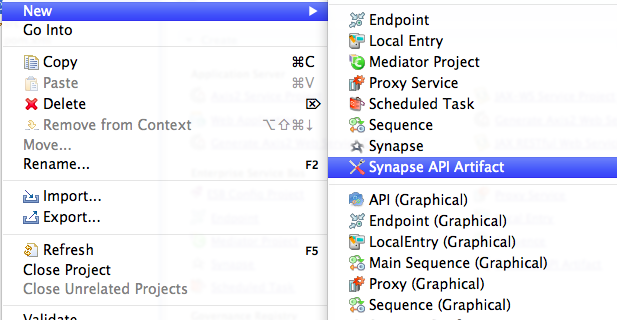
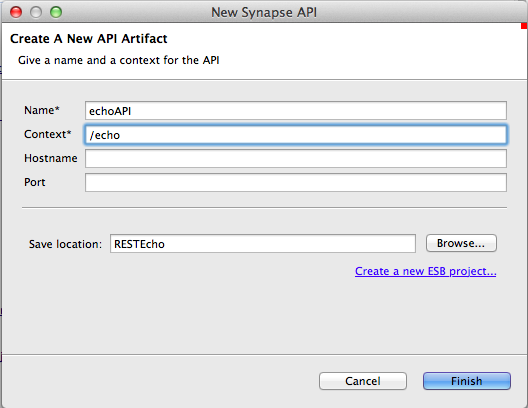
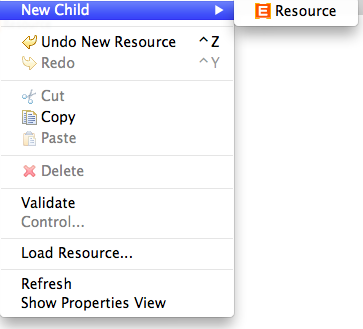
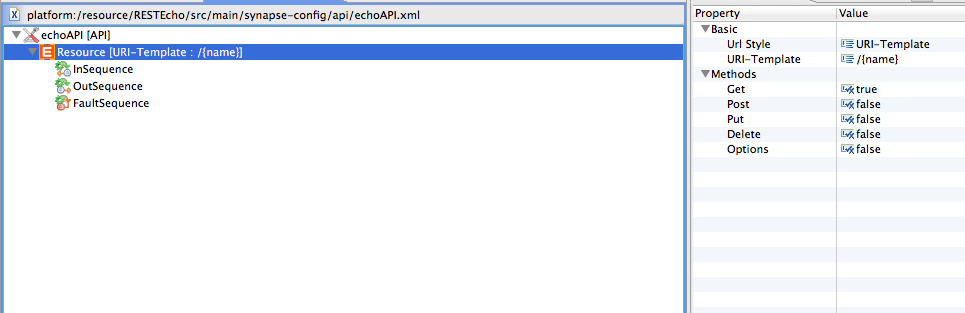
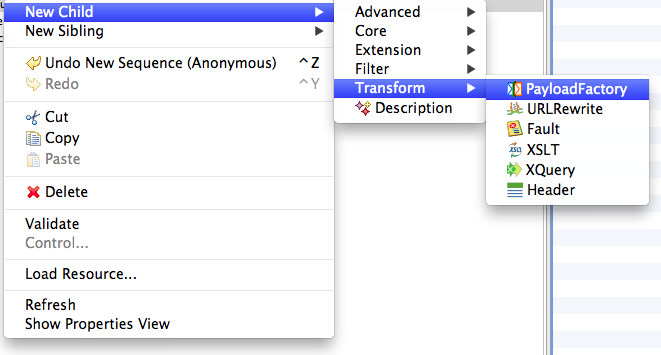
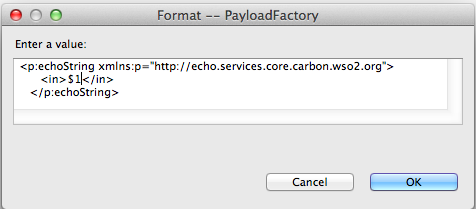
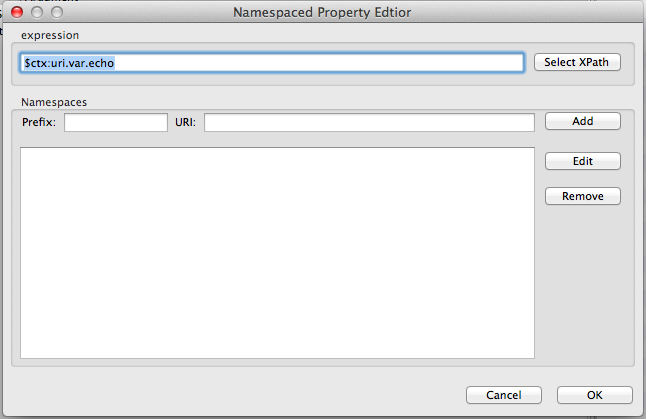
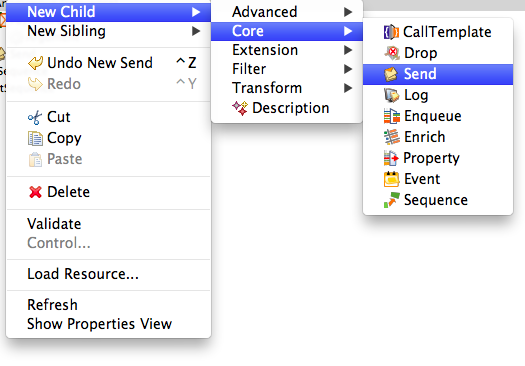
1. Goto the Developer Studio 2.1.0 Dashboard
2. Click **ESB Config Project**
3. Use the name **RESTEcho**
4. Click **Next**
5. Leave the Maven info the same:  
   
6. Click Finish
7. Right Click on the project and **New->Synapse API Artifact** ****
8. Use:  
     
   Name: **echoAPI**Context: **/echo**  
   
9. **Finish**
10. Right click on the echoAPI item.
11. Choose New Child->Resource: 
12. Select the Resource and on the right hand side edit the properties to:  
      
    URL Style: URI Template  
    URI-Template: /{name}  
    Get: true  
      
    Note the template option only appears after you select the style.
13. Expand the little arrow next to the Resource: 
14. Right click on InSequence and New Child-> Sequence (Anonymous)
15. Right click on the anonymous sequence and add a   
    New Child->Transform->PayloadFactory:  
    
16. Click on the PayloadFactory mediator and using the properties pane on the right hand side, edit the “Format” replacing <inline/> with the XML below.  
    
17. Add a new **Argument** child to the PayloadFactory. Use the following properties:  
    Argument Type: Expression  
    Argument Expression: $ctx:uri.var.name  
    This will grab the {name} property from the incoming URL and map it to argument $1, which will be embedded in the XML.
18. Now add a new Send mediator below the PayloadFactory. You can do this by creating a new child on the sequence.
19. On a Send mediator create a new Child -> Address Endpoint
20. Change the properties on the Address Endpoint:  
    Address: http://127.0.0.1:9763/services/echo  
    Message Format: SOAP1.1
21. You have now created an ESB API that will:

* Listen at /echo/{name}
* Extract the {name} value
* Construct an XML message
  + Using the name parameter
* Send this as SOAP11 to our server endpoint
* Send the response back to the client

1. Before deploying this in the ESB, take a look at the XML configuration behind this configuration. Click on the Source tab (bottom left corner of the API design pane). Your XML should look like this:

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

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

<inSequence>

<payloadFactory>

<format>

<p:echoString xmlns:p="http://echo.services.core.carbon.wso2.org">

<in xmlns="http://ws.apache.org/ns/synapse">$1</in>

</p:echoString>

</format>

<args>

<arg expression="$ctx:uri.var.name"/>

</args>

</payloadFactory>

<send>

<endpoint>

<address encoding="UTF-8" format="soap11"

statistics="disable" trace="disable" uri="http://127.0.0.1:9763/services/echo">

<timeout>

<duration>0</duration>

<responseAction>discard</responseAction>

</timeout>

<markForSuspension>

<retriesBeforeSuspension>0</retriesBeforeSuspension>

<retryDelay>0</retryDelay>

</markForSuspension>

<suspendOnFailure>

<initialDuration>0</initialDuration>

<maximumDuration>0</maximumDuration>

<progressionFactor>1.0</progressionFactor>

</suspendOnFailure>

</address>

</endpoint>

</send>

</inSequence>

</resource>

</api>

1. Let’s briefly examine this:  
     
   Firstly, we are defining an API which is a collection of resource definitions (in the REST style). Each resource is actually implemented by a sequence of flow logic. In this case, we are looking for a GET and mapping it to a simple flow with two mediators. First we create an XML payload, then we send that to an endpoint. The endpoint definition is quite long because the editor has filled in plenty of config for what to do in failure cases.
2. Now we need to deploy this into the ESB. Unfortunately there is a bit of a