**Automation Project Created by:** Stephen Costales

**Documentation Author:** Stephen Costales

**Summary:** The purpose of this document is to provide a step by step process and solution for test automation. Please note that further information can be given by the QA Engineer/Software Developer to Product Management (Business) on how to automate web services in general (REST, SOAP, etc.) using methods (GET, POST, PUT, DELETE).

However, for this example the following will be performed:

1. Call a REST service api using GET method
2. Create/Add Assertions for validating request-response

**Note**: This can be done using UI or custom scripts (QA Engineer can demonstrate both but this example shows UI) *see “additional notes” for further information*

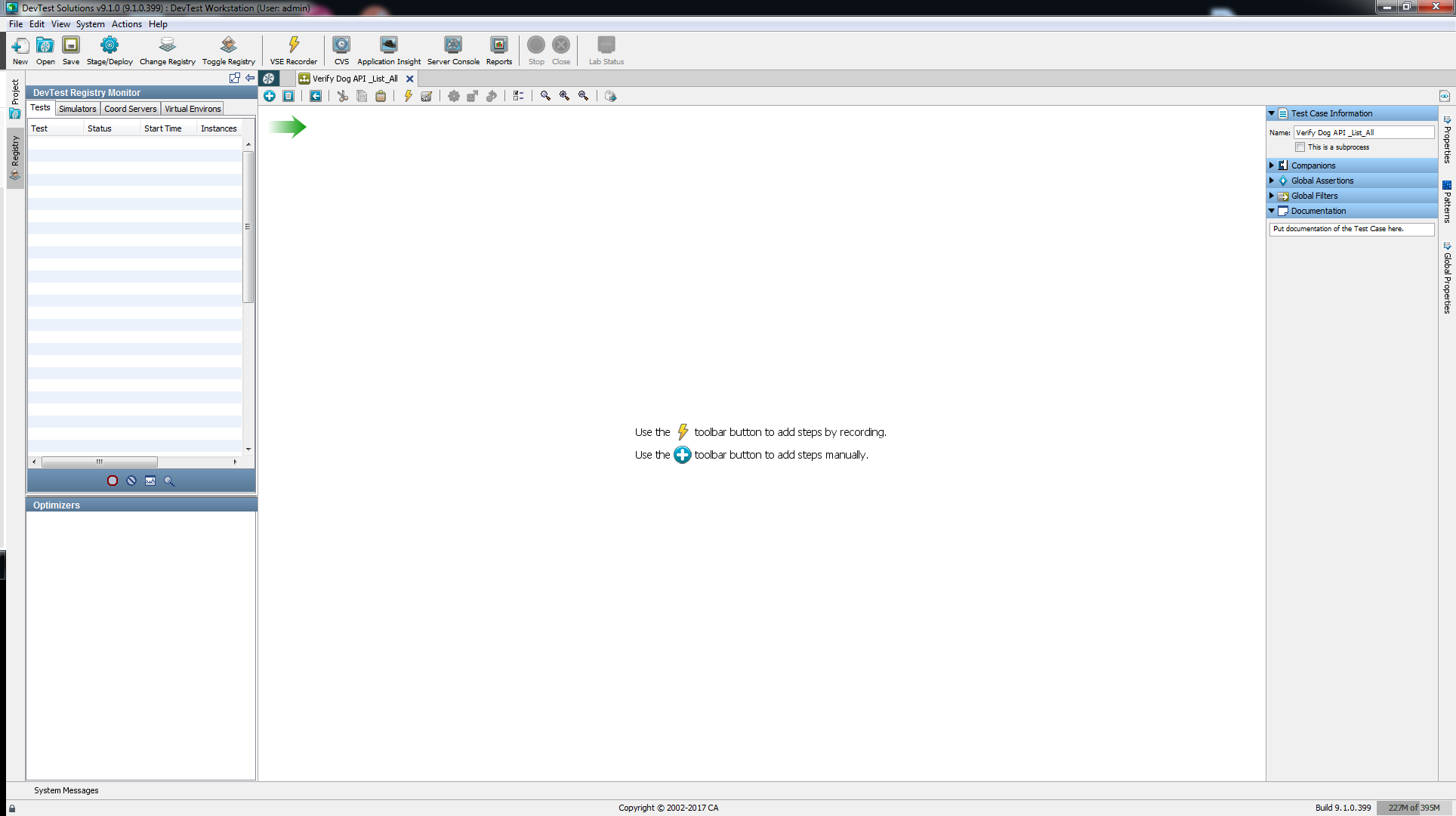
1. Create filters to parse through the response and capture desired data
2. Create a delimited file for testers to add different parameters and/or headers to loop through and make variety of calls to the service for both positive and negative test cases
3. Output log message to verify to the user that the service is behaving as expected

**Pre-Conditions:** User must have Lisa-DevTest Workstationinstalled. For Instructions on installing DevTest Workstation please refer to **Automation\_Required\_Installations.doc**

*File located under* QAEngineer\_SampleCode\_Automation\_w\_LISA\validate\_dog-ceo-api

**Procedure:** Refer to **lisa.project** located 🡪 QAEngineer\_SampleCode\_Automation\_w\_LISA\validate\_dog-ceo-api\QA\_Engineer\_Automation\lisa\AutomationScripts shows how to automate a REST Service

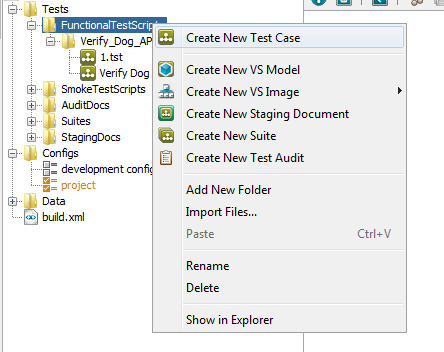
1. Open DevTest Workstation and Create New Project
2. This image shows a new blank project



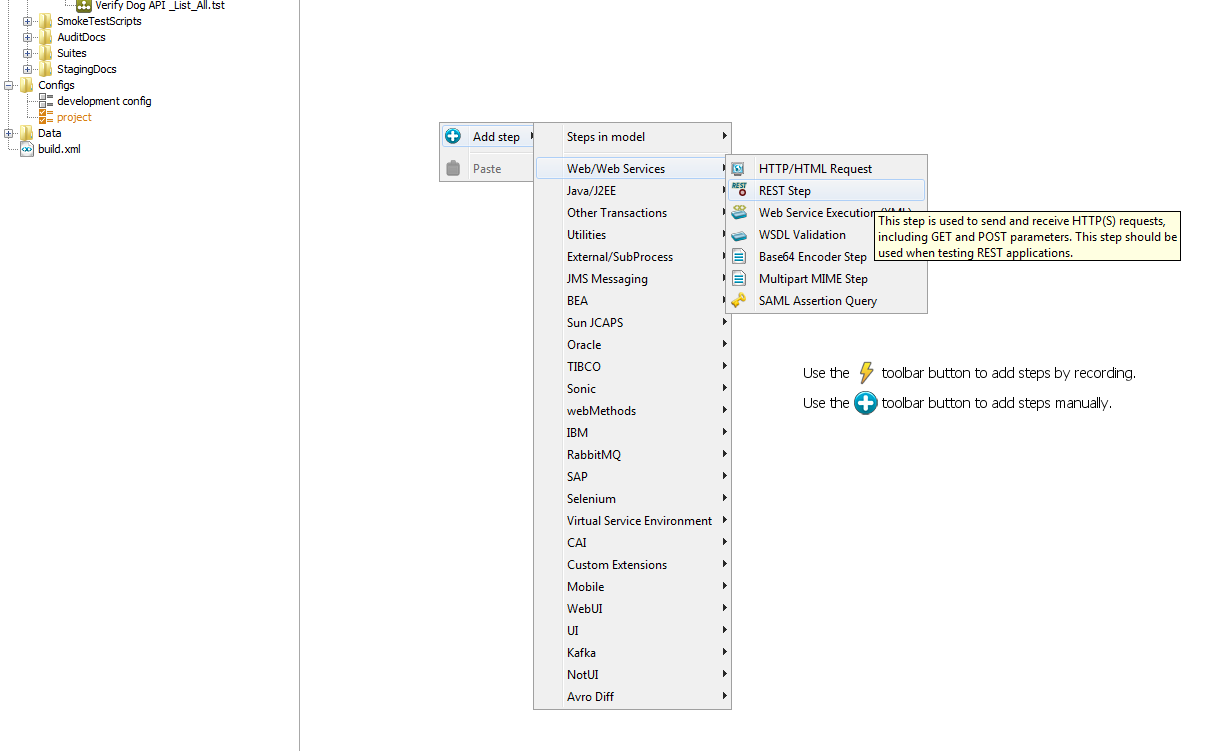
1. We begin by creating a test case and adding a REST step to call the Service API. In this case we are using <https://dog.ceo/api/breeds/list/all> (sample code to build this is also included in repo)

Note: I named the test case Verify Dog API\_List\_All

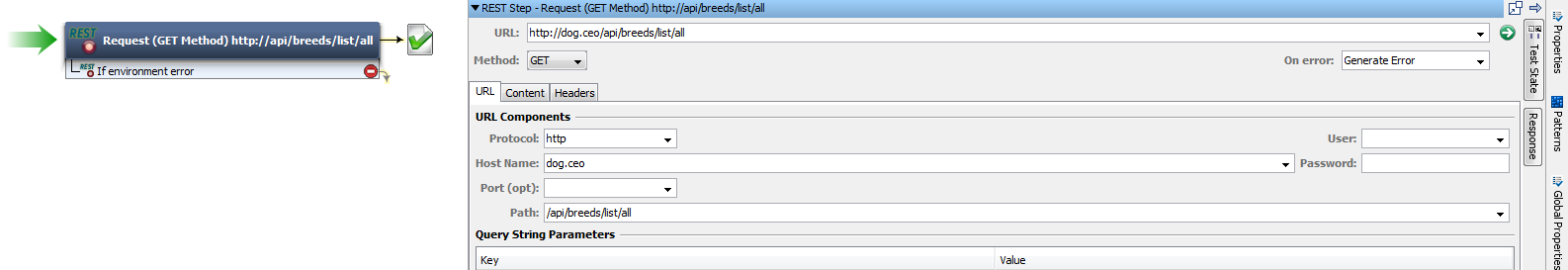
1. A.



B.

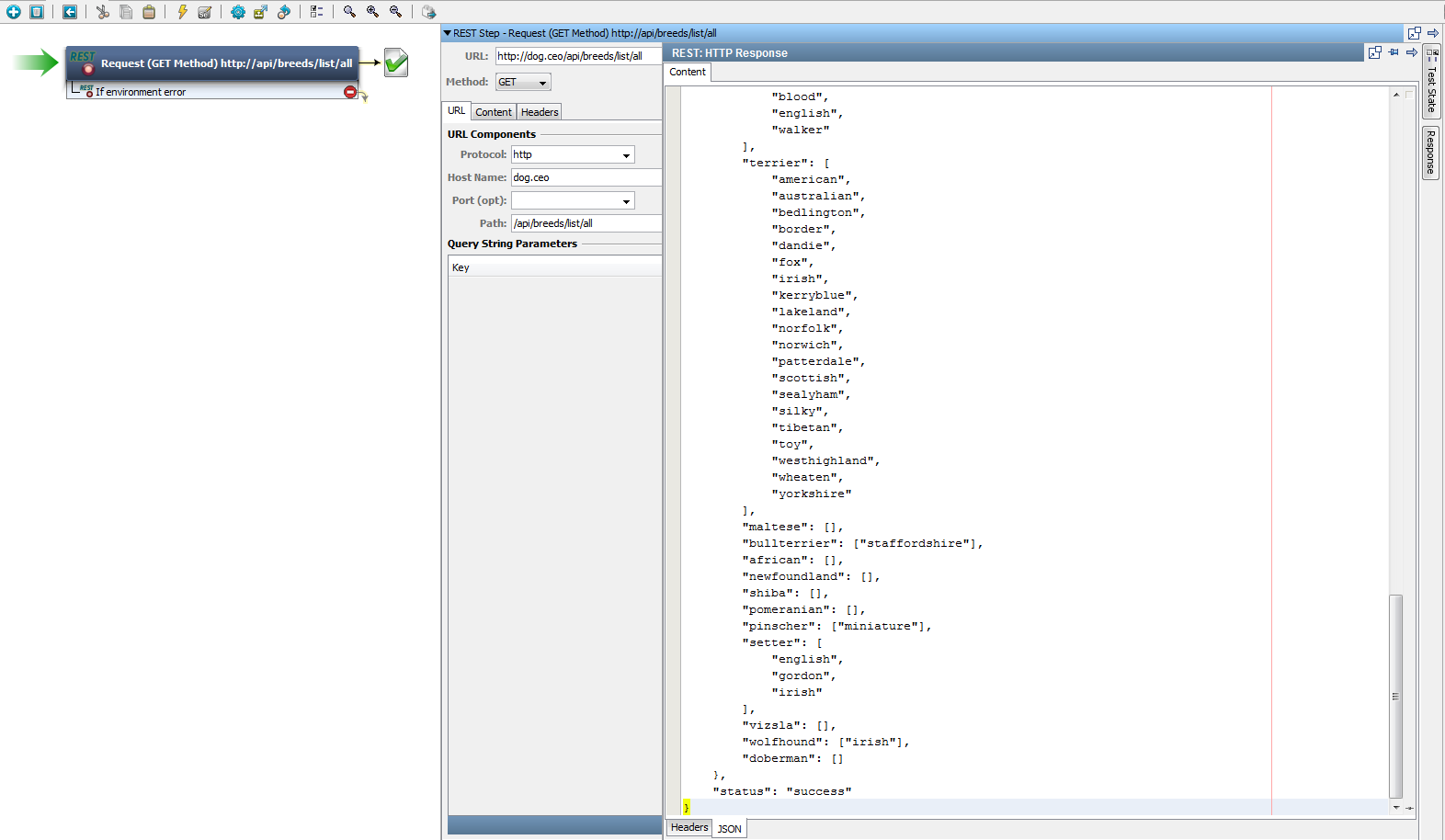


1. Add URL (with body parameters and request headers as necessary) to the REST Step and Click the Play button (color green) on the right to execute the call



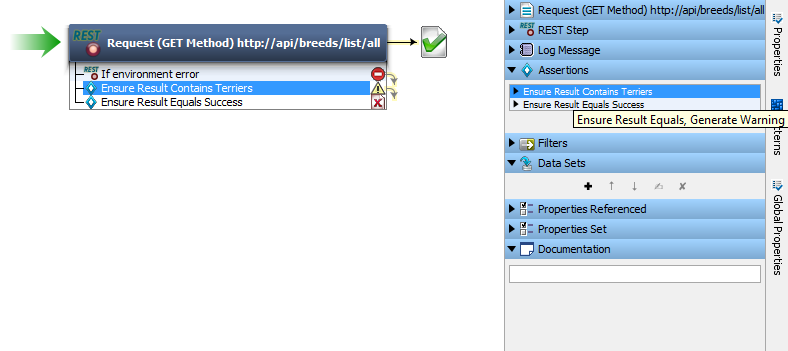
Then click the Response Tab on the tool bar to the right. The Result should look like

(iii)

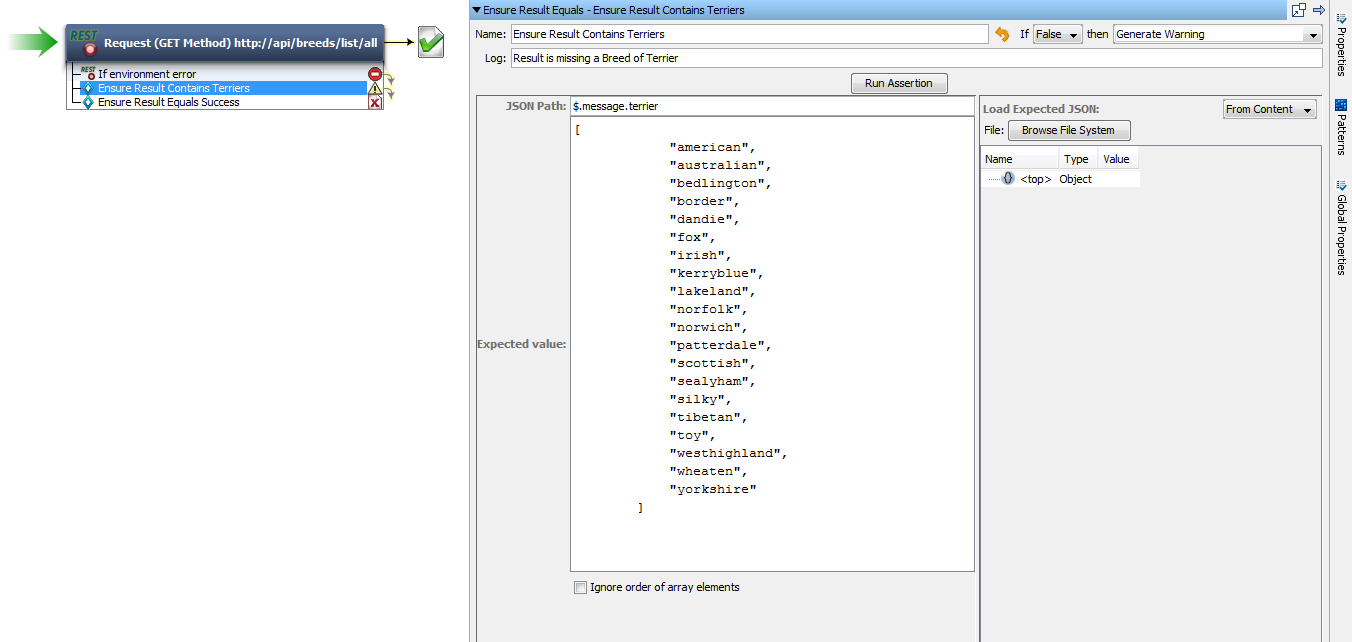


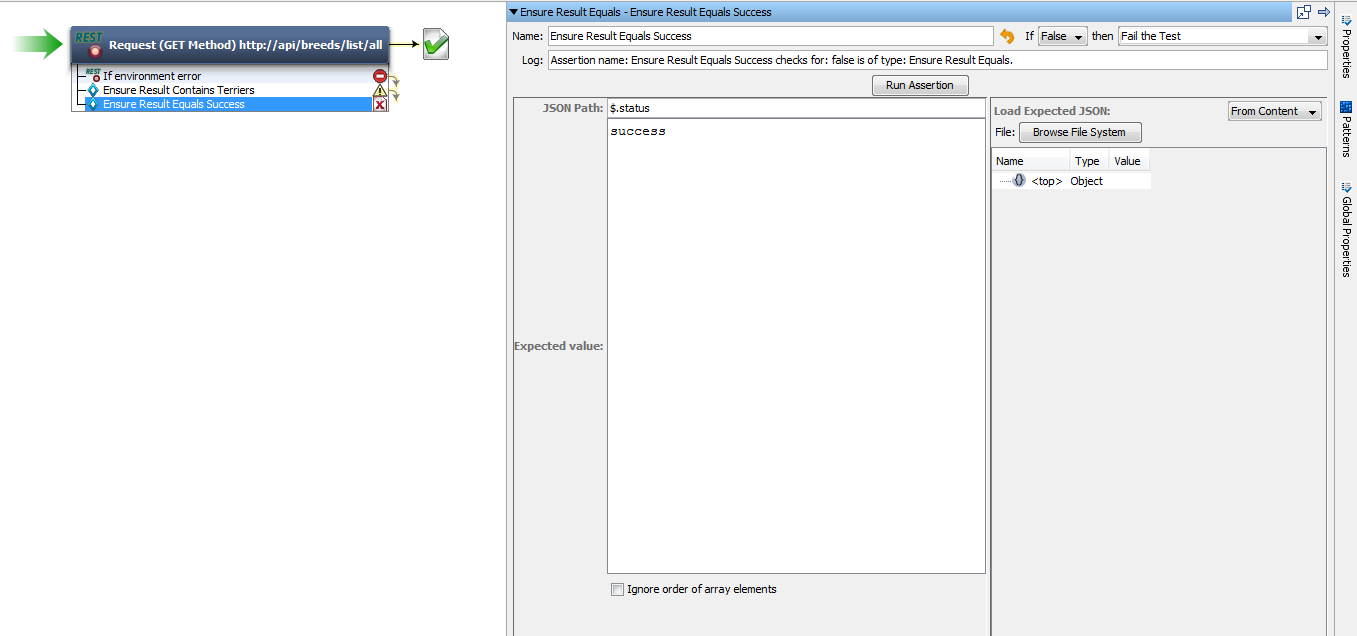
1. Now we are ready to add assertions. In this example, I want to verify that the JSON response contains “status”: “success” and all the values for “terrier” as per JSONPath Syntax $.message.terrier

(iv)



(v)



(vi)

1. We can now create filters. Of the many options, I used the utility filter “Store Step Response” as well as “parse HTML Result HTTP Header value” for all of the following:

**Accept-Ranges →**bytes

**Access-Control-Allow-Origin →**\*

**Age →**0

**Cache-Control →**no-cache, private

**Connection →**keep-alive

**Content-Length →**1755

**Content-Type →**application/json

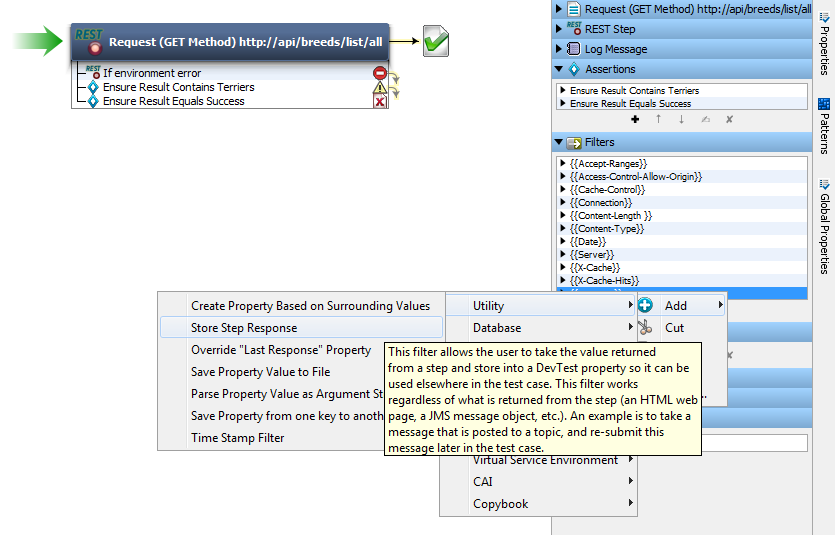
**Date →**Tue, 31 Oct 2017 20:30:36 GMT

**Server →**Apache/2.4.26

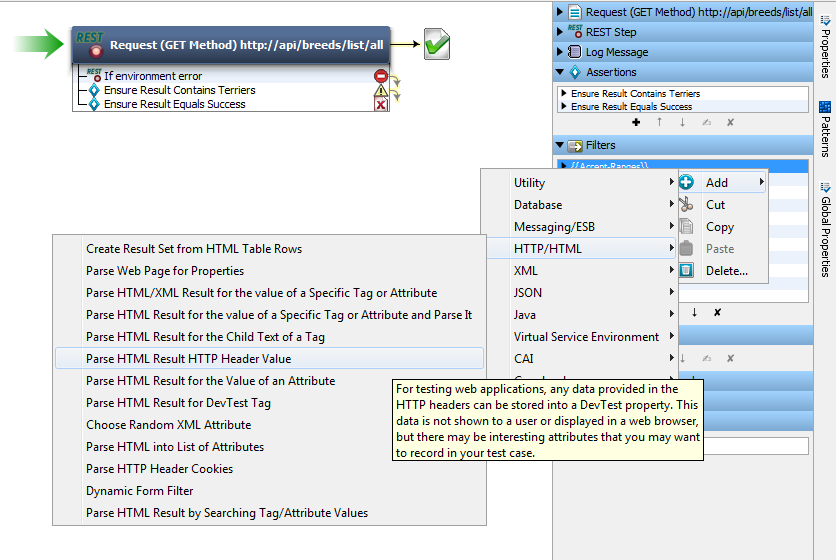
**Via →**1.1 varnish-v4, 1.1 varnish-v4

**X-Cache →**MISS

**X-Cache-Hits →**0

(vii)

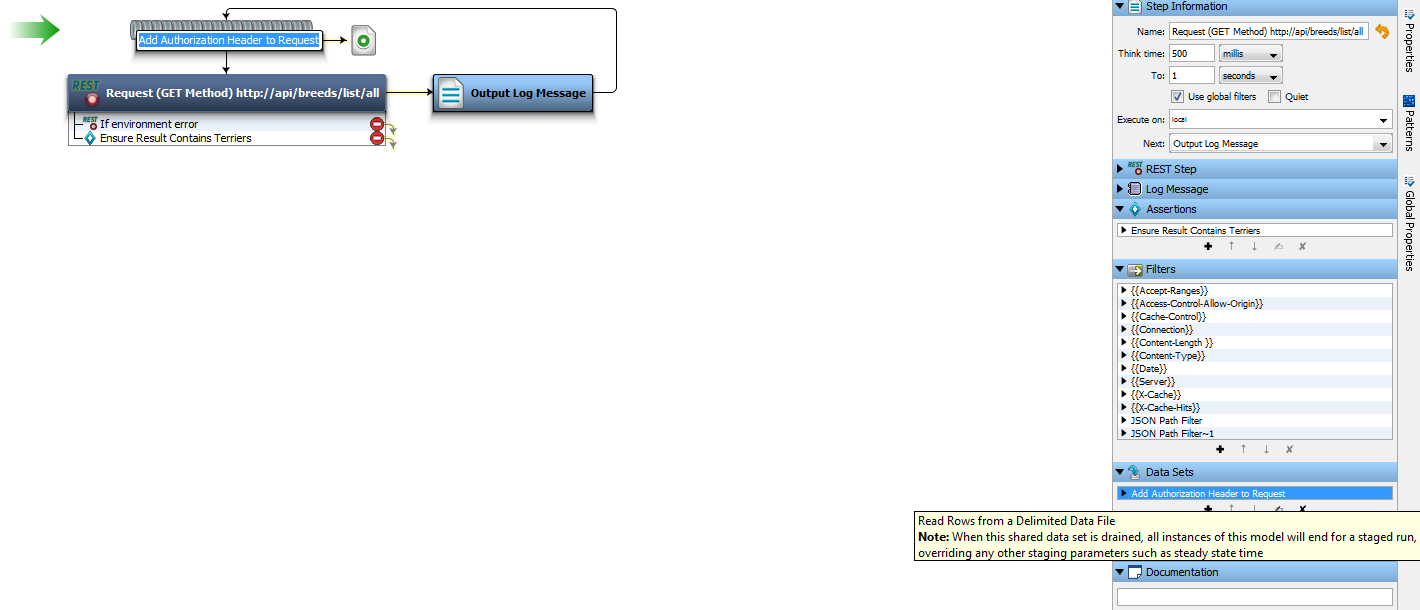
(viii)

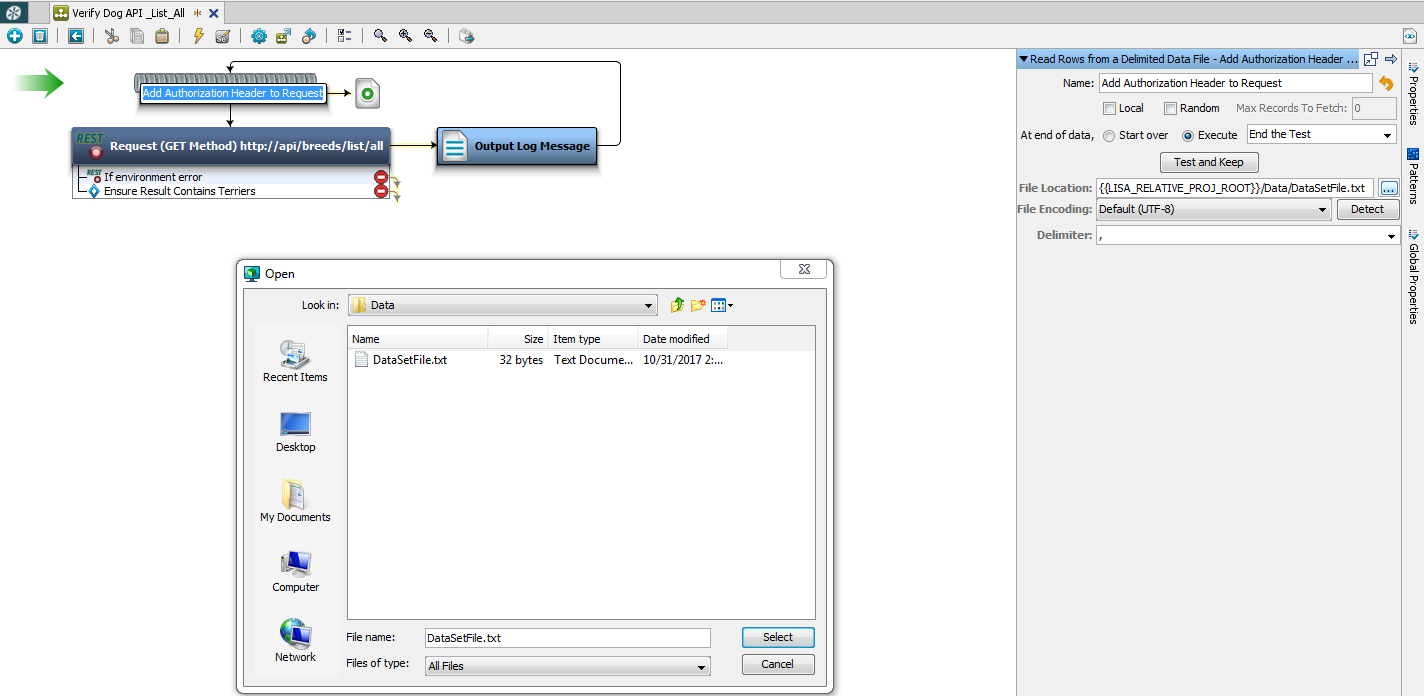


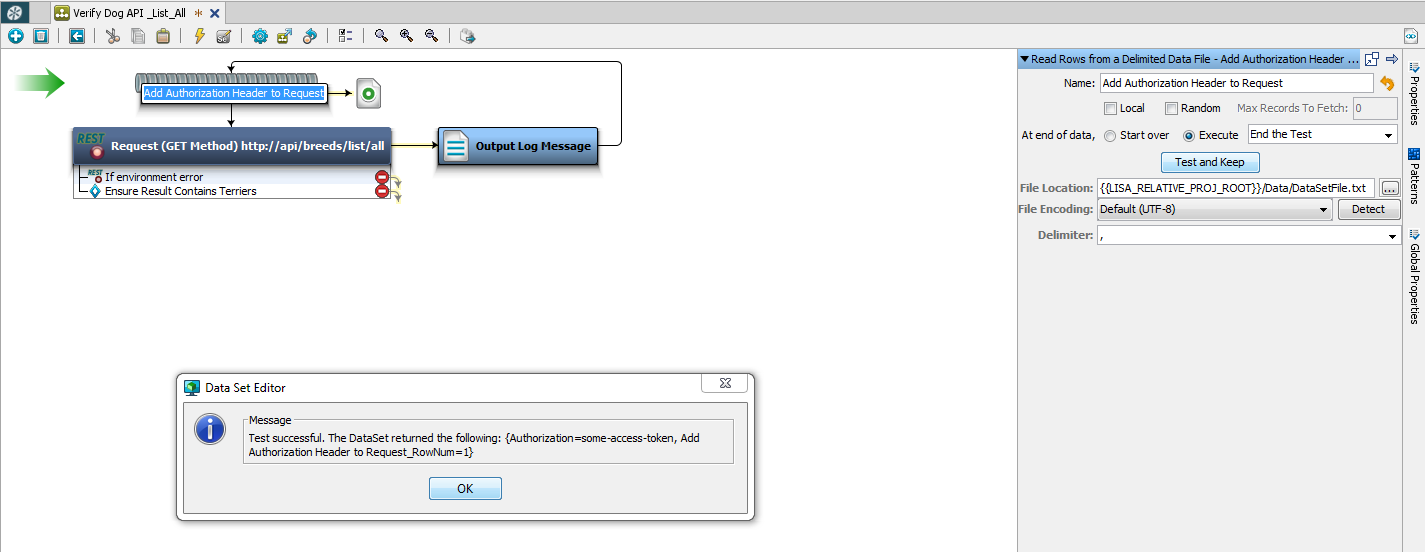
1. Now we can create a delimited file to loop through as well as add Output log messaging which will utilize all the filters we created in Step 5.

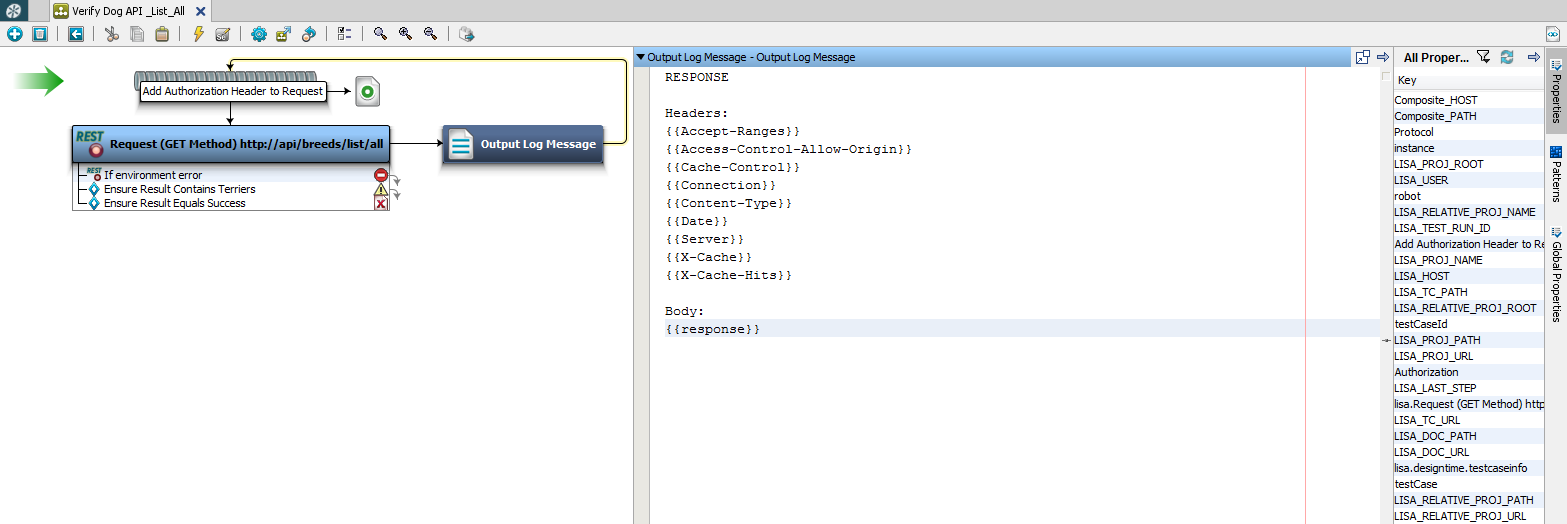
For simplicity and to demonstrate the entire flow, the following images will depict the remaining steps before running the test case. Note for the delimited file (.txt) the user can input anything they desire in order to the REST service with various requests. In this case I added “Authorization:some-access-token”

(ix)



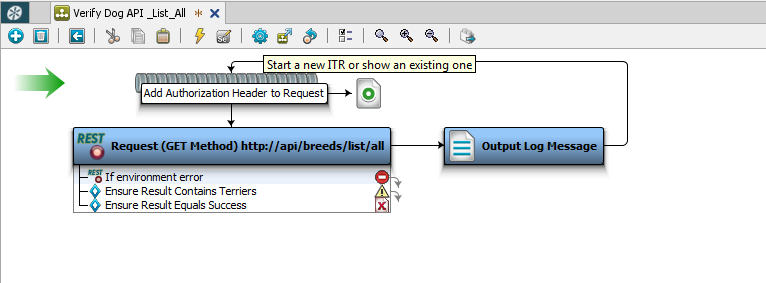


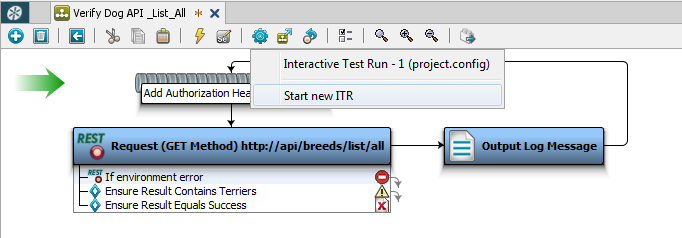




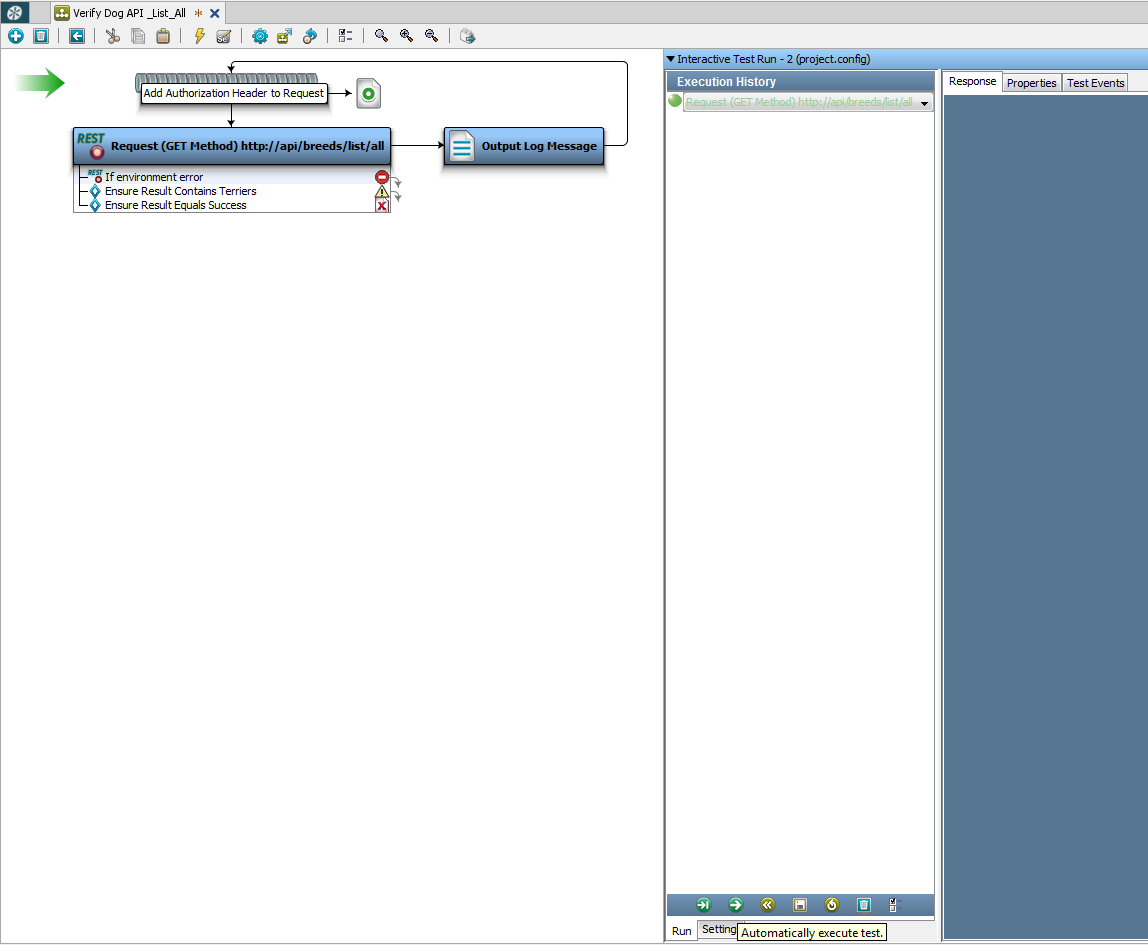
1. Now we are ready to run the test case. See the following images for Reference

(x)



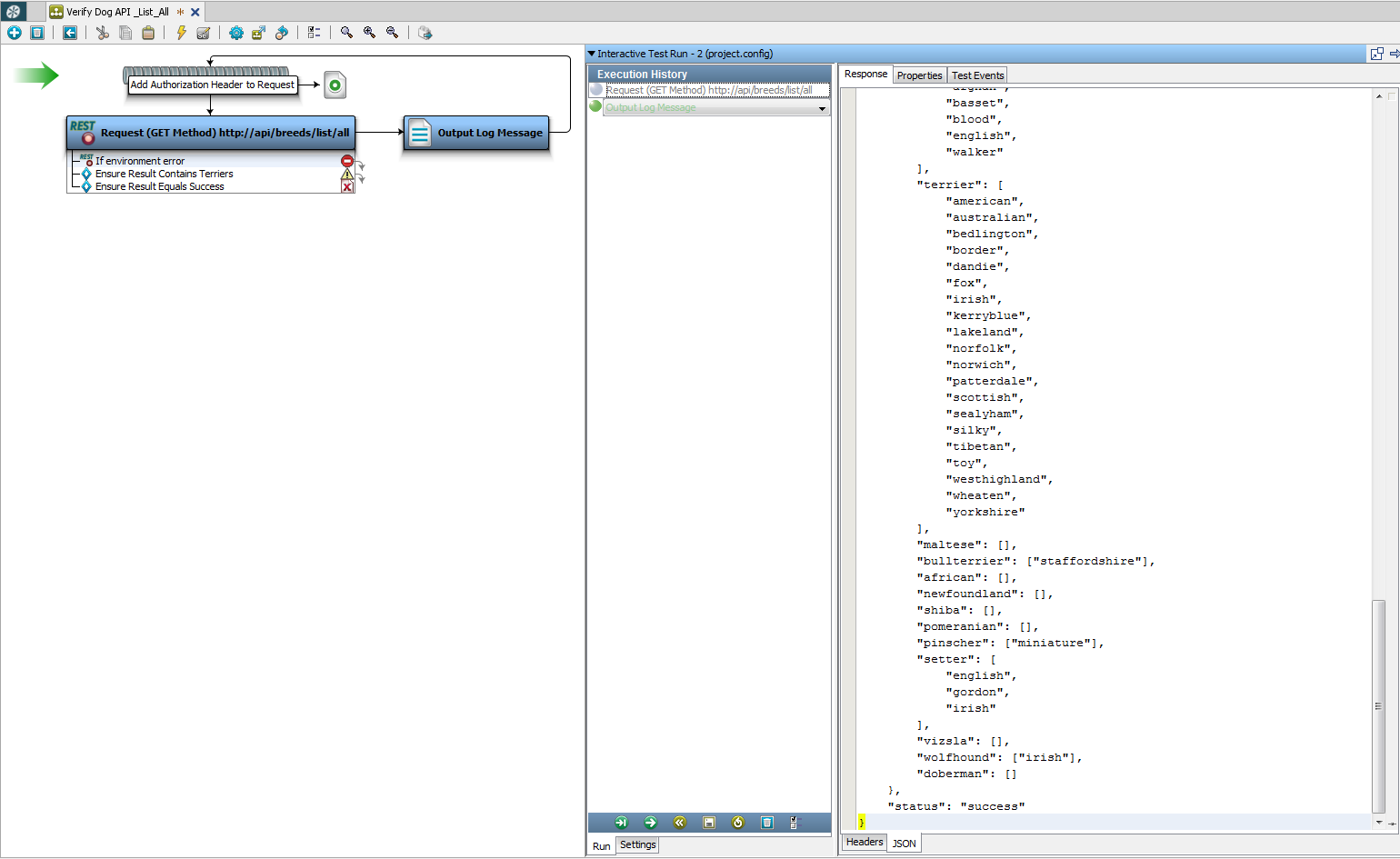


Click the Play (green button at bottom to run the test case)

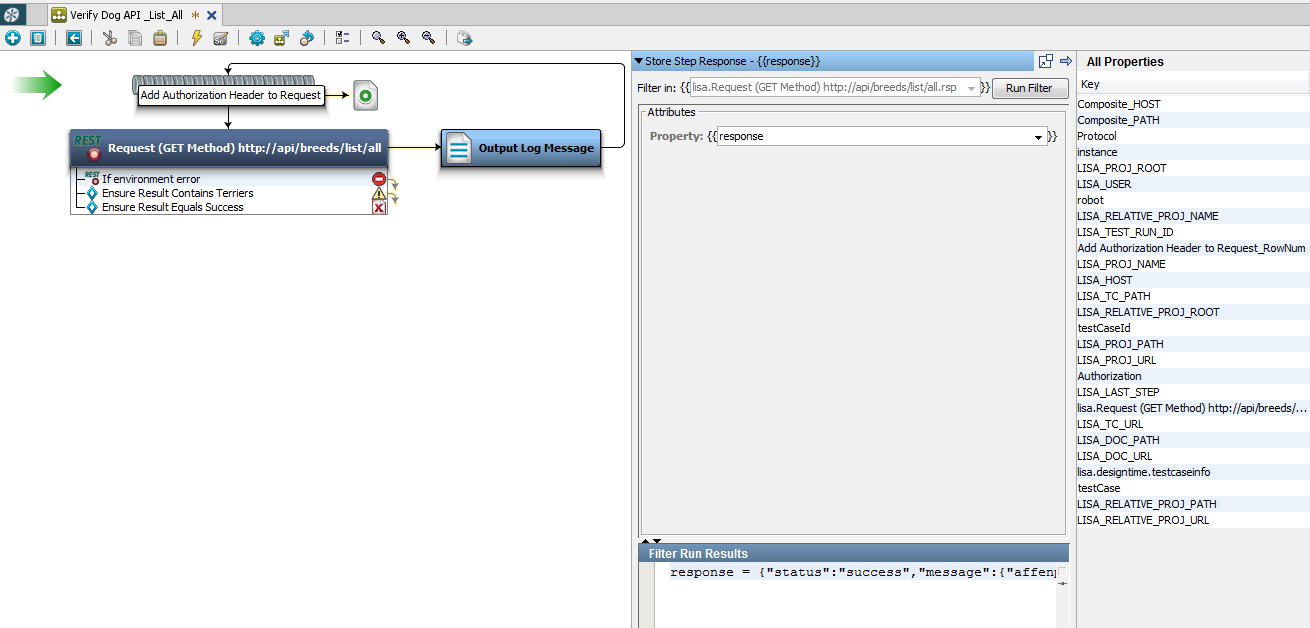


The Test Case will run as follows:

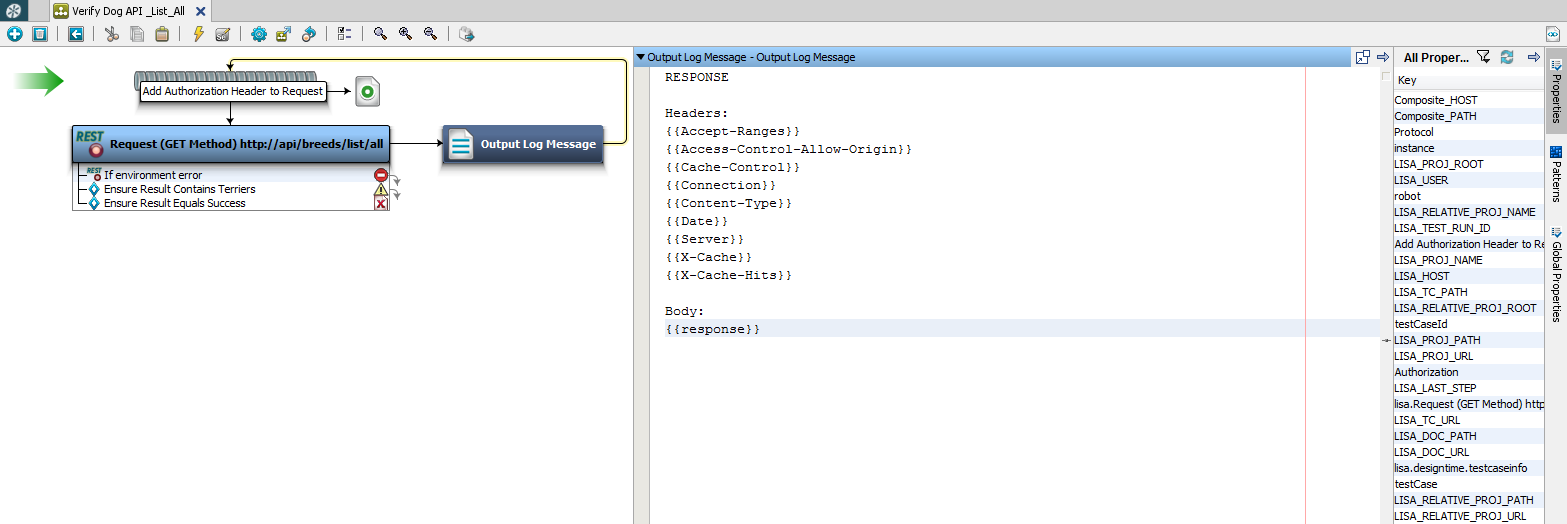
1



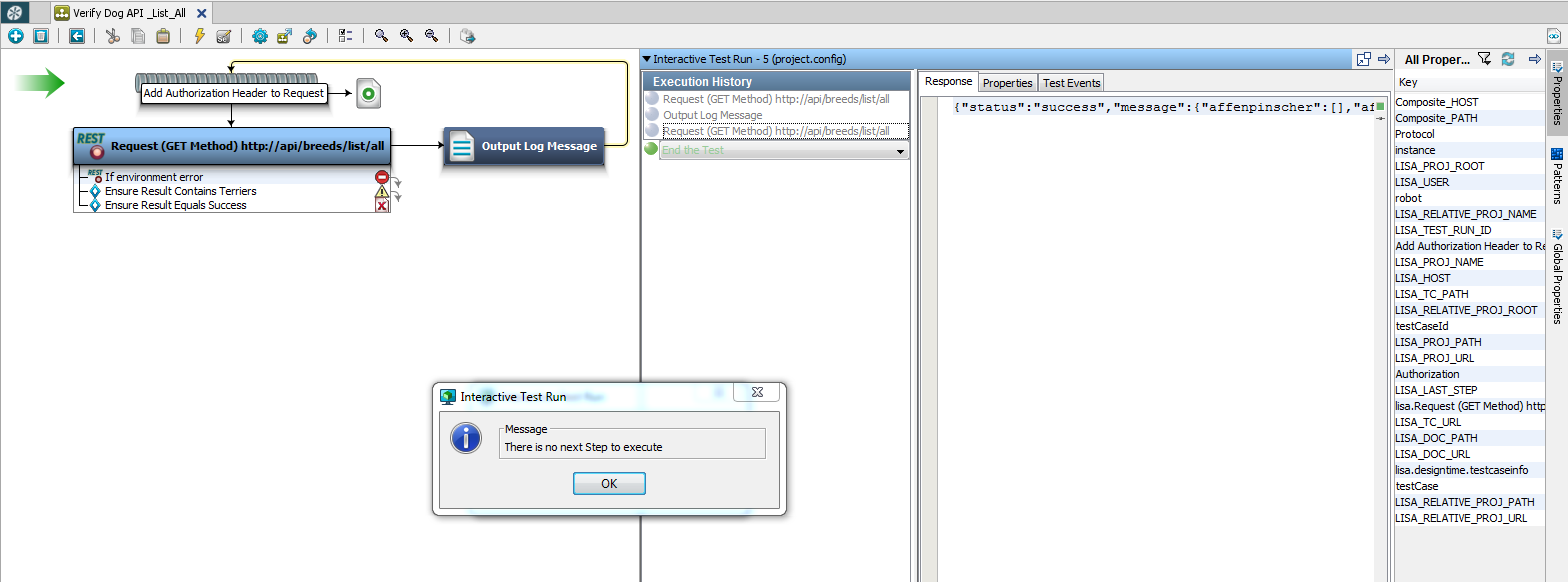
2



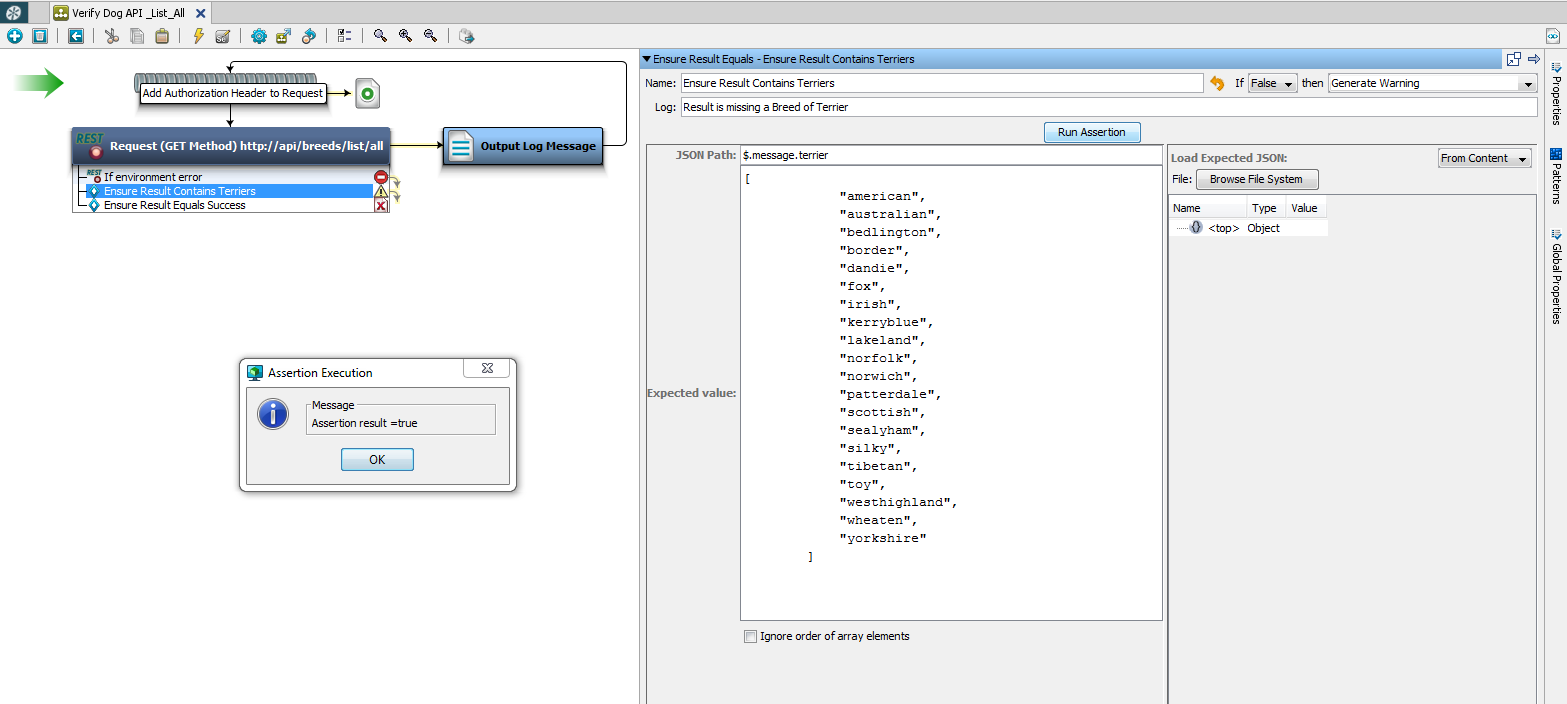
3



4

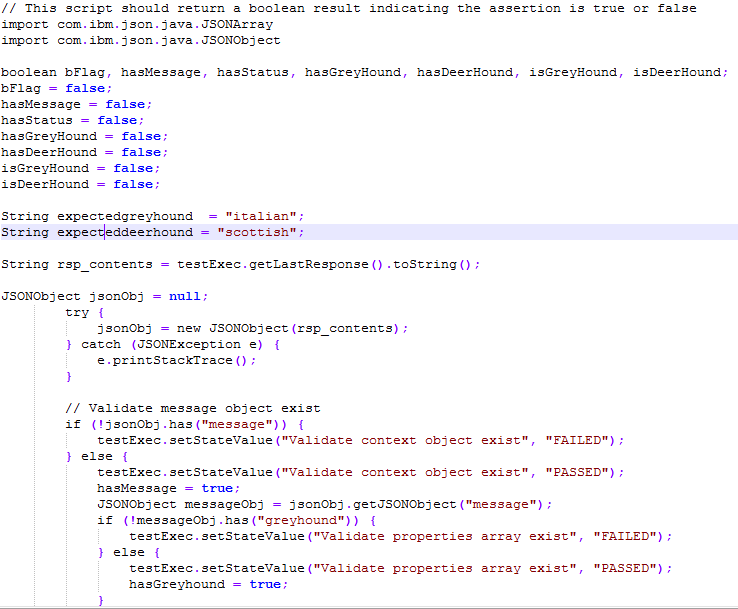


5



**Additional Information:**

* Scripted Assertions can also be used when UI is not preferred



* Tester can also run a “TestSuite” for running multiple test cases such as from directory {{LISA\_RELATIVE\_PROJ\_ROOT}}/Tests/FunctionalTestScripts/Verify\_Dog\_APIs
* User can create config for parameterizing name-value pairs. For example: Instead of inputting the entire URL into a REST call for each test case, you can create modify config such that

{{Protocol}}://{{CompositeHOST}}/{{CompositePath}}is the defined targeted URL