## SOAP

* SOAP is an XML-based protocol for accessing web services over HTTP.
* SOAP is known as the Simple Object Access Protocol.
* SOAP is a protocol or in other words is a definition of how web services talk to each other or talk to client applications that invoke them.
* SOAP was developed as an intermediate language so that applications built on various programming languages could talk easily to each other and avoid the extreme development effort.
* In today's world, there is huge number of applications which are built on different programming languages. For example, there could be a web application designed in Java, another in .Net and another in PHP.
* Exchanging data between applications is crucial in today's networked world. But data exchange between these heterogeneous applications would be complex. So will be the complexity of the code to accomplish this data exchange.
* One of the methods used to combat this complexity is to use XML (Extensible Markup Language) as the intermediate language for exchanging data between applications.
* Every programming language can understand the XML markup language. Hence, XML was used as the underlying medium for data exchange.
* But there are no standard specifications on use of XML across all programming languages for data exchange. That is where SOAP comes in.
* SOAP was designed to work with XML over HTTP and have some sort of specification which could be used across all applications.

## Advantages of SOAP

* SOAP is the protocol used for data interchange between applications. Below are some of the reasons as to why SOAP is used.
* When developing Web services, you need to have some of language which can be used for web services to talk with client applications. SOAP is the perfect medium which was developed in order to achieve this purpose
* SOAP is designed to be platform independent and is also designed to be operating system independent. So the SOAP protocol can work any programming language based applications on both Windows and Linux platform.
* It works on the HTTP protocol –SOAP works on the HTTP protocol, which is the default protocol used by all web applications. Hence, there is no sort of customization which is required to run the web services built on the SOAP protocol to work on the World Wide Web.

**SOAP MESSAGE STRUCTURE:**

* Whenever a client application calls a method in the web service, the web service will automatically generate a SOAP message which will have the necessary details of the data which will be sent from the web service to the client application.

A simple SOAP Message has the following elements –

* The Envelope element
* The header element and
* The body element
* The Fault element (Optional)

The SOAP message is nothing but a mere XML document which has the below components.

**Envelope** : An **Envelope** element that identifies the XML document as a SOAP message – This is the containing part of the SOAP message and is used to encapsulate all the details in the SOAP message. This is the root element in the SOAP message.

**Header** : A **Header** element that contains header information – The header element can contain information such as authentication credentials which can be used by the calling application.

* It can also contain the definition of complex types which could be used in the SOAP message. By default, the SOAP message can contain parameters which could be of simple types such as strings and numbers, but can also be a complex object type.

**Example for Complex Type:**

* A simple example of a complex type is shown below.
* Suppose we wanted to send a structured data type which had a combination of a "Tutorial Name" and a "Tutorial Description," then we would define the complex type as shown below.
* The complex type is defined by the element tag <xsd:complexType>. All of the required elements of the structure along with their respective data types are then defined in the complex type collection.
* <xsd:complexType>
* <xsd:sequence>
* <xsd:element name="Tutorial Name" type="string"/>
* <xsd:element name="Tutorial Description" type="string"/>
* </xsd:sequence>
* </xsd:complexType>

**Body** : A **Body** element that contains call and response information – This element is what contains the actual data which needs to be sent between the web service and the calling application.

* Below is an example of the SOAP body which actually works on the complex type defined in the header section.
* Here is the response of the Tutorial Name and Tutorial Description that is sent to the calling application which calls this web service.

<soap:Body>

<GetTutorialInfo>

<TutorialName>Web Services</TutorialName>

<TutorialDescription>All about web services</TutorialDescription>

</GetTutorialInfo>

</soap:Body>



1. As seen from the above SOAP message, the first part of the SOAP message is the envelope element which is used to encapsulate the entire SOAP message.
2. The next element is the SOAP body which contains the details of the actual message.
3. Our message contains a web service which has the name of "Guru99WebService".
4. The "Guru99Webservice" accepts a parameter of the type 'int' and has the name of TutorialID.

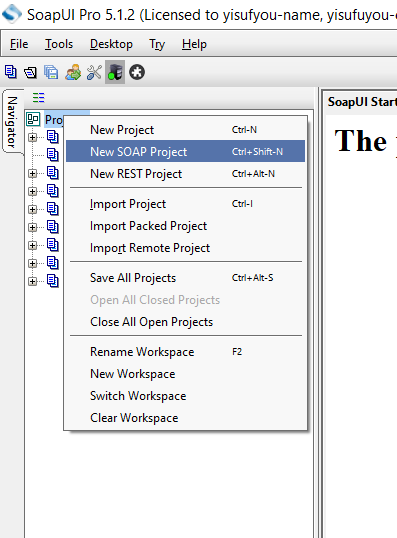
Now, the above SOAP message will be passed between the web service and the client application.

* You can see how useful the above information is to the client application. The SOAP message tells the client application what is the name of the Web service, and also what parameters it expects and also what is the type of each parameter which is taken by the web service.

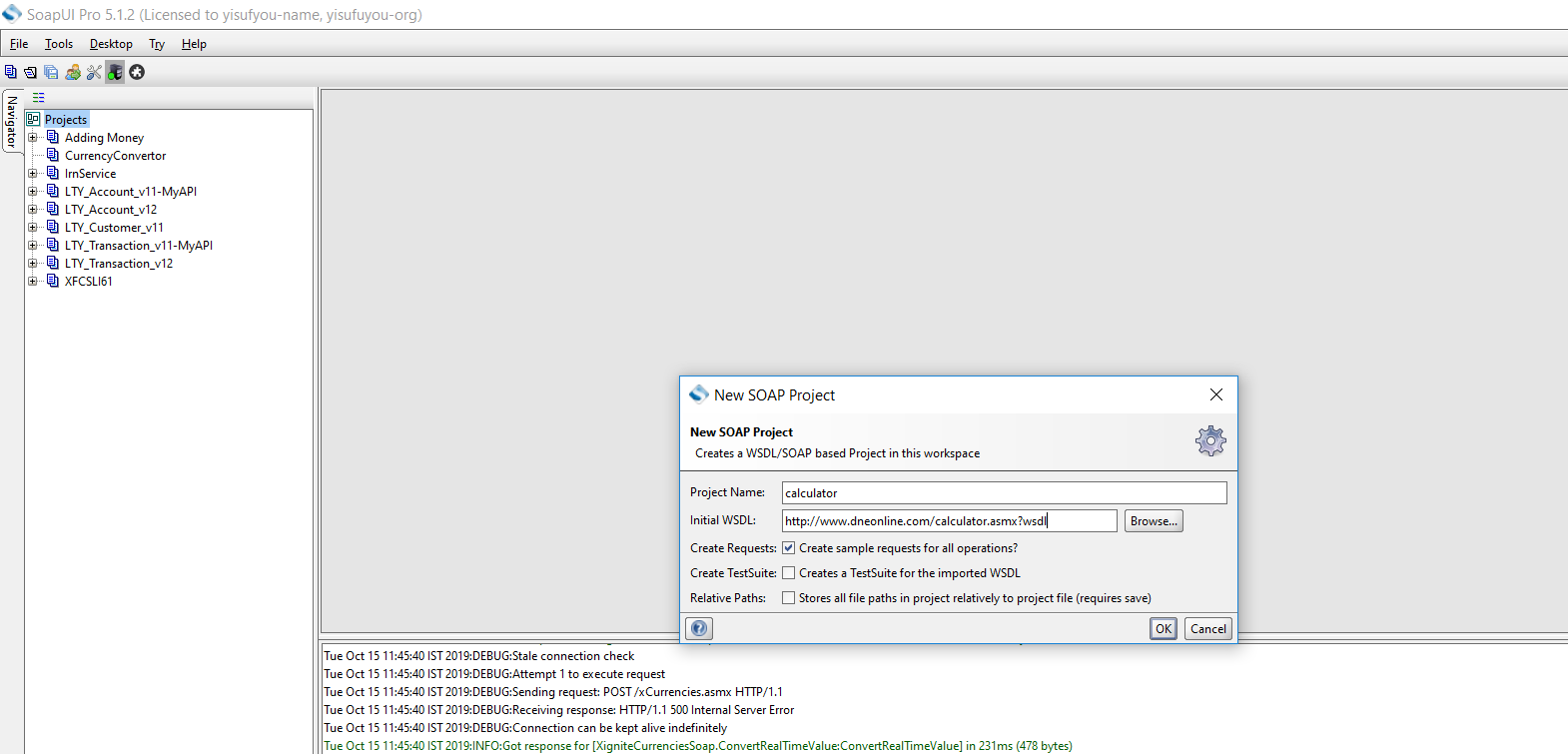
# **Create a Project, Test Suite, TestCase**

## Create a Project

**Step 1:**Now, depending upon the project, we need to import SOAP/REST protocol. We will create a new SOAP Project.

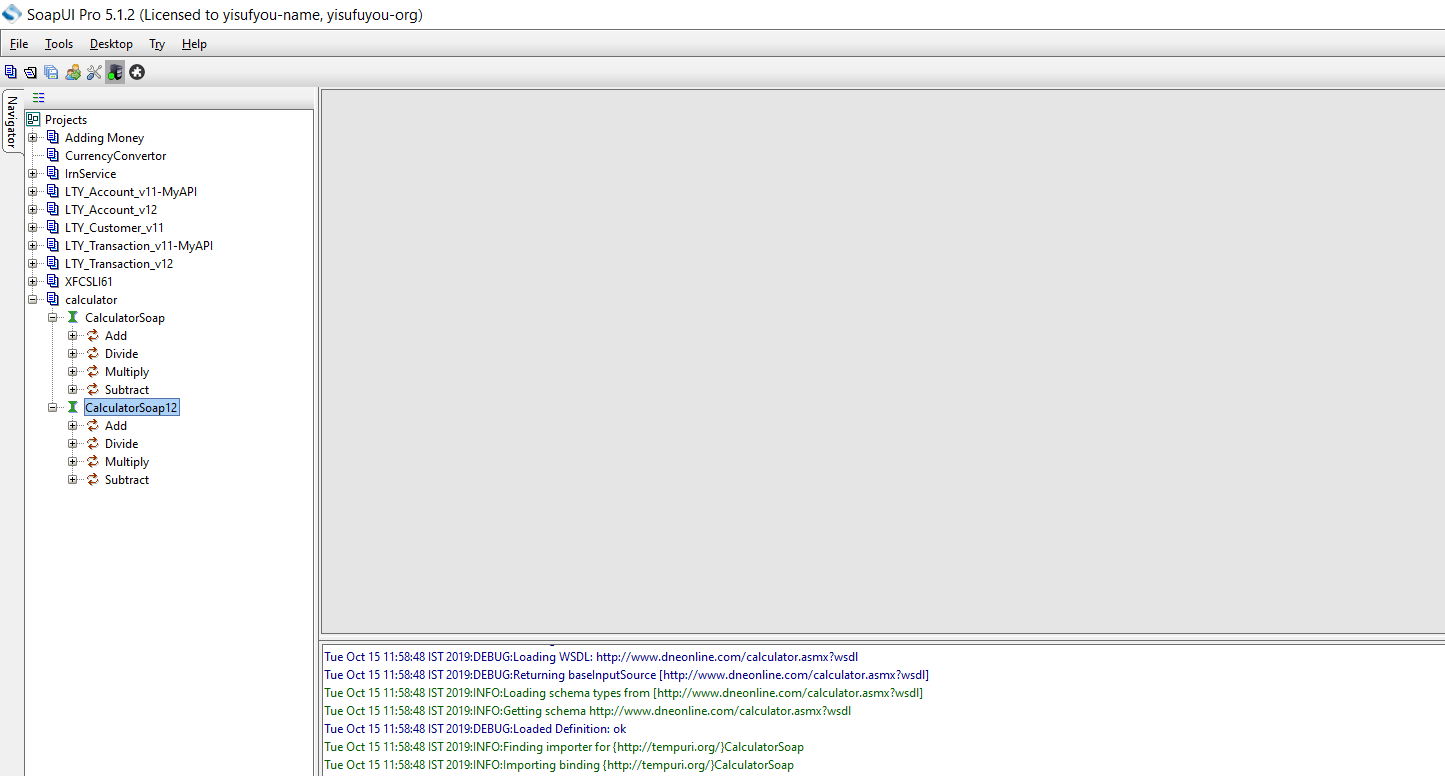


**Step 2:**We will make use following SOAP request <http://www.dneonline.com/calculator.asmx?wsdl>



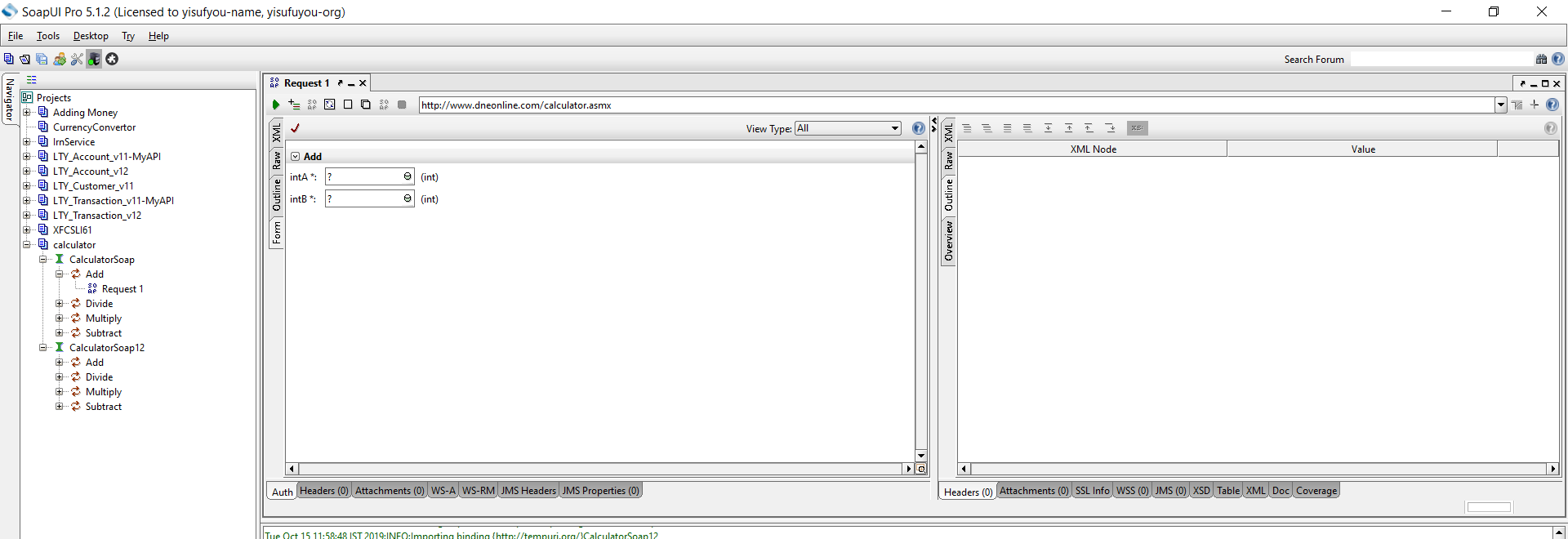
1. Enter the Project Name
2. Enter the path of the WSDL request. In this case http://www.dneonline.com/calculator.asmx?wsdl
3. Click OK

**Step 3:**Upon creating the SOAP project with the above-said WSDL, we will be able to see that there are two operations that will be imported into the project.

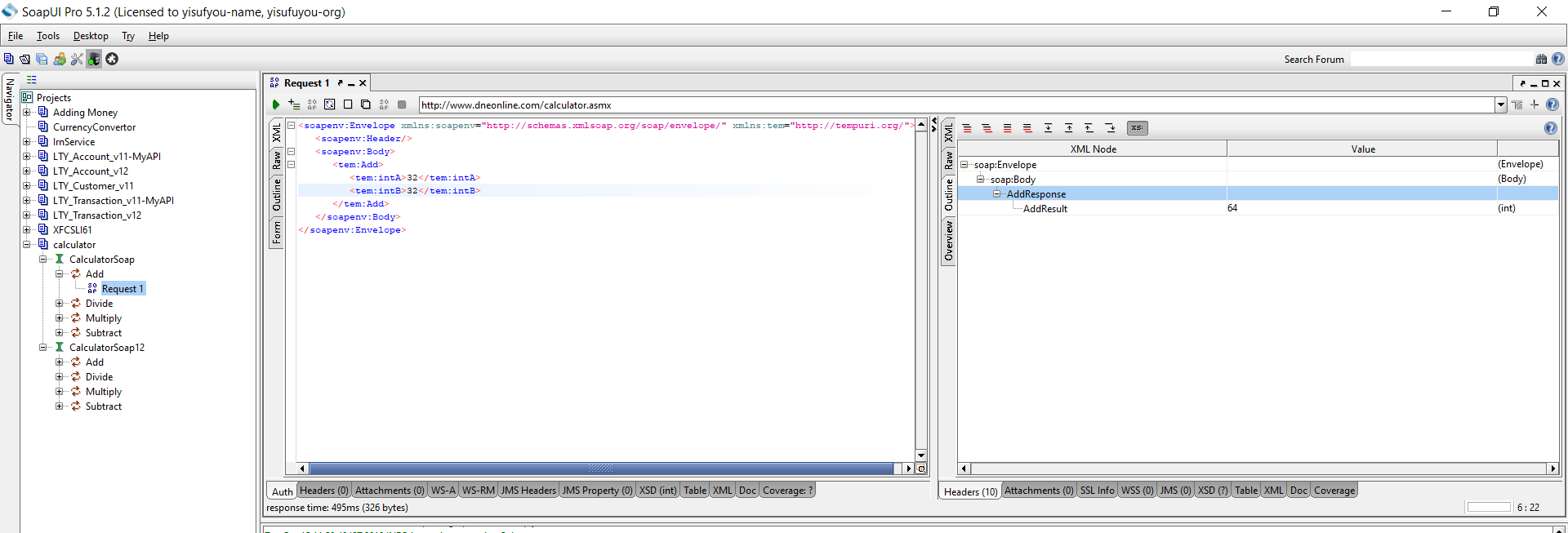


**Step 4** Expand the first request and right-click on the 'Add.' Then click on 'New Request'.

Then Click on 'OK'. It will display the SOAP request in the XML format

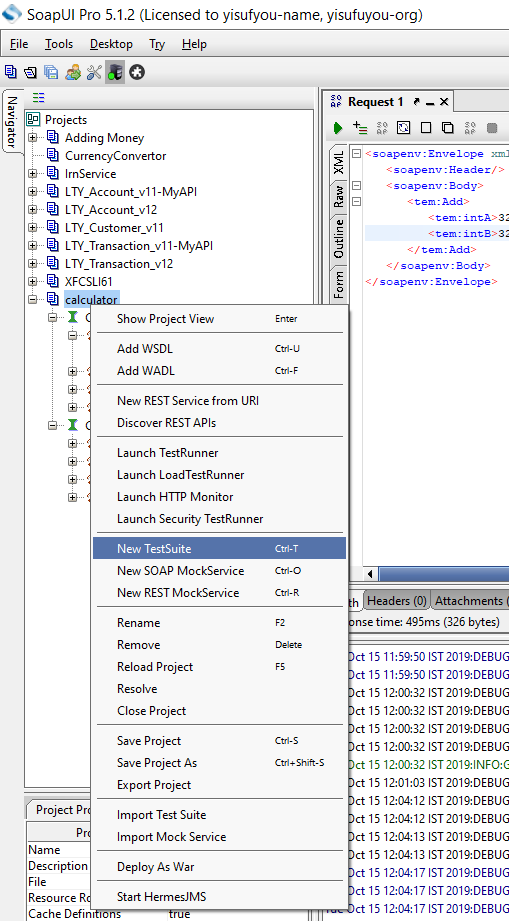


1. Enter the 'intA' and 'intB'
2. Click on the submit button
3. Response XML will be displayed right side pane.

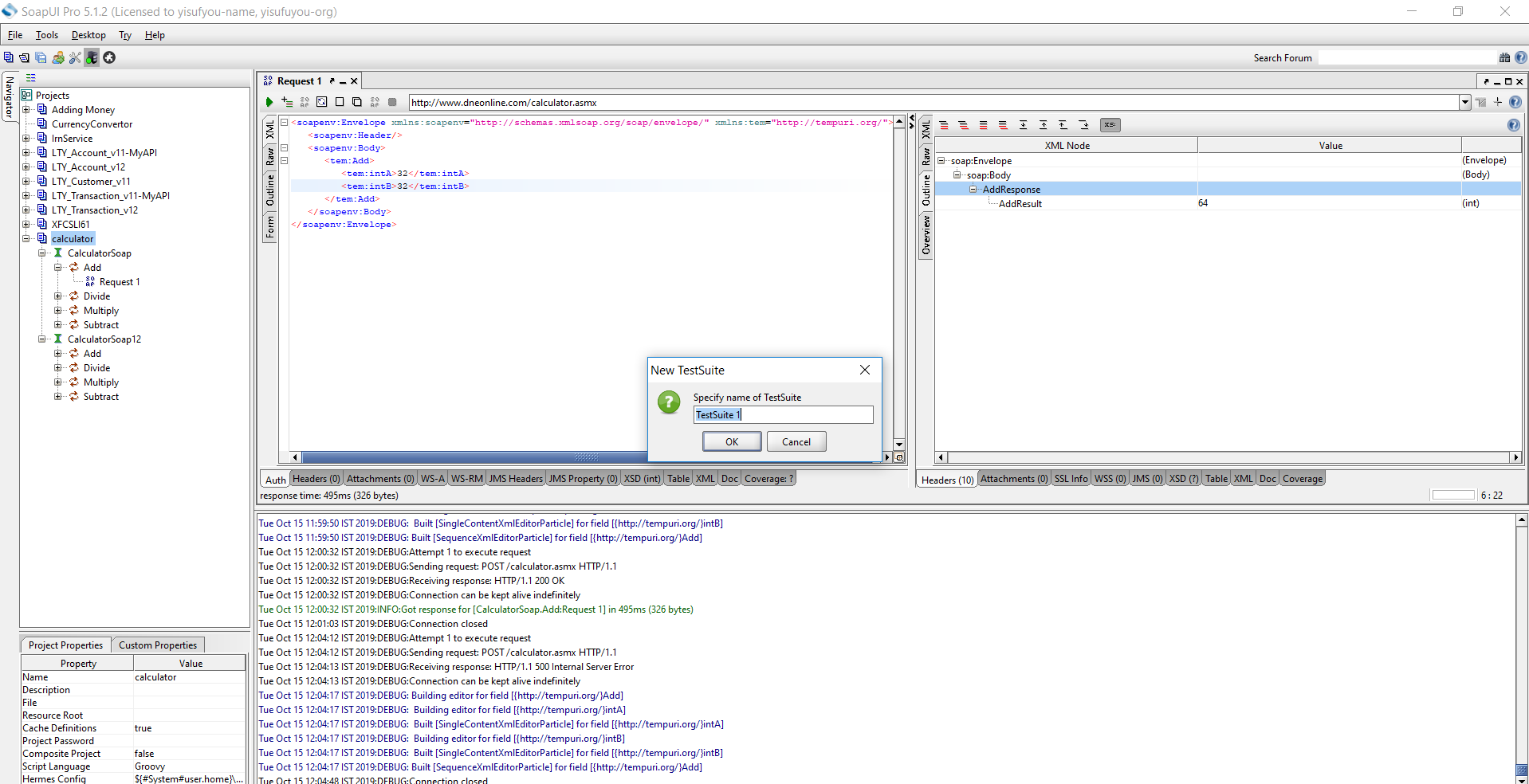


## Creating Test Suite

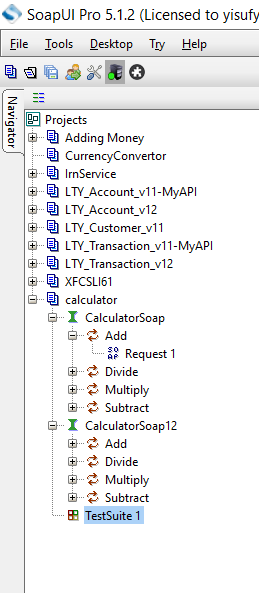
**Step 1:**Within the project, testers can create a test suite by performing a right-click on the root of the project.



**Step 2:**We need to enter the name of the test suite and press OK.

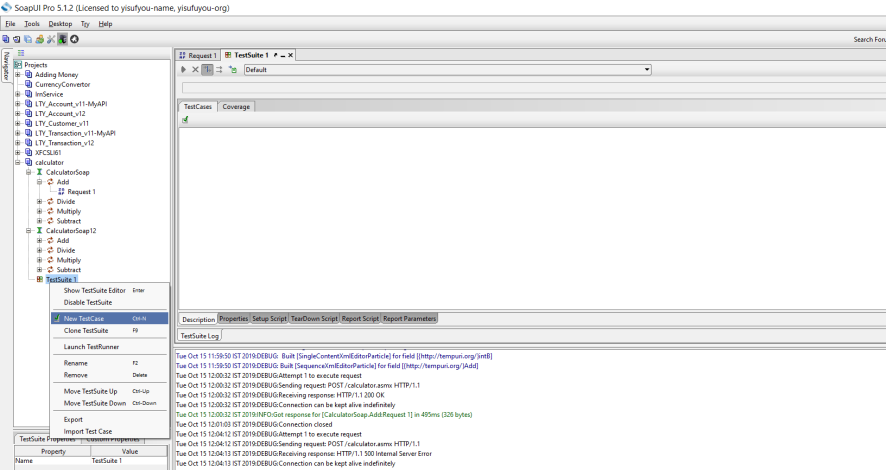


**Step 3:**The created test suite is displayed the navigator pane as shown below.

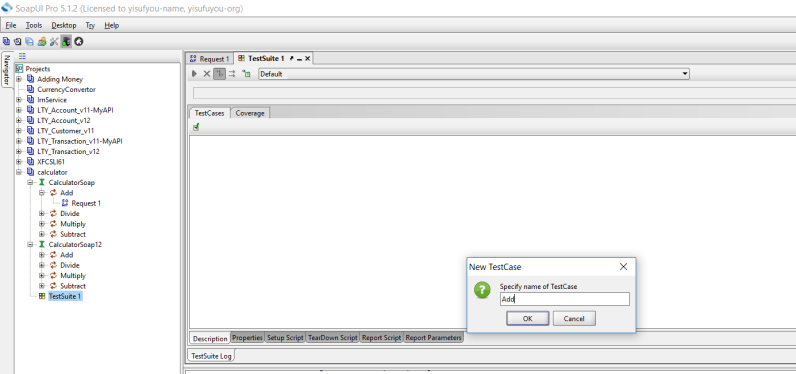


## Creating Test Case

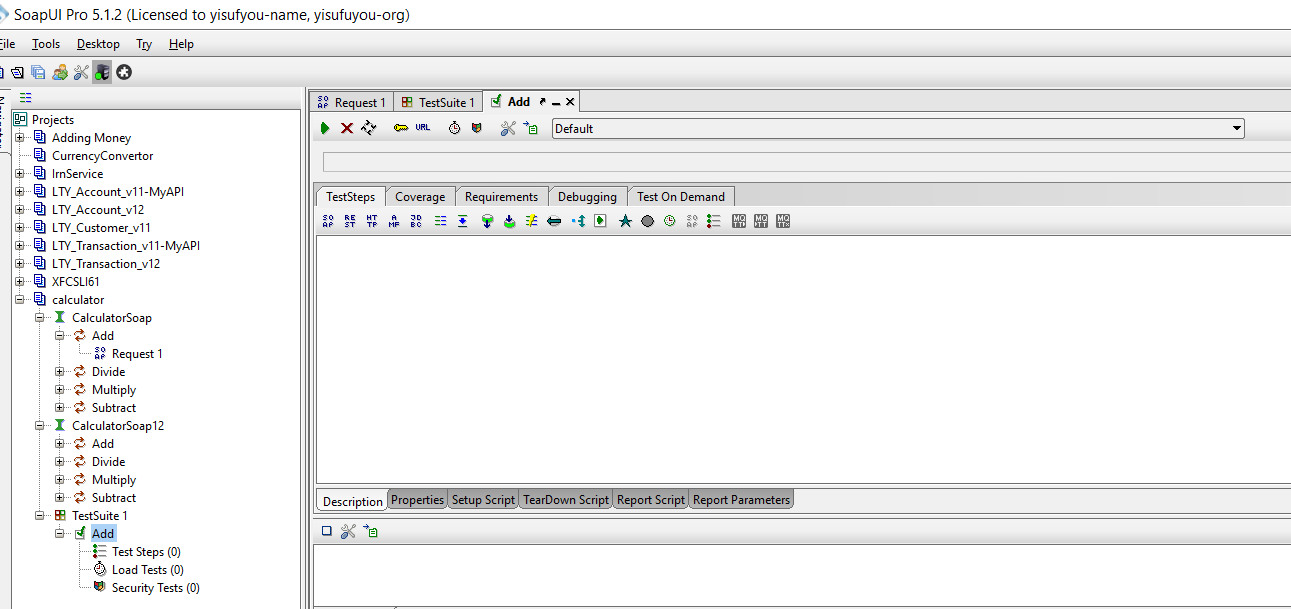
**Step 1:**Within a test suite, we can create multiple tests by performing right click on the 'test suite' and choosing 'New TestCase'.



**Step 2:**Specify the name of the[Test Case](https://www.guru99.com/test-case.html)and click 'OK'.



**Step 3:**The created test case has zero steps as shown below.

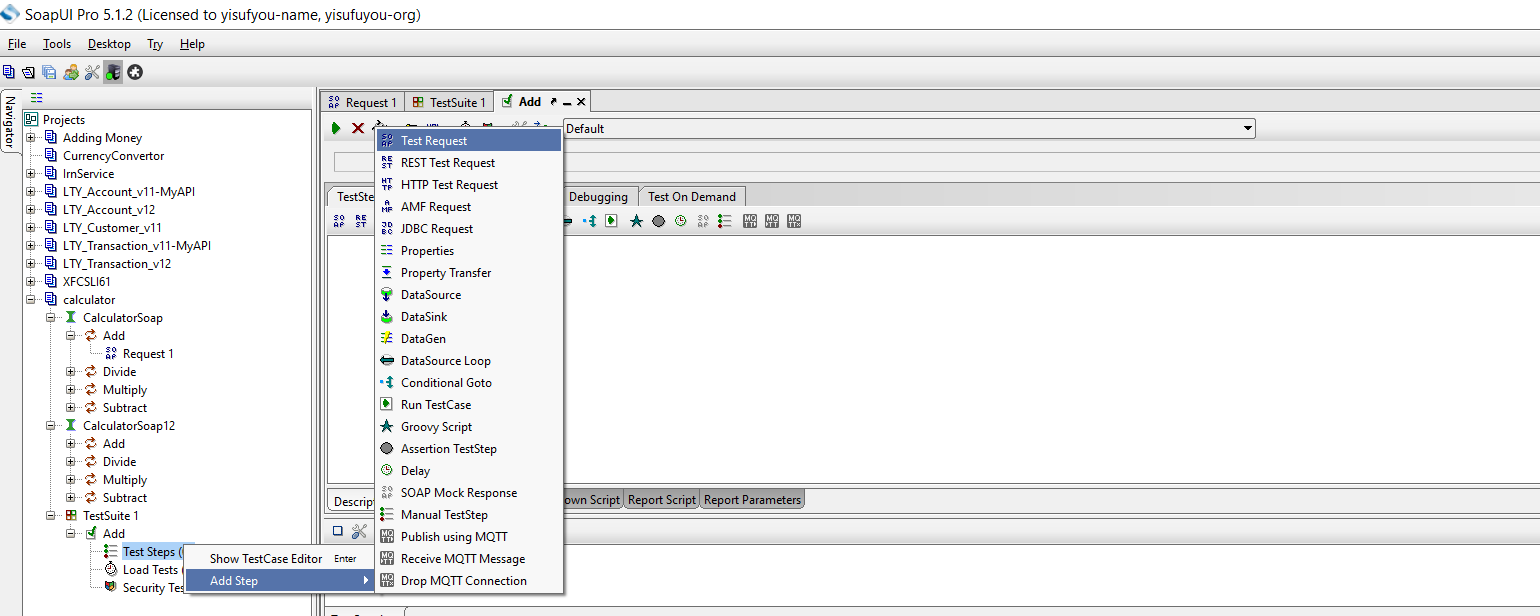


**NOTE:** We can insert a variety of test steps by performing a right-click on test steps and selecting an appropriate test step as shown below. So, if you were to test a REST Webservice, you would select the REST Test Request.

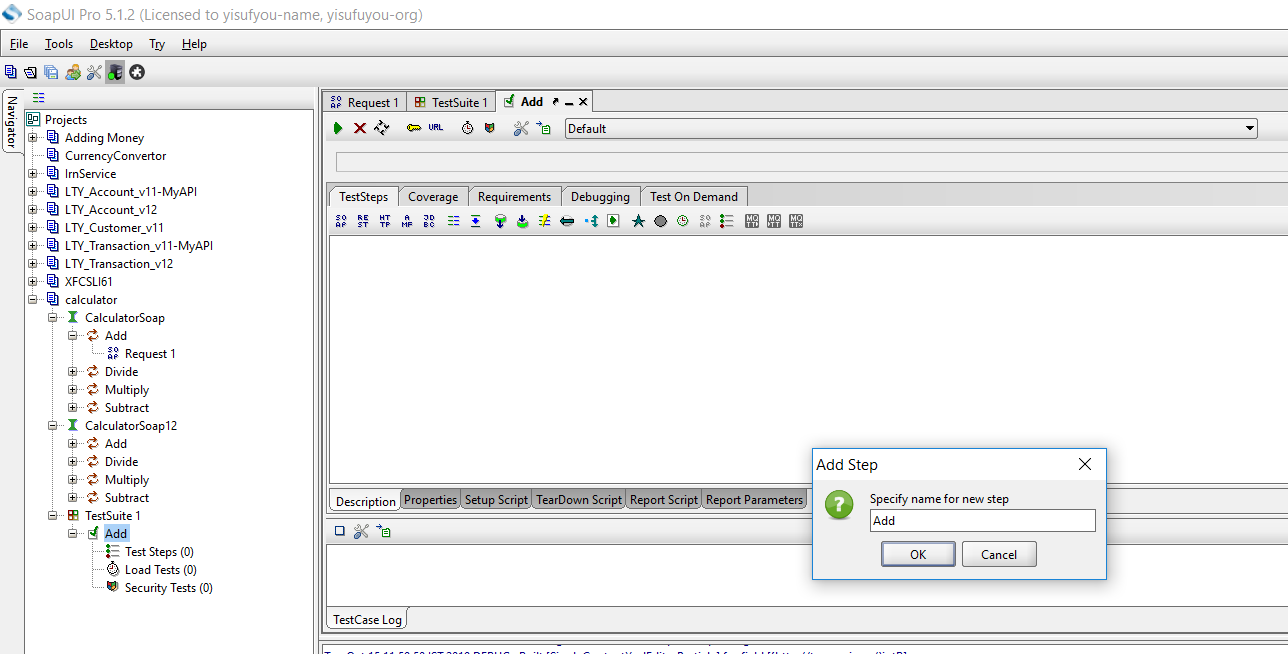
## Test Step Insert

Now let us add a test step to validate the imported SOAP request.

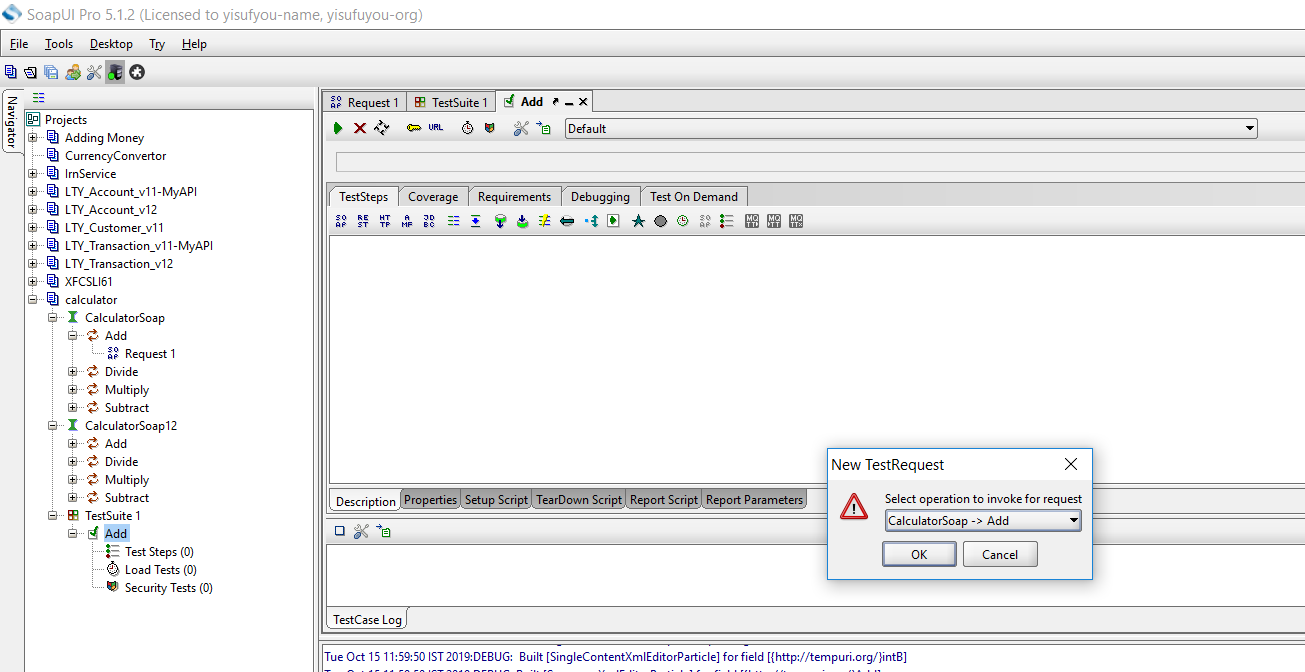
**Step 1:**Add a new step 'SOAP Request' as shown below.



**Step 2:**Enter the step name and click OK.

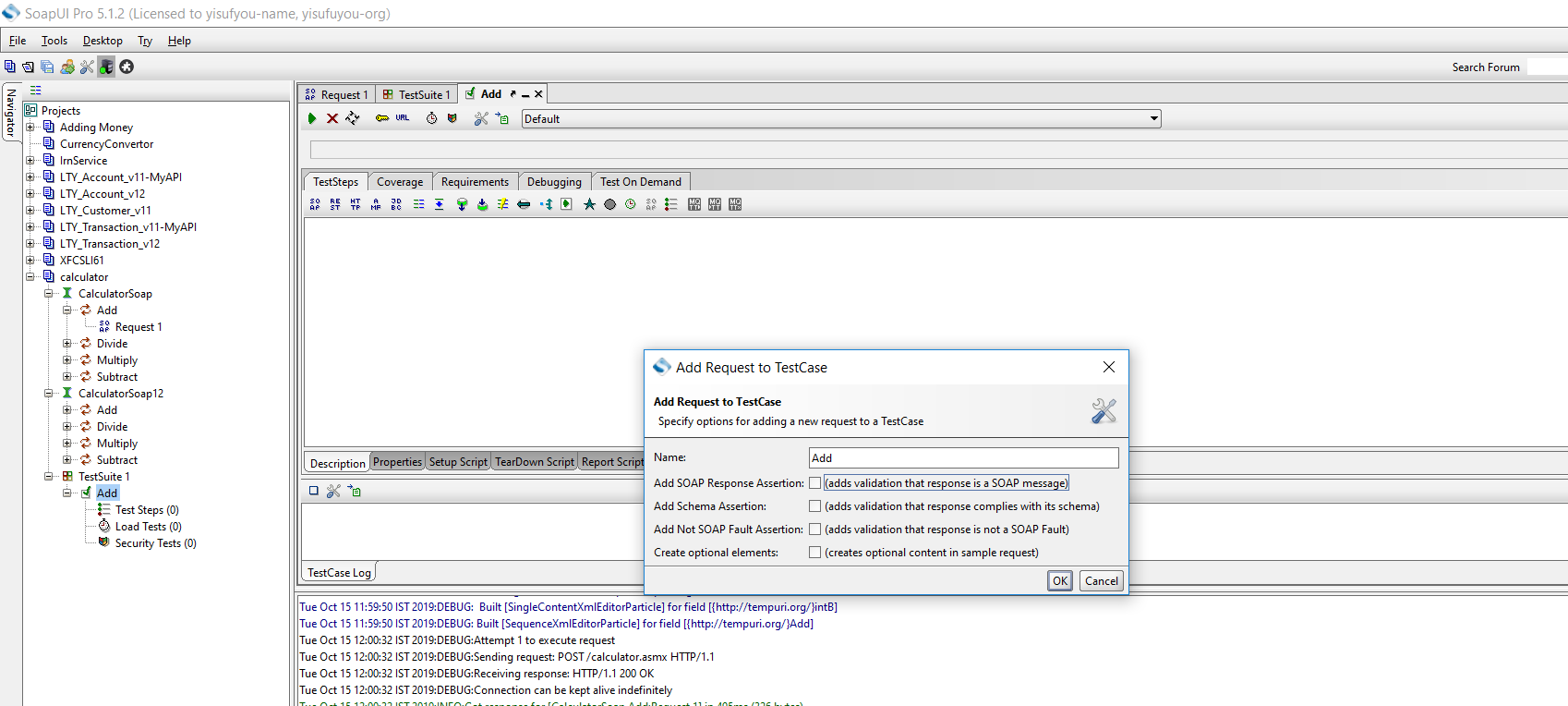


**Step 3:**Upon clicking 'OK', a dialog pops up to select the operation to invoke. All the operations are listed, and user can select the operation that they would like to invoke. Upon Selecting the operation, click 'Ok'



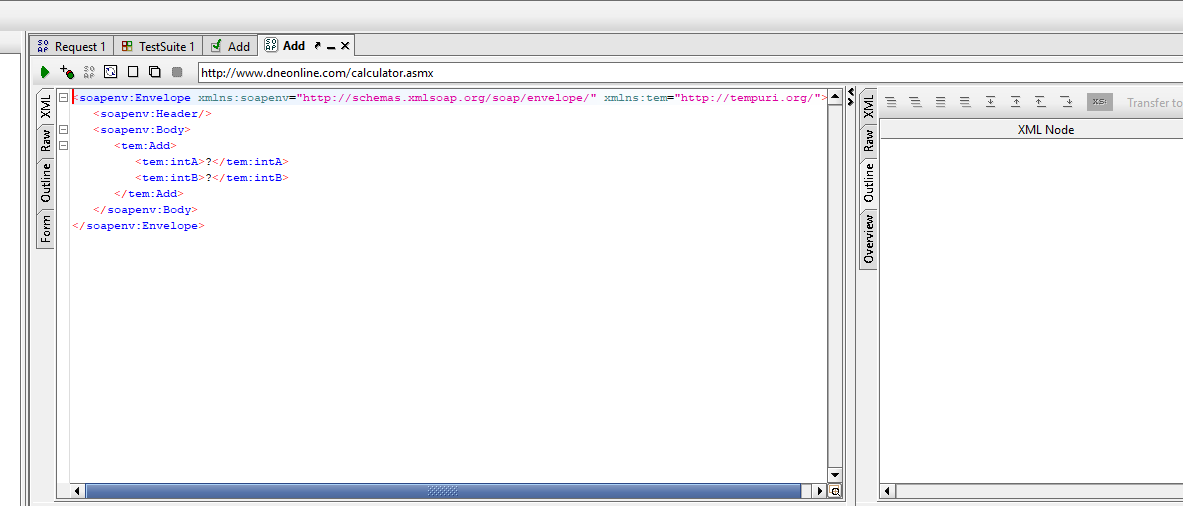
**Step 4:**Whileaddinga test case, we can add standard assertions. Assertions also called as checkpoints/validation points.

We can add following checkpoints/assertions while creating test case. Let us create a test case with the option which means creating test step WITHOUT any of the below validation points



1. Verifies if the response message is SOAP, upon executing the test.
2. Verifies if the response schema is valid.
3. Verifies if the SOAP response contains FAULT.

**Step 5:**Upon creating the test case, the request XML is shown below. The structure of the XML is explained within the below snapshot.

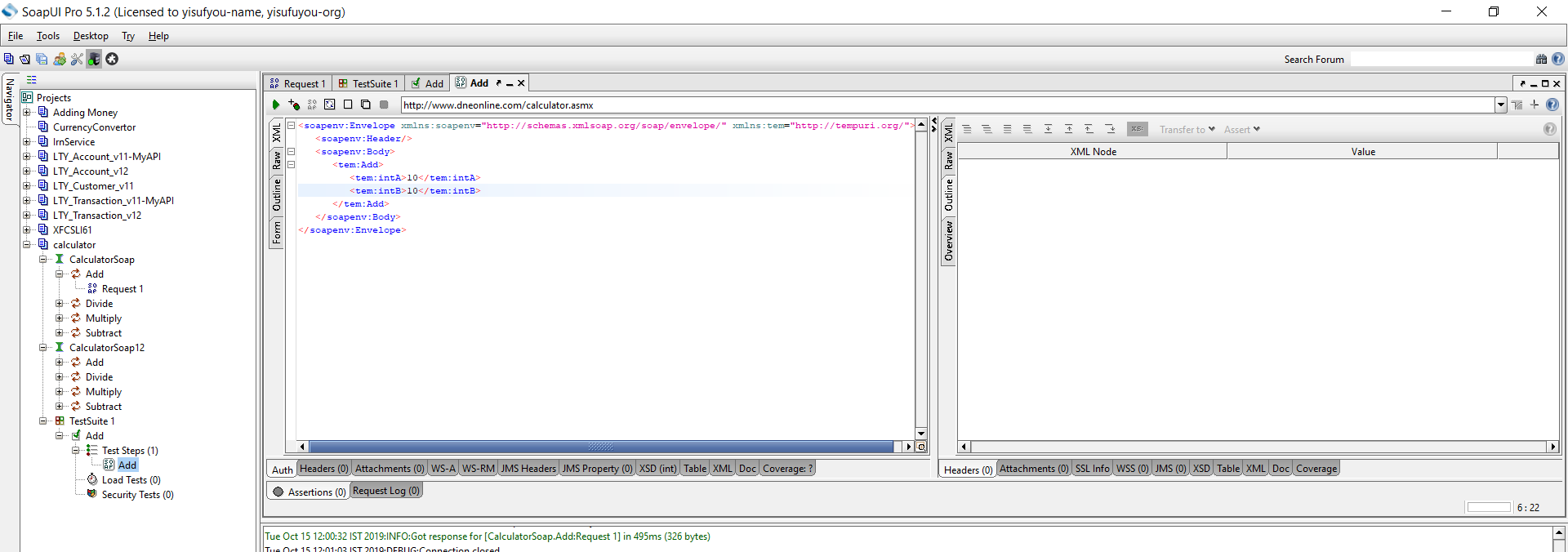


## Send Request Manually & Reading Response

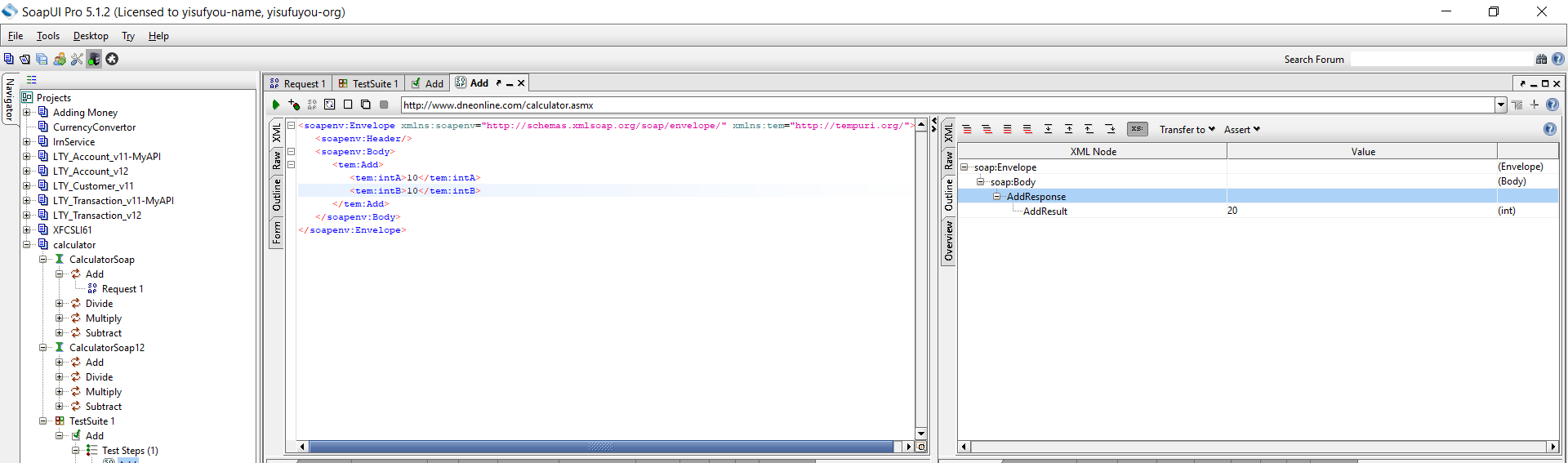
**Step 1:**We would like to add two Integer Number.

Next,

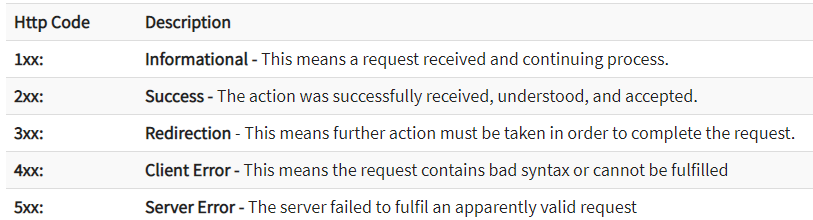
1. We need to enter these inputs in place of the question mark which will be sent as request XML.
2. After inputting those values into the corresponding XML tags, click 'submit request' button to check the response.



**Step 2:**Upon submitting a request the web service request is processed by the webserver and sends back a response as shown below.



**ERROR CODES:**



**Assertion**:

Once a request is sent to a web server a response is received. We need to validate if the response contains the data that we expect. In order to validate the response, we need to use assertions.

### **Types of Assertion**

There are various ways of asserting a response; however, we will focus on the commonly used Assertions types while validating a response. Below are the ones that are available in Open Source version of SoapUI.

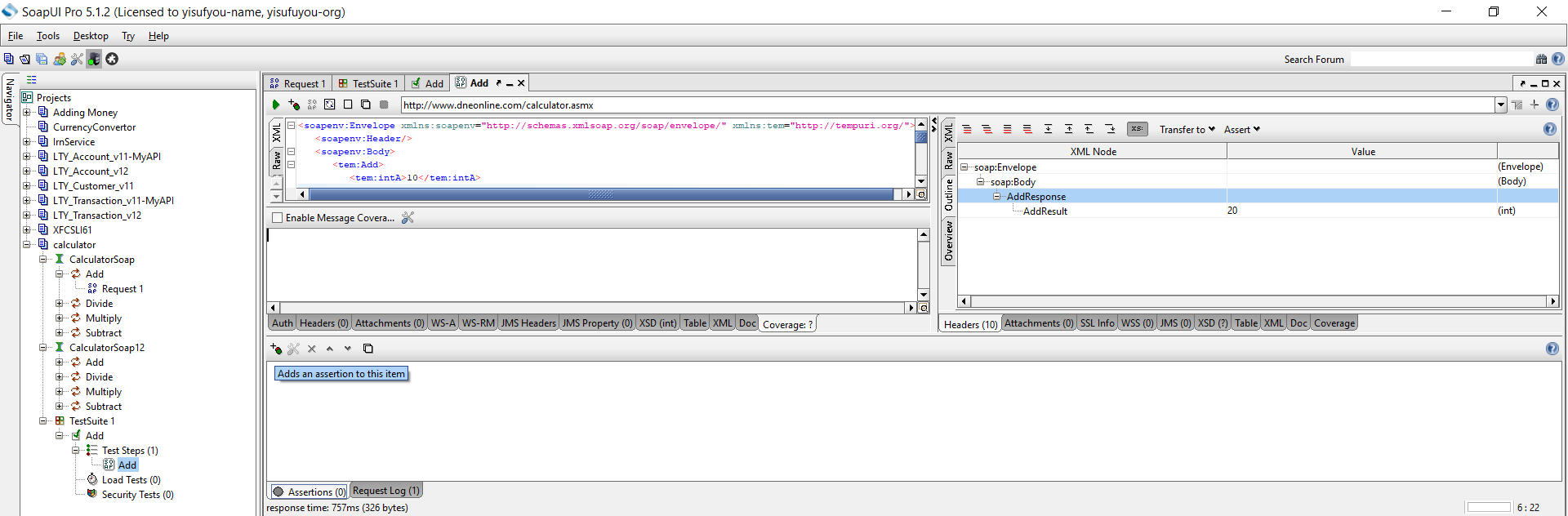
1. Property Content
2. Compliance Status Standard
3. Script
4. SLA
5. JMS
6. Security

## Contains Assertion

Searches for the existence of the specified string. It also supports regular expression.

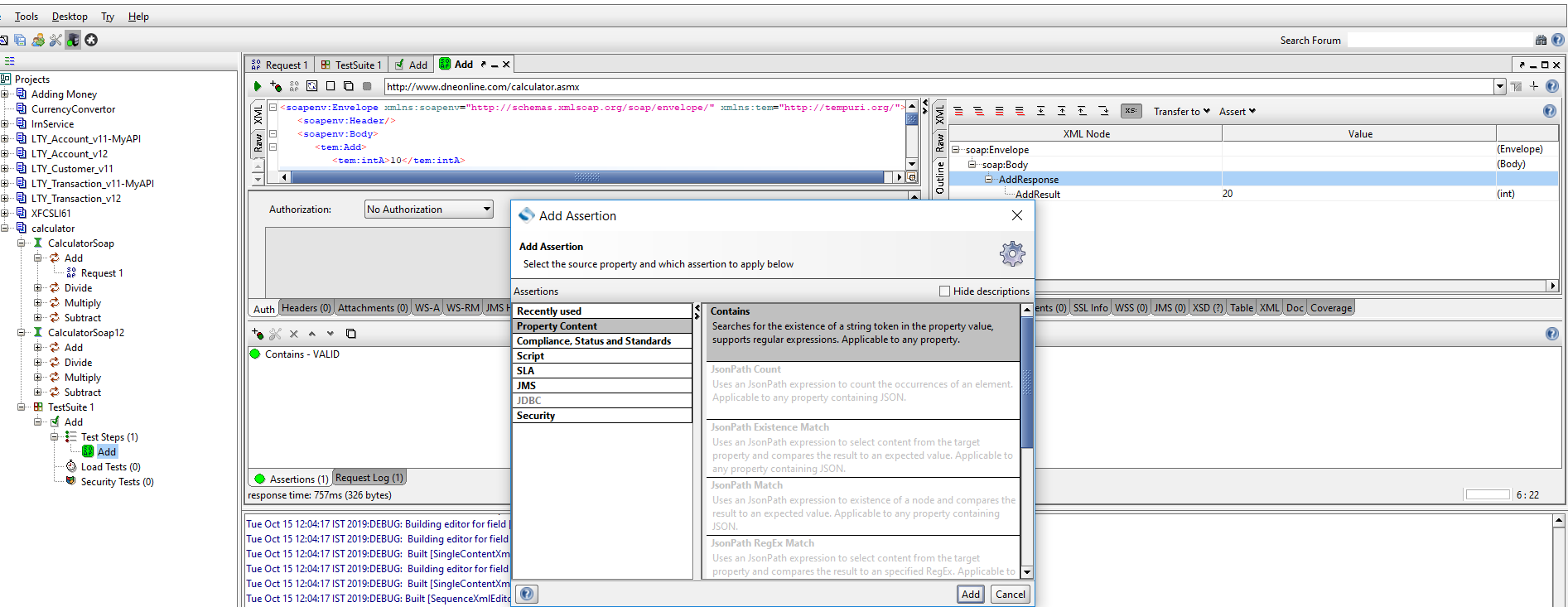
**Step 1:** By Default there are no assertions.

* The Number of Assertions are shown in the Assertions Tab.
* To add a new assertion, click on 'Add New Assertion' button.



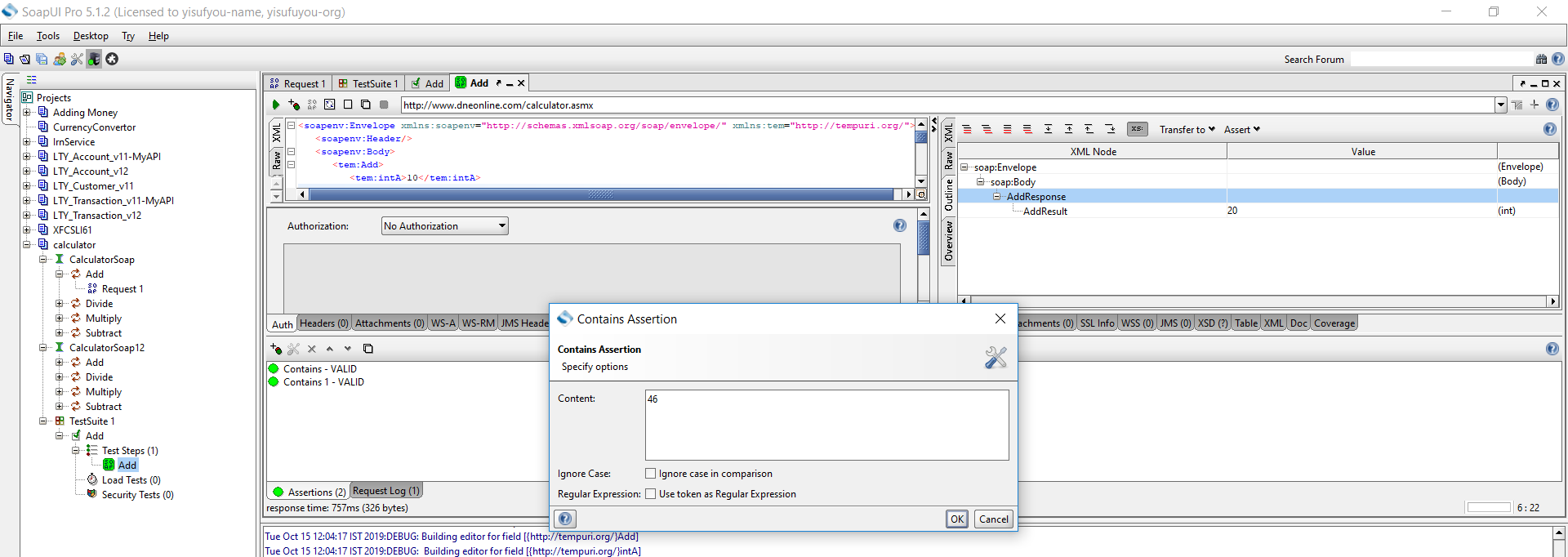
**Step 2:** Now,

1. Select the Assertion Category.
2. Select the Assertion Type.
3. Click 'Add'

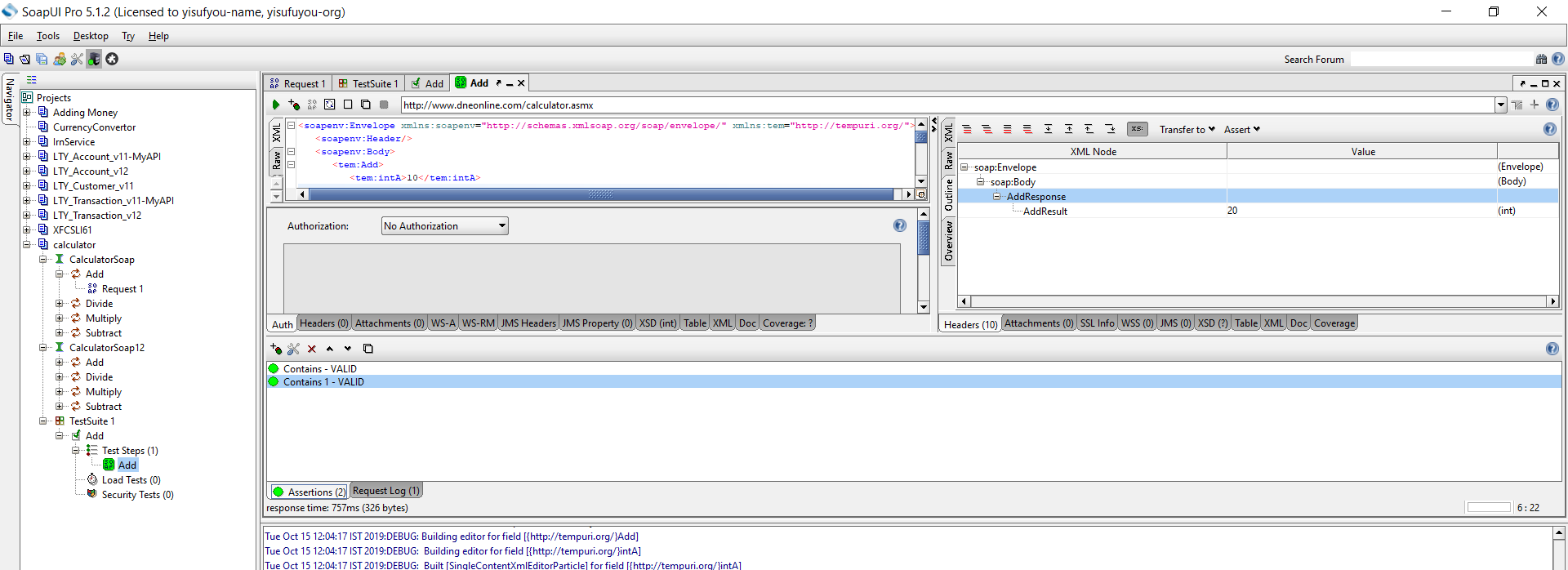


**Step 3:** Let us validate if the string 20 exist in the response. Click 'OK'

Note: We can also ignore case and add regular expression.



**Step 4:** Upon adding it, immediately assertion is executed and shows if VALID or INVALID.



NOTE:Assertion fails if that doesnot contains value

**Not Contains Assertion**

Searches for the Non-existence of the specified string. It also supports regular expression.

**Step 1:** Now after clicking on 'add new assertions' button,

1. Select the Assertion Category.
2. Select the Assertion Type – In this case 'NOT Contains'
3. Click 'Add'

## Xpath Assertion

Uses[XPath](https://www.guru99.com/xpath-selenium.html)expression to select the target node and its values. XPath, is an XML query language for selecting nodes from an XML document.

**Step 1:** Now after clicking on 'Add New Assertions' button,

1. Select the Assertion Category.
2. Select the Assertion Type – In this case 'XPath Match'
3. Click 'Add'

## Scripting Assertions

This Assertion technique is the most widely used one as it is extremely difficult to manage and maintain hundreds of assertions.

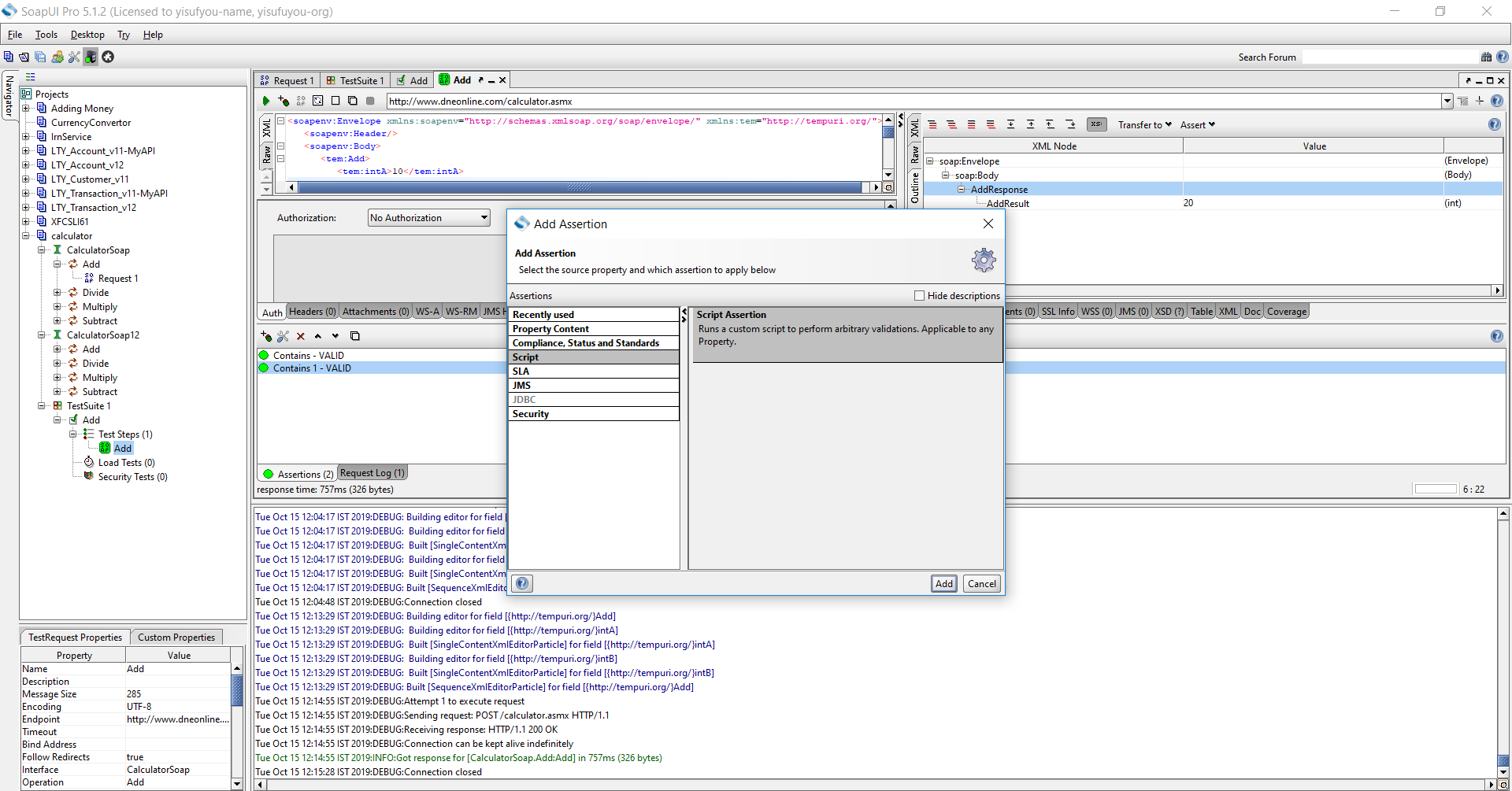
SOAP UI uses either Groovy Scripting or[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)for scripting assertions. The scripting technique is adopted for developing a framework for testing SOAP. Scripting assertions are used under following circumstances.

Scripting allows user to perform some operations before and after executing a TestCase using set up and tear down methods respectively. Set up is a procedure which is executed before executing a particular method(example – Object creation and Initialization) while tear down is a procedure which is executed after executing the method(eg: Destroying objects and clean up). This feature is not available in other Assertion types and can be done only through coding.

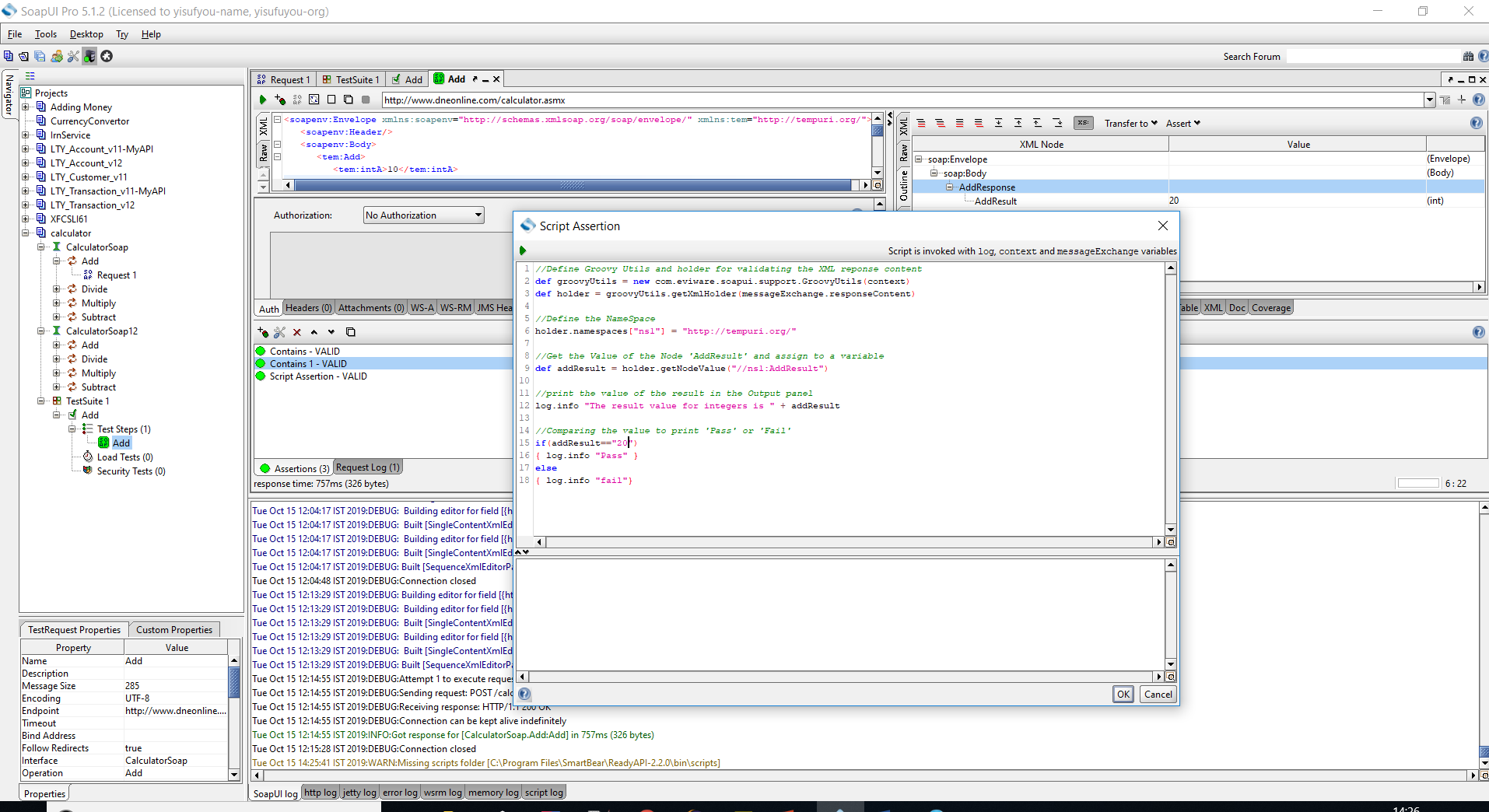
It allows users to perform opening/closing a Project, inorder to initialize or clean-up Project related settings and also to work with environmental variables which is very helpful during scripting.

It helps us in asserting a dynamic Response content.

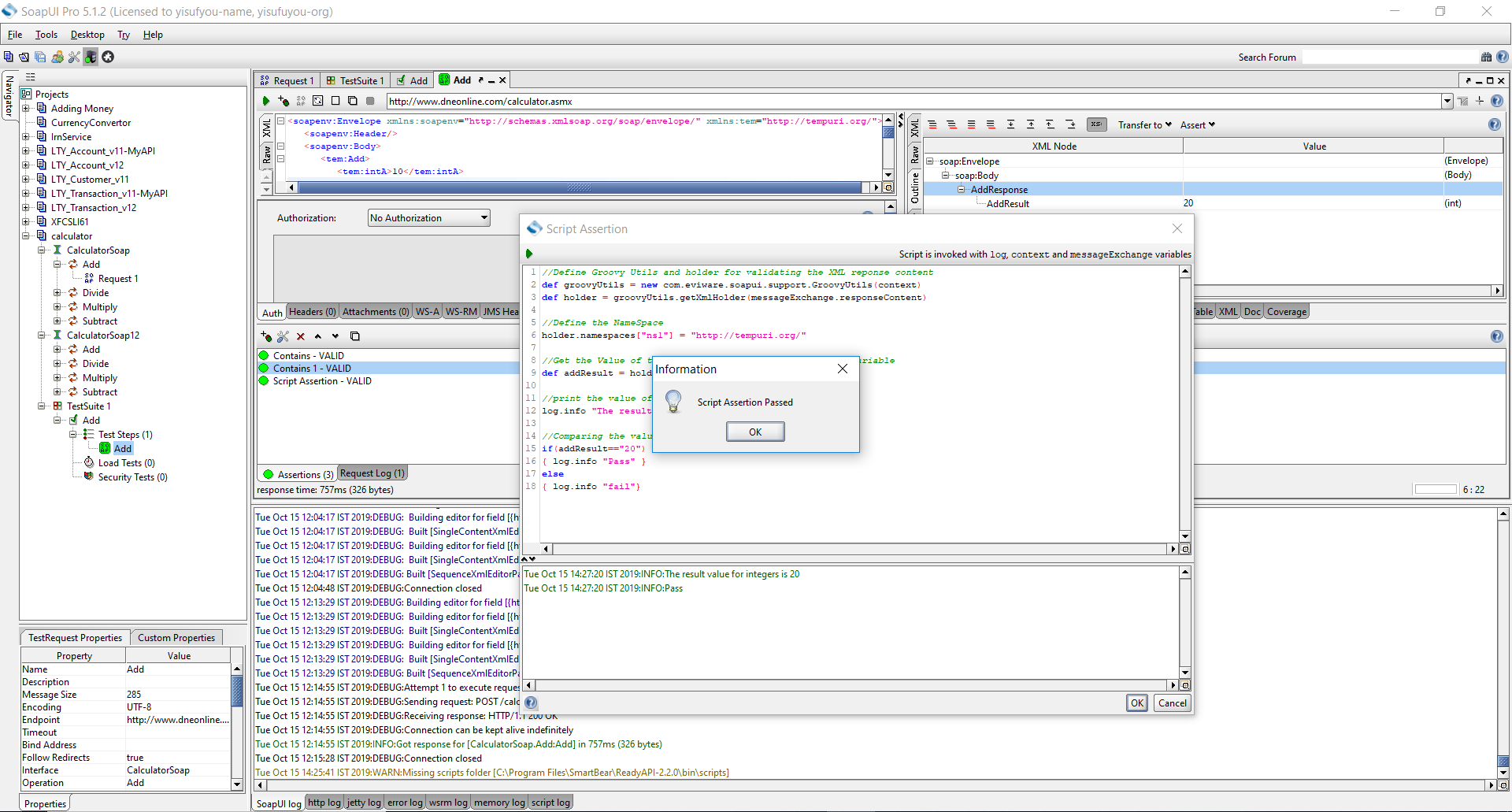
Scripting assertions are used for creating user defined assertions that are NOT predefined by SOAP UI.



**Groovy script to validate the Conversion Rate.**



1. Click 'Execute' Button to trigger the execution.
2. The output of the Script is shown in the Output pane. It has printed both, Conversion Value as well as the end result (Pass or Fail)
3. The Information is displayed that 'Script Assertion Passed'. Click OK.



Negative Case:

