**Write Groovy Script in SoapUI**

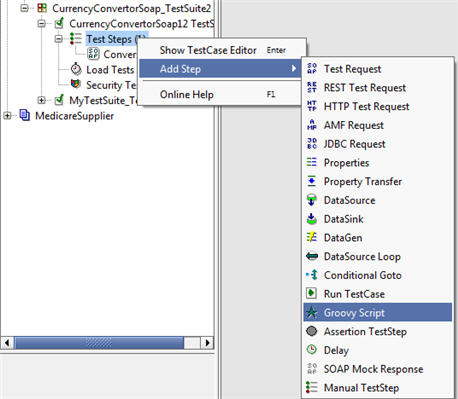
### ****SoapUI Groovy Script Test step:****

[Groovy Script](http://www.groovy-lang.org/) test step is included for custom automation test script creation in SoapUI / Pro. It can be used for functional/ load/regression.

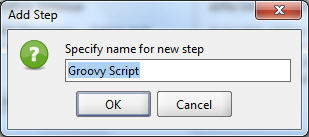
Groovy is a scripting language which internally includes all the java libraries, therefore all java related keywords and functions can be used in the groovy script directly. The Java libraries come with SoapUI and are integrated during the SoapUI Pro installation itself.

**Here is how Groovy script can be added to a test:**

**Step #1.** In SoapUI Pro create a [SOAP project with valid WSDL document](http://www.softwaretestinghelp.com/soapui-tutorial-4-working-with-soapui-projects/). Under the project, create a test suite with a desired name. Inside the test suite, add groovy script test step as shown below:

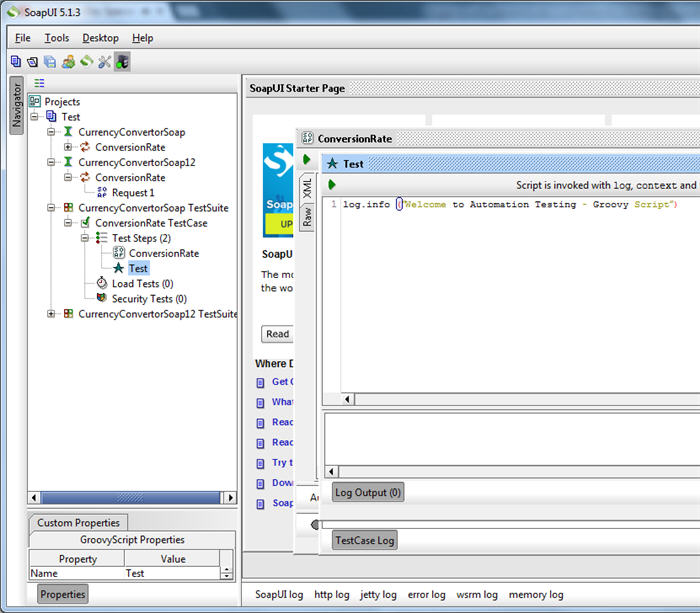
[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/Working-with-Operators-1.jpg)

**Step #2.** Enter the name of the step in the dialog that comes up as below and Click OK

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/Working-with-Operators-2.jpg)

**Step #3.** An editor where you can write your script is displayed. Alternately you can double click on the groovy step name from your test case (Groovy step is the one that has a star prefix to it).

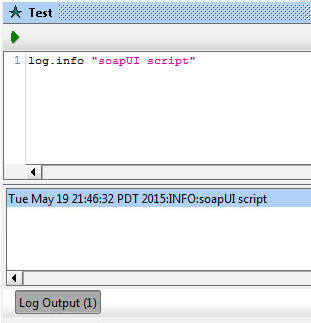
(Click on image for enlarged view)

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/Working-with-operators-66.jpg)

**For example**: Let us write a simple script that shows a message in the log. Here is the one line script.

**log.info ”soapUI script”**

**Step #4.** To execute the above script in SoapUI Pro, click on the Run icon and see the results in the Log Output section.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/Working-with-operators-44.jpg)

**Few points:**

* Test script execution: When the run button inside the groovy editor is clicked, the code inside the groovy step will only get executed. On the other hand, when the Run button is clicked for the entire test case, all the steps are executed in an order.
* This way any kind of programming can be done to the test scripts to add validations as required.
* There can be any number of groovy test steps to a test case.
* With Groovy script it is not required to compile and interpret separately to execute the code like other programming languages such as C, C++, Java, etc.
* Steps can be enabled or disabled inside a test suite by using the comment feature. To do so, use the following:

**//** – indicates single line comment and  
**/\* <some script> \*/** – denotes multi-line comment

------------

### Arithmetic Operations:

In the groovy step editor all of the below can be performed:

/\* Adding Two numbers \*/  
int a;  
int b;  
int c;

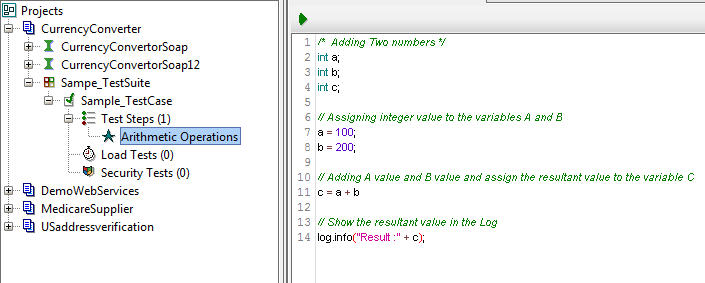
// Assigning integer value to the variables A and B  
a = 100;  
b = 200;

// Adding A value and B value and assign the resultant value to the variable C  
c = a + b

// Show the resultant value in the Log  
log.info(“Result :” + c);

In the above script, A, B and C are the variables which are used to store or transfer the values.

(Click on image for enlarged view)

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/Working-with-operators-55.jpg)

**Note:** Variables in Groovy script are case sensitive. Exercise caution when using them.

The following are the operators supported in Groovy:

**Arithmetic Operators:**

**+**   Addition operator / String concatenation  
**–**    Subtraction operator  
**\***   Multiplication operator  
**/**   Division operator  
**%**   Remainder operator

// Arithmetic Operators Samples

// Addition Operator  
int x1 = 100 + 200  
log.info (“Addition Result :” + x1);

// Concatenation of Two Strings using PLUS ( + ) operator  
String city =”Timothy E.” + ” Shepherd”;  
log.info(“String Concatenation:” + city);

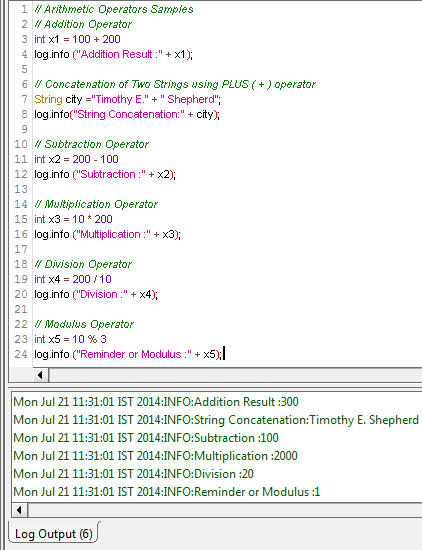
// Subtraction Operator  
int x2 = 200 – 100  
log.info (“Subtraction :” + x2);

// Multiplication Operator  
int x3 = 10 \* 200  
log.info (“Multiplication :” + x3);

// Division Operator  
int x4 = 200 / 10  
log.info (“Division :” + x4);

// Modulus Operator  
int x5 = 10 % 3  
log.info (“Reminder or Modulus:” + x5);

The following is a screenshot of all of the above scripts and the respective results:

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/Working-with-Operators-6.jpg)

**Unary Operators:**

Unary operators are the ones that work with only one operand. For example: **++**– it is called as **Increment operator** which increments the current value by 1

**Here’s the example:**

int A = 100;  
A++;                     // Equivalent to A = A + 1  
log.info (A);

The above script will produce the output as 101. This increment operation is called post increment. Similarly we can use this operator as a pre-increment operation as below:

int A = 100;  
log.info (++A);

There is also (–) the decrement operator. It will decrease the current value by **1**. We can implement this operator to the above discussed examples.

int A = 100;  
A–;                       // Equivalent to A = A – 1  
log.info (A);

The above script will produce the following output:  
Mon Jul 21 18:02:16 IST 2014:INFO:99

The pre and post operations can be used with decrement operator as well.

**Assignment Operators:**

The basic assignment operator is equal sign (=). Likewise, there are other useful assignment operators available. They are +=, -=, \*=, /=, %=.

Let us see the samples.

int A=100;  
A += 10;                // Similar to A = A + 10  
log.info(A);

The above script produces 110. If we use minus equal to operator in the below script, output will be 40.

int B=50;  
B -= 10;  
log.info(B);

Likewise we can use the remaining operators like this.

int C=10;  
C \*= 10;  
log.info(C);

And,

int D=50;  
D /= 10;  
log.info(D);

Here’s the reminder operator is used as

int E=10;  
E %= 3;  
log.info(E);

This will divide the value 10 by 3 and the remainder will be assigned to the variable “E”.

### ****Conclusion:****

This is just a start and there are many other operators available and supported by groovy such as logical, comparison, conditional etc.