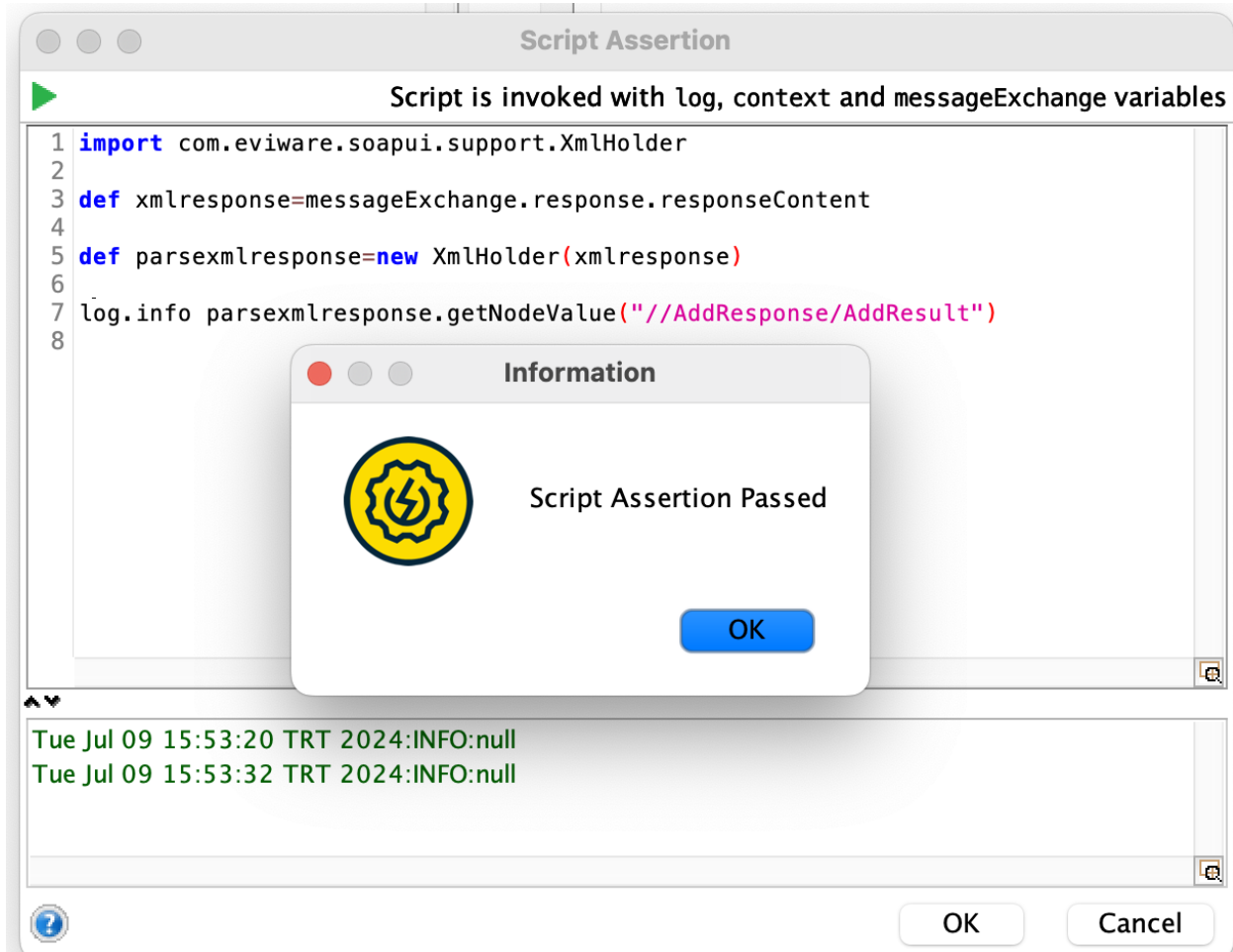
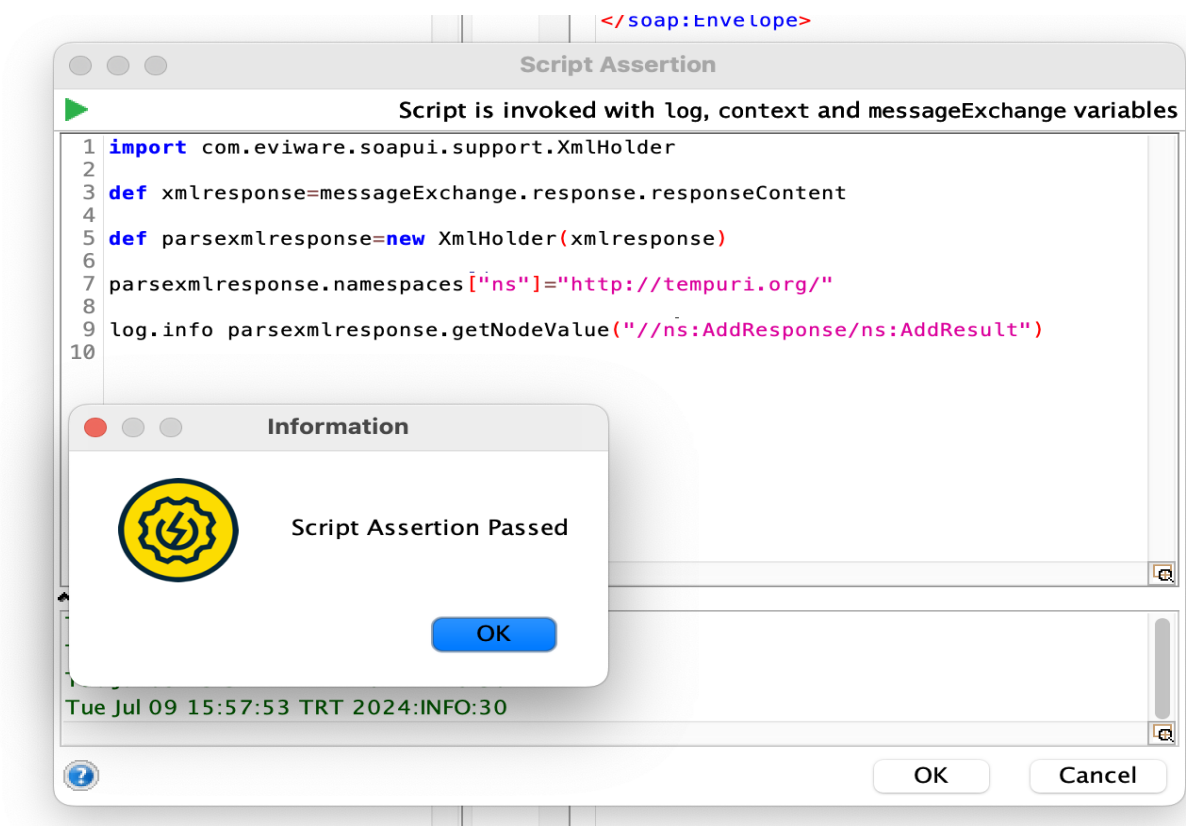
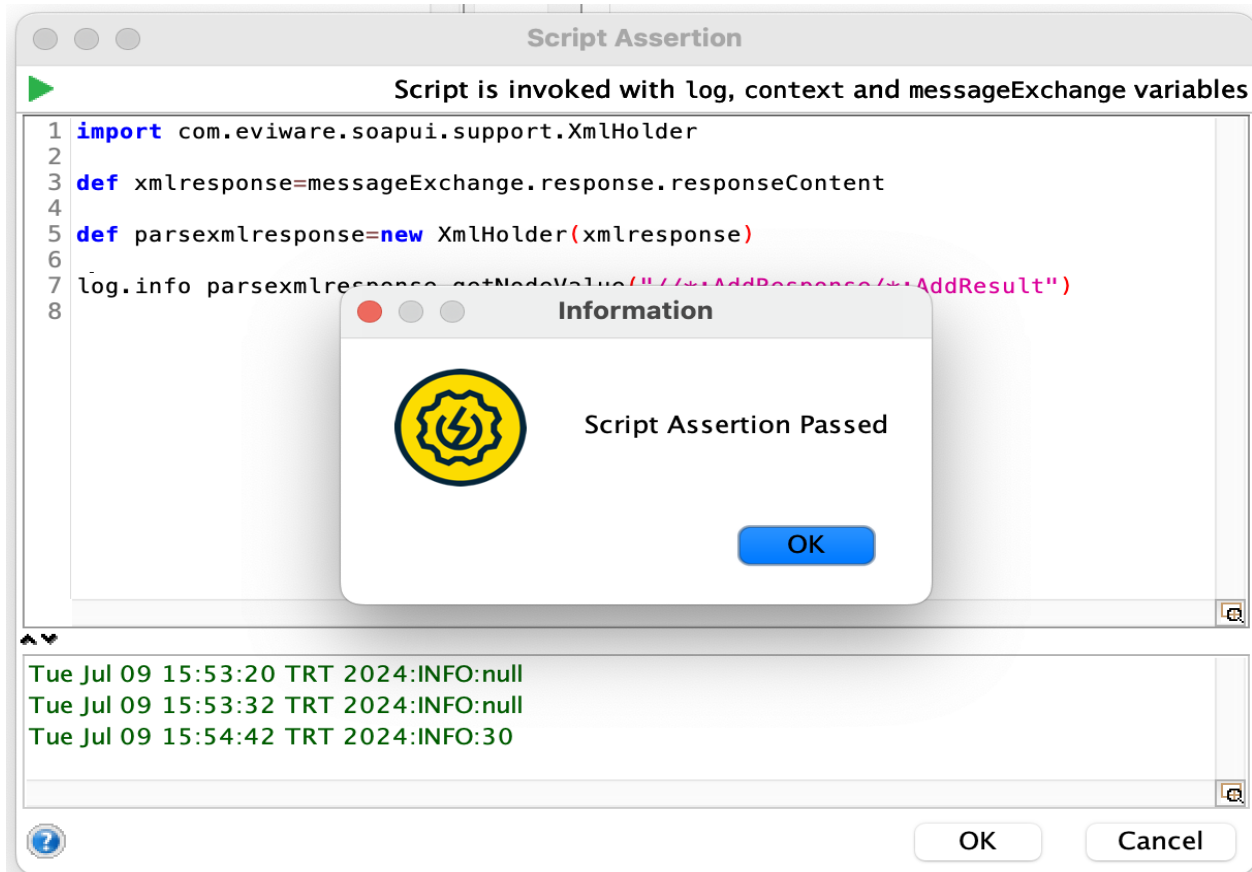


Assertions : Script Assertion for XML Response

The Script-Assertion allows for arbitrary validation of a message, which included message content, HTTP headers, etc. Validation scripts are written in the project scripting language (Groovy or JavaScript) and are executed whenever the assertion-status of the containing sampler TestStep is updated.

During the test development in SoapUI, there will be specific scenarios when the test assertions using the SoapUI are not sufficient enough for all the validations. Therefore, to handle such situations, SoapUI provides the capabilities to implement the validations using Groovy Scripts.





- CalculatorSoap TestSuite
 - ✓ Add TestCase
 - Test Steps (1)
 - Add
 - Load Tests (0)
 - Security Tests (0)


Custom Properties	
Name	Value
expsum	30

Script Assertion

Script is invoked with log, context and messageExchange variables

```
1 import com.eviware.soapui.support.XmlHolder
2
3 def xmlresponse=messageExchange.response.responseContent
4
5 def parsexmlresponse=new XmlHolder(xmlresponse)
6
7 parsexmlresponse.namespaces["ns"]="http://tempuri.org/"
8
9 def actsum=parsexmlresponse.getNodeValue("//ns:AddResponse/ns:AddResult")
10
11 log.info context.getTestCase().getPropertyValue("expsum")
```

Information

 Script Assertion Passed

OK

Tue Jul 09 16:05:48 TRT 2024:INFO:30

?

OK

Cancel

Raw XML view of a SOAP request and response. The request is for the URL `http://www.dneonline.com/calculator.asmx` and contains an `Add` operation with inputs 10 and 20. The response shows an `AddResult` of 30.

```
<?xml version='1.0' encoding='utf-8'>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header/>
  <soap:Body>
    <tem:Add>
      <tem:intA>10</tem:intA>
      <tem:intB>20</tem:intB>
    </tem:Add>
  </soap:Body>
</soap:Envelope>
```

```
<?xml version='1.0' encoding='utf-8'>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://schemas.xmlsoap.org/XMLSchema-instance">
  <soap:Body>
    <AddResponse xmlns="http://tempuri.org/">
      <AddResult>30</AddResult>
    </AddResponse>
  </soap:Body>
</soap:Envelope>
```

Script Assertion - VALID

Raw XML view of a SOAP request and response. The request is for the URL `http://www.dneonline.com/calculator.asmx` and contains an `Add` operation with inputs 100 and 200. The response shows an `AddResult` of 300.

```
<?xml version='1.0' encoding='utf-8'>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header/>
  <soap:Body>
    <tem:Add>
      <tem:intA>100</tem:intA>
      <tem:intB>200</tem:intB>
    </tem:Add>
  </soap:Body>
</soap:Envelope>
```

```
<?xml version='1.0' encoding='utf-8'>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://schemas.xmlsoap.org/XMLSchema-instance">
  <soap:Body>
    <AddResponse xmlns="http://tempuri.org/">
      <AddResult>300</AddResult>
    </AddResponse>
  </soap:Body>
</soap:Envelope>
```

Script Assertion - FAILED

-> assert actsum==expsum | | '300' | '30' false

Projects

- Football Soap
- calculator
 - CalculatorSoap
 - Add
 - Divide
 - Multiply
 - Subtract
 - CalculatorSoap TestSuite
 - Add TestCase
 - Test Steps (1)
 - Add
 - Load Tests (0)
 - Security Tests (0)

Raw

http://www.dneonline.com/calculator.asmx

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <tem:Add>
      <tem:intA>10</tem:intA>
      <tem:intB>20</tem:intB>
    </tem:Add>
  </soapenv:Body>
</soapenv:Envelope>
```

Script Assertion

Script is invoked with log, context and messageExchange variables

```
1 import com.eviware.soapui.support.XmlHolder
2
3 def xmlresponse=messageExchange.response.responseContent
4
5 def parsexmlresponse=new XmlHolder(xmlresponse)
6
7 parsexmlresponse.namespaces["ns"]="http://tempuri.org/"
8
9
10 def actsum=parsexmlresponse.getNodeValue("//ns:AddResponse/ns:AddResult")
11
12 def expsum=context.getTestCase().getPropertyValue("expsum")
13
14 assert actsum==expsum
15
```

Script Assertion - VALID

Assertions (1) Request Log (7)

response time: 215ms (326 bytes)

Custom Properties

Name	Value
expsum	30

Projects

- Football Soap
- calculator
 - CalculatorSoap
 - Add
 - Divide
 - Multiply
 - Subtract
 - CalculatorSoap TestSuite
 - Add TestCase
 - Test Steps (1)
 - Add
 - Load Tests (0)
 - Security Tests (0)

Raw

http://www.dneonline.com/calculator.asmx

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <tem:Add>
      <tem:intA>100</tem:intA>
      <tem:intB>200</tem:intB>
    </tem:Add>
  </soapenv:Body>
</soapenv:Envelope>
```

Script Assertion

Script is invoked with log, context and messageExchange variables

```
1 import com.eviware.soapui.support.XmlHolder
2
3 def xmlresponse=messageExchange.response.responseContent
4
5 def parsexmlresponse=new XmlHolder(xmlresponse)
6
7 parsexmlresponse.namespaces["ns"]="http://tempuri.org/"
8
9
10 def actsum=parsexmlresponse.getNodeValue("//ns:AddResponse/ns:AddResult")
11
12 def expsum=context.getTestCase().getPropertyValue("expsum")
13
14 assert actsum==expsum
15
```

Script Assertion - FAILED

assert actsum==expsum | | '300' | '30' false

Assertions (1) Request Log (8)

response time: 229ms (327 bytes)

Custom Properties

Name	Value
expsum	30

How to check the existence of a string in the SOAP response?

```
import com.eviware.soapui.support.XmlHolder

def holder = new XmlHolder( messageExchange.responseContentAsXml )
def responseXml = holder.getPrettyXml()
assert responseXml.contains('Richard E. Silverman')
```

Let's understand the above code snippet in detail:

1. The first line imports the required library *XmlHolder*, which parses *XML* in *SoapUI*.
2. We then initialize an *XmlHolder* object and pass the response of the service as a parameter to it as shown below:

```
def holder = new XmlHolder( messageExchange.responseContentAsXml )
```

messageExchange.responseContentAsXml returns the response of the *SOAP* call in *XML* format.

3. Next, we need to convert the XML response to the string, and we can achieve the same can be achieved with the help of *getPrettyXml()* as shown below:

```
def responseXml = holder.getPrettyXml()
```

4. Then we can use the Groovy Script's *assert* method to check for existence if the string value in the response.

```
assert responseXml.contains('Richard E. Silverman')
```

We can evaluate the 'assert' expression as a boolean expression. If the value of the expression returns *false*, then the script assertion fails and vice-versa.

How to validate that a string is not present in the SOAP response?

```
import com.eviware.soapui.support.XmlHolder

def holder = new XmlHolder( messageExchange.responseContentAsXml )
def responseXml = holder.getPrettyXml()
assert !responseXml.contains('Mark Twain')
```

As we can see, the above code also looks for the string 'Mark Twain' using the contains the method of the string. The only difference we will find is the *logical, not operator (!)* used before the *contains()* method. It will negate the result and will pass the assertion only when the expected string is not present in the response.

How to validate the value of a node in the SOAP response?

```
import com.eviware.soapui.support.XmlHolder

def holder = new XmlHolder( messageExchange.responseContentAsXml )
def responseXml = holder.getPrettyXml()
def title = holder.getNodeValue('//ns1:BooksResult/ns1:Books/ns1:CustomBookModel[2]/
assert title.equals('Learning JavaScript Design Patterns')
```

Let's understand the above code snippet in more detail:

Apart from using the other methods of the Xmlholder class, we have used the "*getNodeValue*" method to get the value of the specific node, as shown below:

```
def title = holder.getNodeValue('//ns1:BooksResult/ns1:Books/ns1:CustomBookModel[2]/
```

Here, we have mentioned the *XPATH* of the title of the second book. The '*ns1*' refers to the namespace in the XML response. The first namespace in the XML is referred to as '*ns1*' by default. If there would have been a second namespace, then it would have been '*ns2*' and so on.

We have used '*ns1: CustomBookModel[2]*' to get the title of the second book. Pay attention that the first element of the list starts at index 1 and not index 0.

And then we have used the equals() to validate the value of the title as shown below:

```
title.equals()
```

How to validate the headers of the SOAP response?

The below code snippet validates the header '*#status#*', which contains the response code of bookstore API.

```
def status = messageExchange.responseHeaders["#status#"]  
assert status.toString().equals('[HTTP/1.1 200 OK]')
```

As we can see, the *responseHeaders* method of *messageExchange* interface fetches response headers. Once, we have all the headers; we can put any assertions for validating the values, as in the above case, we converted it to string and checked the value to be equal to "[HTTP/1.1 200 OK]"

How to validate the response time of the SOAP response?

We can also use the script assertions to validate the response time of the service as well. As we know, *response time* is the total time it takes to respond to the request of the service. Let's see how we can do this using scripts in SoapUI:

```
assert messageExchange.timeTaken < 900
```

The *messageExchange.timeTaken* is the property used to get the *response time* of the *SOAP* request.

How to validate the values for a range in the SOAP response?

Suppose we need to check whether the pages of our book are within range of 200-300 pages. We can use the following script assertion to validate the same:

```
import com.eviware.soapui.support.XmlHolder

def holder = new XmlHolder( messageExchange.responseContentAsXml )
def responseXml = holder.getPrettyXml()
def pages = holder.getNodeValue('//ns1:BooksResult/ns1:Books/ns1:CustomBookModel[2]/
log.info pages
assert pages.toInteger() > 200 && pages.toInteger() < 300
```

Let's understand what we trying to do here is:

1. We have got all the pages in the XML response using the "*getNodeValue*" method for the "*Pages*" as shown below:

```
def pages = holder.getNodeValue('//ns1:BooksResult/ns1:Books/ns1:CustomBookModel[2]/
```

2. Then we converted the value to "*integer*" using the "*toInteger*" method and compared with values 200 and 300 as shown below:

```
assert pages.toInteger() > 200 && pages.toInteger() < 300
```

So, we can retrieve any value from the XML response and validate it against any expected values using the script assertions in SoapUI.