**Understanding Assertions in SoapUI :**

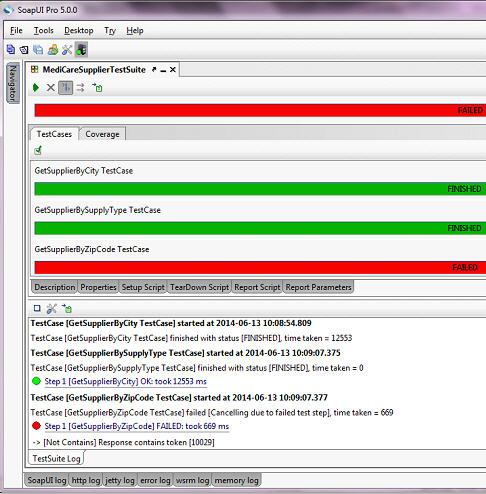
**Introduction to Assertions:**

As with any testing, we have to compare what we want the system to do and what it actually is doing, to arrive at a certain validation or assertion, which is what it is called in the context of web services. As testers, it does not matter if we executed 1000 or even million test steps, but for us the result comparison is what determines the outcome of a test.

Therefore, we will spend this entire article on understanding how we can do that with SoapUI, although web services can be asserted manually. Also manual assertion is time consuming when there are multiple responses and responses with large data. SoapUI assertions are excellent at overcoming these shortfalls.

**SOAPUI Assertions** compare the parts/all of the response message to the expected outcome. We can add a variety of assertions provided by SoapUI to any test step. Each type of assertion targets specific validations on the response such as matching text, comparing XPATH or we could also write queries based on our need.

When the test steps get executed, then the associated assertions receive the response for the respective test steps. If any response is failed then the respective assertion will be processed and the corresponding test step will be marked as failed. This notification can be viewed in the test case view. Also we can find failed test steps in the test execution log. The sample test step assertion screen looks as below:

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

In the above image, some of the test steps have FAILED and some of them have PASSED. The reason is assertion.

As we discussed earlier, if the assert condition is not met with the expected results then the result is FAILED.

### Working with different kinds of assertions in SoapUI:

Let us now see how to work with different types of assertions like:

* Contains and Not Contains assertions
* XPath match and
* XQuery match assertions.

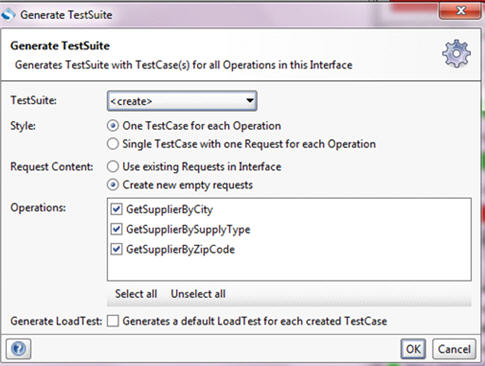
Firstly, we need a valid WSDL schema location. For that, we can use Medicare Supplier web service that is available at <http://www.webservicex.net/medicareSupplier.asmx?WSDL>.

**Follow the steps as below:**

**Step 1.** [Create a new SOAP project](http://www.softwaretestinghelp.com/soapui-tutorial-4-working-with-soapui-projects/) by pressing CTRL + N and follow the steps. After creating the project, SOAPUI generates the list of interfaces and the corresponding requests.

**Step 2.** To add test suite to this project follow these steps:

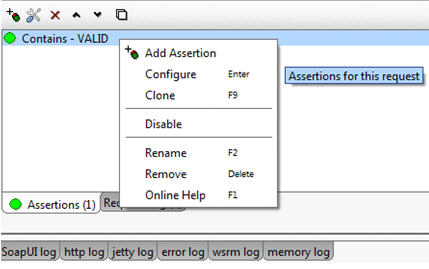
* Right click on the interface name **MedicareSupplierSoap**
* Click **Generate Testsuite**option from the context menu
* Click OK on the below window that comes up:

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

* In the next popup, need to enter your desired test suite name and click OK
* SOAPUI PRO will generate the test suite along with the requests in the navigator panel.
* Under the test suite, you will see some of the test steps with SOAP request step.

**Step 3.** To execute this test suite, double click on the request step and specify the input value in the respective location. For instance, open **GetSupplierByCity** request and enter **New York** between the city tags.

* Start this request by clicking on RUN icon – this will receive the response
* Now let’s add assertions. For that, click **Assertions** tab present at the top of log tabs.
* On right clicking, a popup menu will appear with some basic assertion related options as below:

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/SOAPUI-Assertions-3.jpg)

### #1. Contains Assertion:

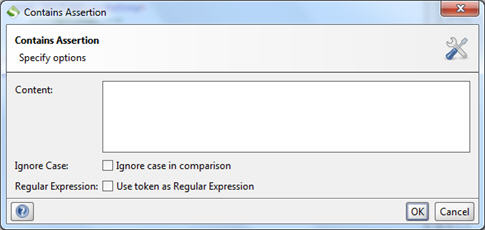
Click Add Assertion option or click it from the tool bar – **Add Assertion** window appears on the screen with different types of assertions.

1. Click **Property Content**category from the list – associated assertion types and their description is displayed

2. Click **Contains**assertion and click Add button

3. This is the assertion configuration window. Here itself we have to specify the expected condition based on the response.

**For exampl**e, let me enter**New York**text in this text field. **Ignore case in comparison**checkbox will ignore even if the expected value is in upper case or lower case.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/SOAPUI-Assertions-4.jpg)

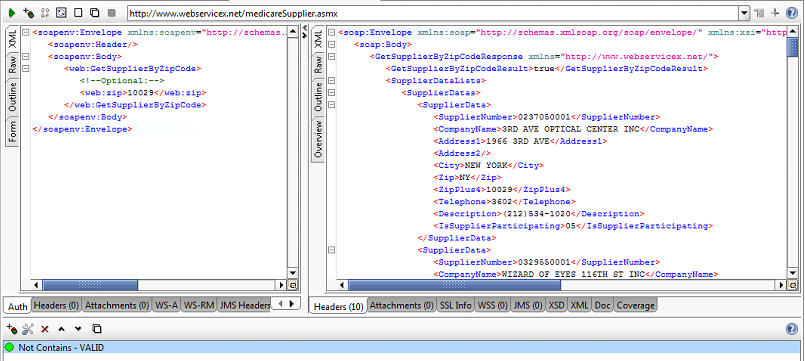
4. Now execute the test suite and verify the results. As you have seen in the test suite window, green indicates the successful execution and red denotes failure.

### #2. Not Contains Assertion

We can use “not contains” assertion for validating request in negative scenarios. We can use **GetSupplierByZipCode** request to learn that.

Open the request tab by double clicking on it. In the input request, enter invalid zip code in the appropriate location, such as **10029**. Run this request now. Check the response data which has the supplier details for the given zip code – take a look at the image below:

***(Click on image for enlarged view)***

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/SOAPUI-Assertions-5.jpg)

The “not contains” assertion, it is highlighted in green color as it is executed successfully.

In the configuration window, we have configured with positive expected value as below:

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

It returns true if the expected conditional value is not found and returns false if the expected value is found in the response message.

Similarly we can change the condition and run the request once again. It generates the results accordingly.

### #3. XPath Match assertion

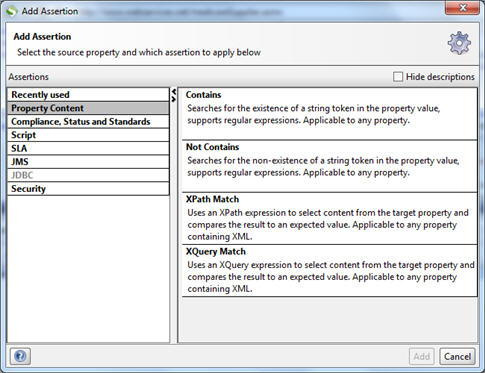
XPath match assertion is little bit different in terms that it will assert the response using actual response data.

**For example**, if we have login authentication web service that will authenticate the user credentials and send the acknowledgement to the client with some Boolean type of data which may be TRUE or FALSE in the form of XML.

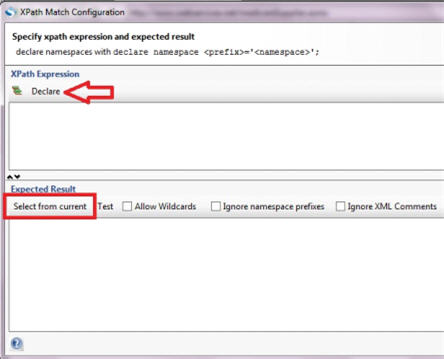
As you know XML documents are built by tags. So when specifying the expected value in the configuration, it should be in the form of XML.

**Let try to do that:**

Add one more assertion for the **GetSupplierByCity** request. In the Add Assertion window, click **Property Content**category and then click **xpath match**assertion.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/SOAPUI-Assertions-7.jpg)

**The below window is displayed:**

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/SOAPUI-Assertions-8.jpg)

The top section is the declaration part and bottom section is expected result part.

When we click on the **Declare**option we will get some auto generated declaration scripts like below:

**declare namespace soap=’http://schemas.xmlsoap.org/soap/envelope/’;**  
**declare namespace ns1=’http://www.webservicex.net/’;**

In the above scripts, first line denotes the response that should be XML data and enclosed SOAP tags. In the next line, the whole response will be assigned or copied into the **ns1**namespace variable during the execution. If we want to filter particular data from the whole response, we have to add the following script.

**//ns1:SupplierData[1]**

As you know, if you execute **GetSupplierByCity** request, it will produce the response which contains the list of supplier’s personal data that belongs to **New York City**. Here, we have used **Xpath Match** expression to extract the specific supplier’s personal details from the bulk response. For that purpose we have used **ns1**variable. Now click on the **Select from Current**button.

Then SOAPUI generates the following result:

|  |  |
| --- | --- |
| 1 | <SupplierData xmlns="http://www.webservicex.net/" |
| 2 | xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" | |

|  |  |  |
| --- | --- | --- |
| 3 | xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" | |
| 4 | xmlns:xsd="http://www.w3.org/2001/XMLSchema"> |

|  |  |
| --- | --- |
| 5 |  |
| 6 | <SupplierNumber>0127051883</SupplierNumber> | |

|  |  |  |
| --- | --- | --- |
| 7 | <CompanyName>KMART CORP</CompanyName> | |
| 8 | <Address1>250 W 34TH ST</Address1> |

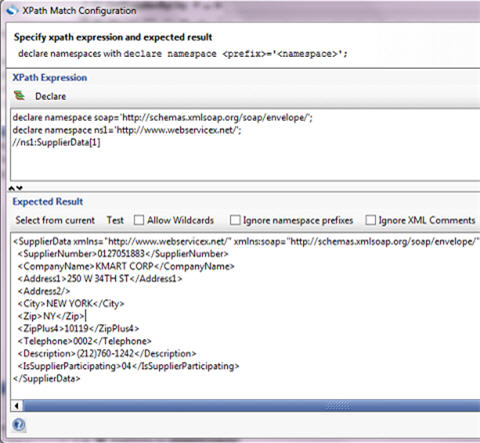
|  |  |  |
| --- | --- | --- |
| 9 | <Address2/> | |
| 10 | | <City>NEW YORK</City> | |

|  |  |
| --- | --- |
| 11 | <Zip>NY</Zip> |
| 12 | <ZipPlus4>10119</ZipPlus4> | |

|  |  |
| --- | --- |
| 13 | <Telephone>0002</Telephone> |
| 14 | <Description>(212)760-1242</Description> | |

|  |  |  |
| --- | --- | --- |
| 15 | <IsSupplierParticipating>04</IsSupplierParticipating> | |
| 16 | </SupplierData> |

**Please refer this screenshot:**

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/SOAPUI-Assertions-9.jpg)

Here in the response data, you can see only one supplier personal data. Based on the number that is present inside the angle brackets, the output will be generated.

By so far, this is about choosing the portion of the response that is required, how can/are we using **Xpath Match** assertion?

**Let get to that:** Click on the Save button once you are OK with the response.

Initially if you would have executed this service after configuring xpath match assertion with no changes, the result will be a successful response, status highlighted in green.

But let’s change the input parameter in the input request to something that is invalid city -“XYZ or ABC”. Run the request and check the results as well as the assertion status. We will get failure response and red status indication for assertion. Because we had already specified that the particular supplier data should be present in the service response in the expected result configuration and when the city name is invalid, that supplier is clearly not present.

This is how we can assert the XML response using Xpath Match expression assertion. Agreed that this is quite simple to start with but if you try with different service responses, you will get a much better idea.

We can also use aggregate functions in XPath Match expression. They are Sum, Min, Max, Count and Avg.

**For example**, if we want to know the total number of suppliers count in the expected results, write the following script.

**count (//ns1:SupplierData)** and it returns **536** as a result. Remember all the aggregate functions should be in lowercase.

### #4. XQuery Match Assertion:

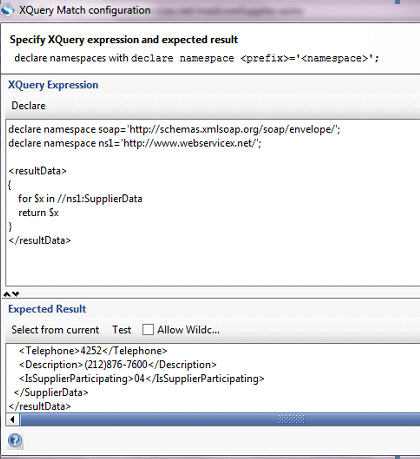
This is slightly similar to XPath Match assertion. As we have seen in the Xpath Match assertion configuration, there will be two sections – declaration and expected result.

* Add XQuery Match assertion for the request
* In the configuration window, click **declare** button and write the following script
* Now click **Select from Current**button
* SOAPUI generates the response for the script

XQuery expression also supports XPath Match expression but it has its own scripting syntax which cannot be used in XPath match assertion.

**For Example**:

We will see one example to fetch all the supplier data response using XQuery expression. Look at this sample screenshot to understand better.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/SOAPUI-Assertions-10.jpg)

**Actual Script:**

|  |  |  |
| --- | --- | --- |
| 1 | declare namespace soap='http://schemas.xmlsoap.org/soap/envelope/'; | |
| 2 | declare namespace ns1='http://www.webservicex.net/'; |

|  |  |
| --- | --- |
| 3 |  |
| 4 | <resultData> | |

|  |  |
| --- | --- |
| 5 | { |
| 6 | for $x in //ns1:SupplierData | |

|  |  |  |
| --- | --- | --- |
| 7 | return $x | |
| 8 | } |

|  |  |
| --- | --- |
| 9 | </resultData> |

There are more built-in functions available to use in XQuery expression. They are **where, order by, for, return**and so on. To get more information about these functions, visit [www.w3cschools.com](http://www.w3cschools.com/)website.

### Conclusion:

Well, those are the most commonly used assertions for you.

The point of emphasis here is that: Assertions are most important to predict the expected result of the web services accurately. That’s the reason it is one of the key [features in SOAPUI PRO](http://www.softwaretestinghelp.com/soapui-tutorial-2-important-features-of-soapui-and-soapui-pro/).