SoapUI

40+ Pages of Step-by-Step Tutorials

SoapUI 101

Beginner's Guide to Functional Testing

Table of Contents

Introduction

Why SoapUI?

If you aren't familiar with SoapUI, let's be-gin with a short int-roduction to the tool and what it does.

SoapUI is an API testing tool that offers a free, open source version as well as a fully featured pro version with extended func-tionality. With over 9 million downloads, it has become the de facto standard for func-tional testing of SOAP and REST APIs, allo-wing users to quickly and easily create auto-mated functional, regression, compliance, and security tests.

Functional testing

While SoapUI includes basic functional, load and security tes-ting as well as support for mocking, functional testing being one of the most common types of testing users spend time on.

Functional testing involves checking that the behavior of the system being tested matches the business expectations in terms of func-tionality. SoapUI empowers users with the abi-lity to perform functional testing easily.

SoapUI is used to send requests and spe-cify assertions to verify that the response being received is correct. It can also be used to define broken requests to verify that the application reacts as expected when it re-ceives unusual or faulty data. Once a test has been created, it can be reused for unit tes-ting, regression testing, security testing, or load testing, all made available through the commercial version of SoapUI in the Ready! API platform.

What is an API?

An Application Programming Interface, or API, specifies how some software compo-nents should interact with each other.[1]

While many people associate APIs with soft-ware libraries, the definition is broad enough to include Web Services and many other types of messaging including technologies that SoapUI supports: SOAP, REST, XML, JSON, MQTT, JMS, JDBC, AMF and HTTP(S).

What is a Web Service?

The W3C defines a "Web Service" as:

A Web Service is a software system designed to support interoperable machine-to-ma-chine interaction over a network. It has an interface described in a machine-readable format (specifically WSDL[sic]). Other sys-tems interact with the Web Service in a man-ner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML seriali-zation in conjunction with other Web-related standards.[2]

In other words, a Web Service is a service available on a network that will allow other systems to communicate with it using a defined protocol.

The "Web" part indicates that the service is using transport protocols designed for the World Wide Web, i.e., it uses HTTP to communicate. Other systems can be Web applications, smart phone apps and similar applications.

APIs come in two main flavors, SOAP and REST. SOAP being the earlier implementation of Web API uses XML for strict message formatting and always produces a WSDL that acts as a descriptor to the operations available on the service. Alternatively, REST (REpresentational State Transfer) is an architectural style for building scalable web services that provides guidelines on how messaging should occur. A "REST-ful" web service generally infers much less emphasis on strict formatting, and typically uses JSON (JavaScript Object Notation) formatted data in message bodies instead of XML, though much in a RESTful web service is left up to the designers to decide. Since both are considered "styles" of an API, we will simply use the abbreviation API to refer to the service henceforth.

1 http://en.wikipedia.org/wiki/Application_programming_interface

2 http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice

SOAP

A SOAP API is defined as a receiver of an XML document and is also expected to respond with an XML document. The documents should follow a format standardized by W3C.

All parameters that the receiver needs to be able to respond to should be a part of the XML document sent.

Testing a SOAP API means manipulating a request XML and interpreting a response XML.

REST

REST or RESTful APIs encode all parameters in the request. A RESTful service should never be stateful, in that the server should not keep any state about a client.

Testing a REST API means manipulating the request URI and interpreting the answer. The answer may be of many different formats. XML is possible, but formats like JSON or plain text are also possible.

Testing APIs

There are some initial challenges when testing an API. There is no graphical user interface and you must understand a response that wasn't created to be read by humans.

No graphical user interface

APIs are primarily intended for computer-to-computer communication. Computers don't need a user interface. This is a challenge if you're a tester who is accustomed to testing software designed for end-users, with a graphical interface you can click, drag-and-drop, collapse, close, and in general manipulate an application or data visually. The solution is to use testing tools that will assist you as much as possible in creating requests and viewing the responses, visualizing both ends of the communication.

SoapUI and SoapUI NG Pro are two tools that will make your life as a tester less comp-licated by providing graphical controls to enter your requests, assertions to validate your responses, various tools to create and maintain your tests, and reports to print and share with your colleagues.

The tools also give you a user interface to both call APIs and analyze answers from APIs. Hence the "UI" ending in SoapUI, meaning User Interface.

Request and response

SOAP-based APIs use XML as a communication protocol. It must provide an XML descriptor (WSDL), that in machine terms, clearly defines all types of request elements the SOAP API will accept, and all types of response elements it will send back. This means that you, as a tester, must be able to read, understand, create, and update XML documents to be able to test a SOAP API.

With REST-based services, the request can be much simpler, for the most part only involving sending a URL (or URI) to the service. The responses returned are usually in JSON or XML format. These are, again, not formatted to be consumed by human beings.

This means that testing the service may be a challenge without a good tool. Once again, SoapUI and SoapUI NG Pro make testing APIs easier by enabling you to make small changes to a request and get a response back from the server that you can validate. SoapUI will assist you with response validations, response format, and response times so that expected SLAs can be asserted on and tracked.

Exploring

A common situation when SoapUI is used is when an unknown API is explored.

With the API computer-to-computer communication, an API will act as the host and you need to have another piece of code to act as an API client. You can set up SoapUI as the API client.

This is very convenient and saves a lot of time. It's common among developers who need to write a client for an unknown service.

Exploring a service also applies to testers, as the increasing dominance of RESTful APIs tend to lack a service descriptor (like how SOAP comes with WSDL) to help understand the expected behaviors known and should be verified. The exploring and discovering has already been done.

Getting started

Running your first API test in SoapUI is easy. For first-time users, the process looks something like this:

- Set up SoapUI.
- Get started with your first project.
- Add a test suite.
- Add a test case.
- Add an assertion.
- Run a test suite.

Let's begin by installing SoapUI.

Installation

Download SoapUI from http://www.SoapUI.org/. Follow the download link to get the version you want. Both SoapUI and SoapUI NG Pro are available from the same location.

SoapUI is available for many different operating systems. Just follow the installation instructions for your operating system:

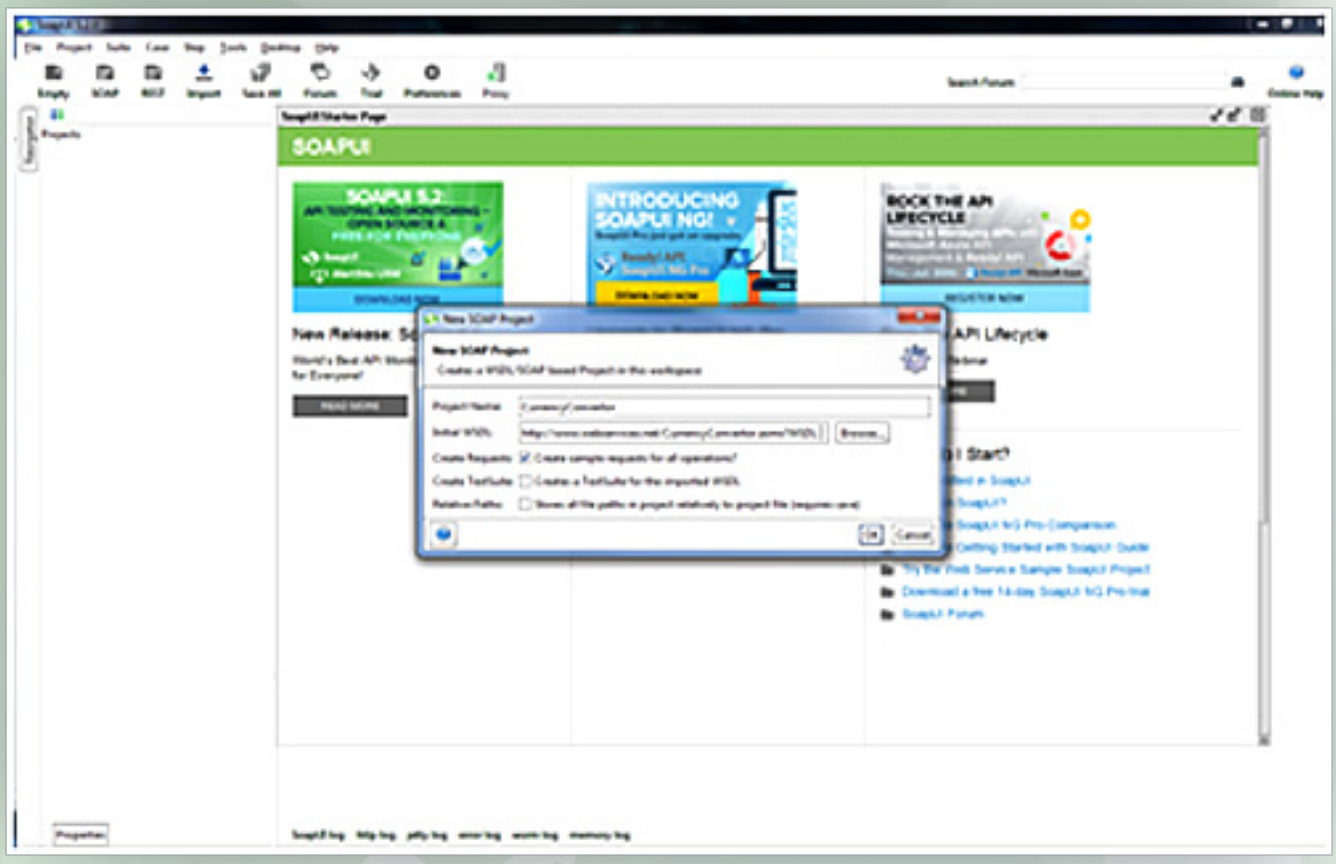
- Windows: www.SoapUI.org/Getting-Started/installing-on-windows.html
- Mac: www.SoapUI.org/Getting-Started/installing-on-mac.html
- Linux: www.SoapUI.org/Getting-Started/installing-on-linuxunix. html

Now that SoapUI is installed, let's open one of the sample projects.

Your first project

The web service we will use for this illustration is an API that will take two currencies and return the currency conversion. It has very few moving parts, so it will be a good starting point.

- Start SoapUI.
- Create a new project by clicking File | New SOAP Project.
- Paste the URL below in the field "Initial WSDL": http://www.webservicex.net/CurrencyConvertor.asmx?WSDL
- Give it a descriptive name. It defaults to "Currency Converter." Based on information from the WSDL definition.
- The checkbox, "Create Requests" is checked by default. Leave the defaults and click "OK." SoapUI will now call the API to get its details.

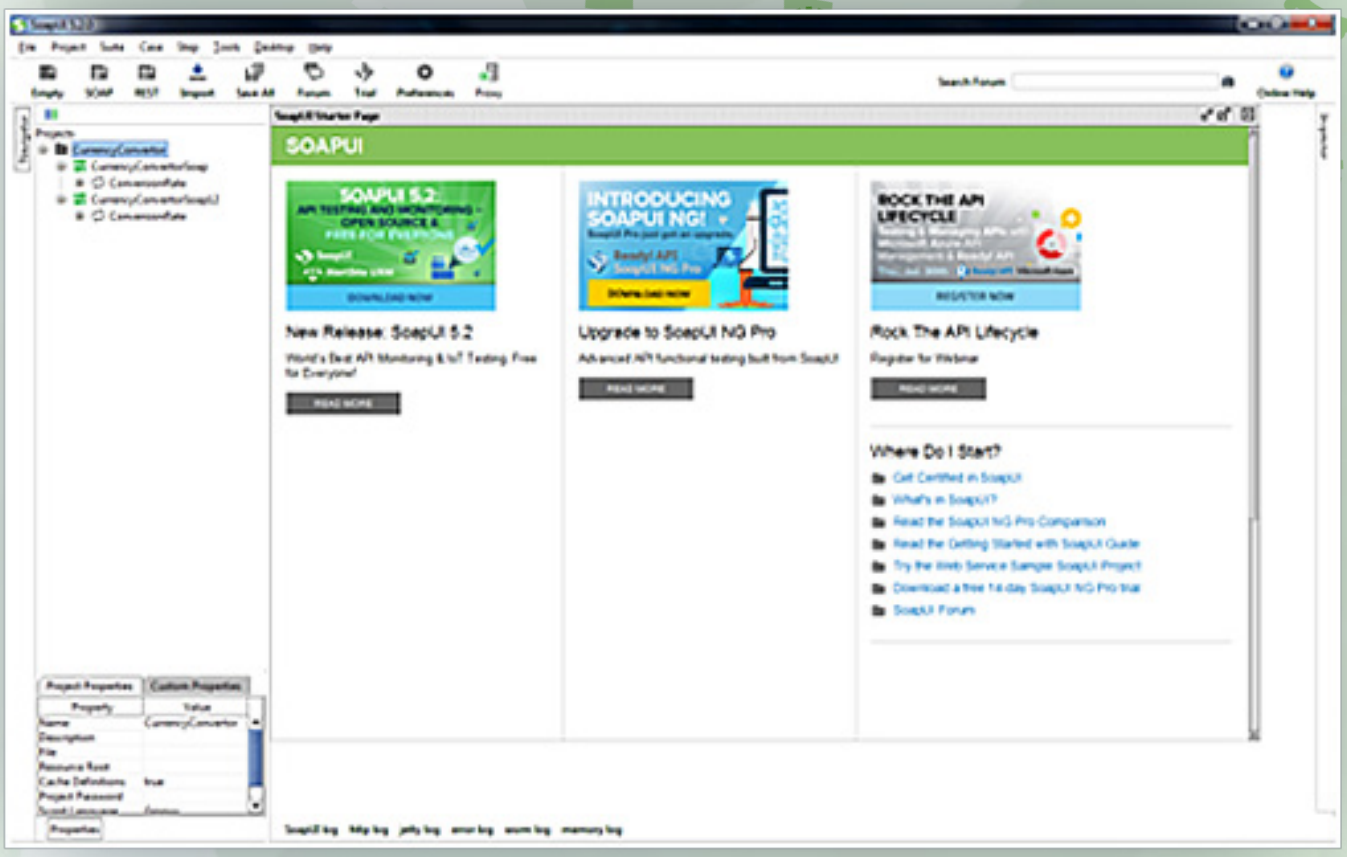


A new project should have shown up in the Navigator area at the top left section of SoapUI. There are two versions of the service:

"CurrencyConverterSoap" and "Currency Converter Soap12."

The difference between these is that the first one follows an older standard of SOAP compared to the latter one. Which one we use in this example doesn't matter, so I will remove "CurrencyConverterSoap12."

Select it and press the Delete button, or right click on "CurrencyConverterSoap12" and choose Remove.



Let's explore the remaining service. Expanding the tree to the left in the project explorer reveals four things:

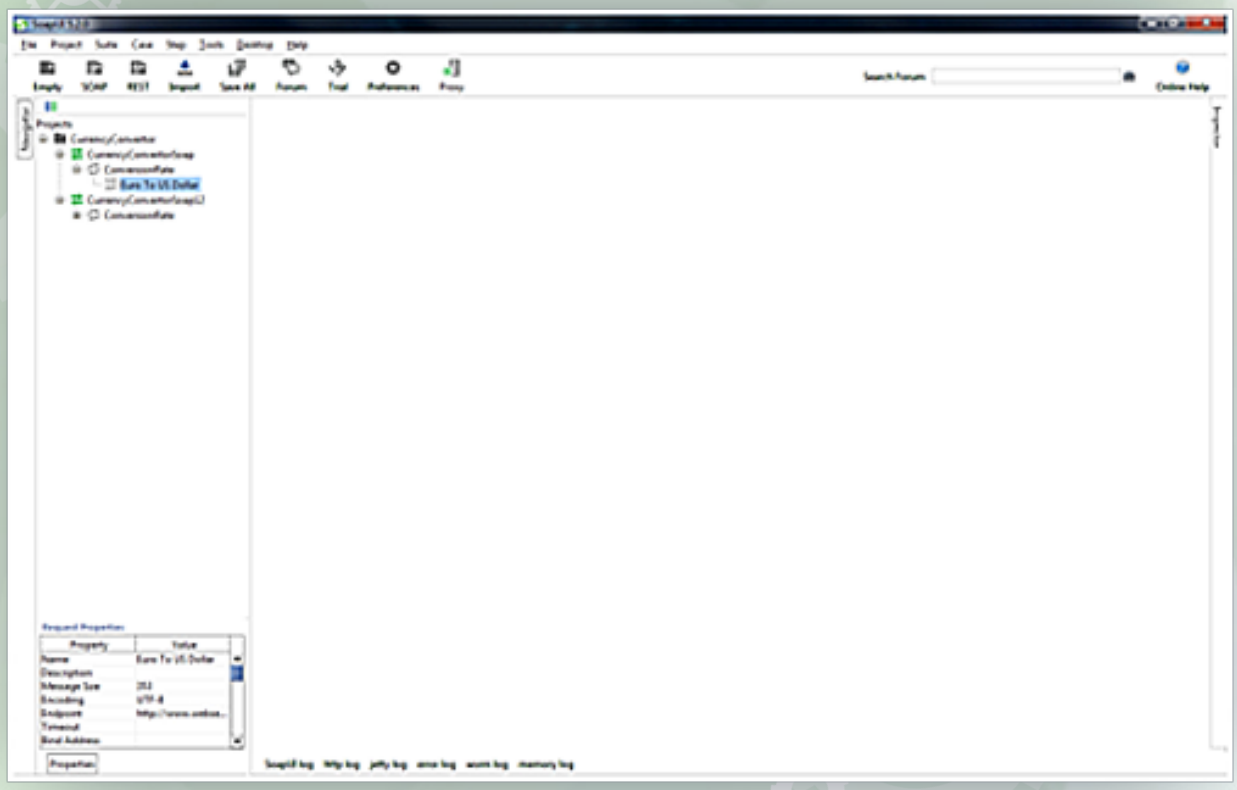
the project name, "CurrencyConverter," below it the name of the interface, "CurrencyConverterSoap," below it, the method name, "ConversionRate," and finally, a sample request to the service, called "Request 1."

"Request 1" is a name that SoapUI has generated for us. We want to change it to a name that is more descriptive.

I chose "Euro To US Dollar." Right click on "Request 1" and select Rename.

Double click on the “Euro To US Dollar” request. A sample request will be generated in the application window.

Requests can be viewed as XML or Raw data. The XML content should look like this:

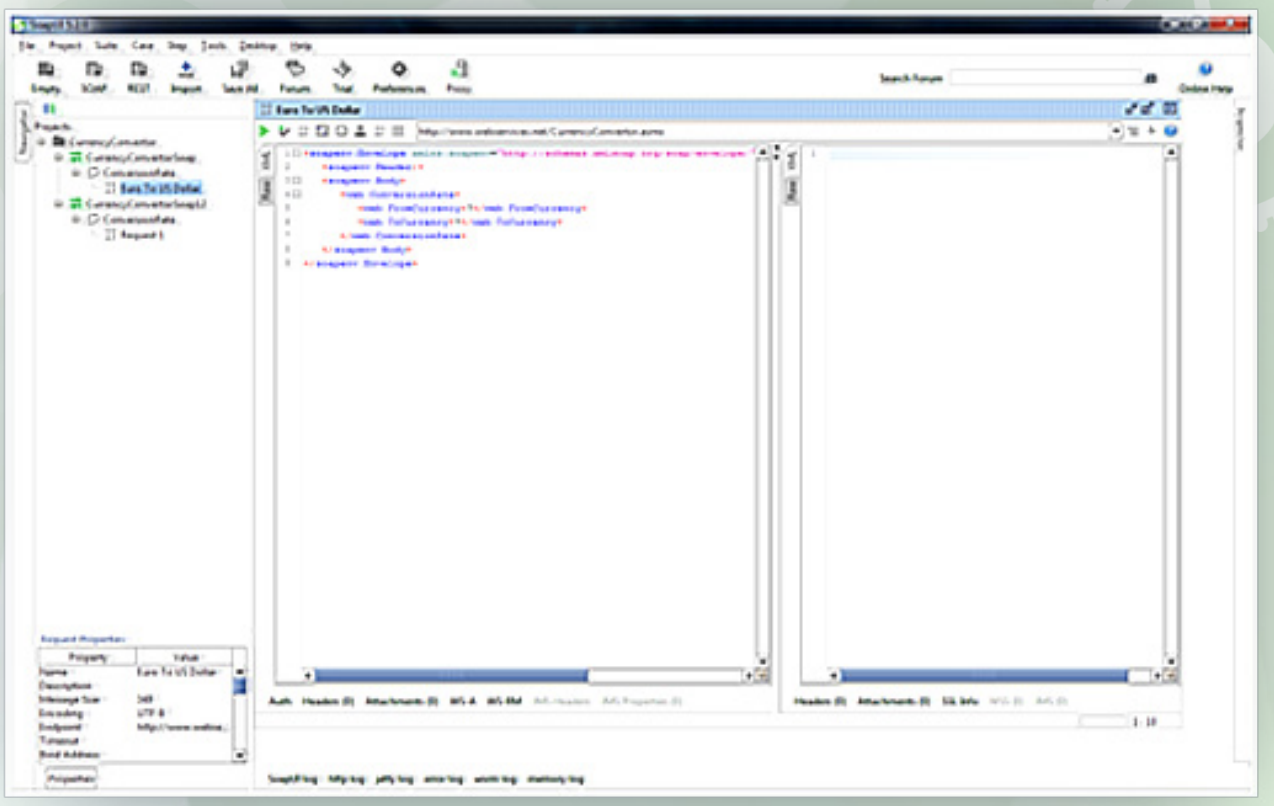


```
<soapenv:Envelope xmlns:soapenv=”http://schemas.xmlsoap.org/soap/envelope/”
xmlns:web=”http://www.webserviceX.NET/”>
<soapenv:Header/>
    <soapenv:Body>
        <web:ConversionRate>
            <web:FromCurrency>?</web:FromCurrency>
            <web:ToCurrency>?</web:ToCurrency>
        </web:ConversionRate>
    </soapenv:Body>
</soapenv:Envelope>
```

This is an XML message that will be sent to the server where the API runs. Sending it as it is will result in an error.

We need to change two things before we can send it and expect something useful in return. The two question marks are placeholders for two parameters that must be set properly.

The names of the para-meters are “From-Currency” and “ToCurrency.” Replace the question mark for “FromCurrency” with the symbol for Euro, EUR. Then replace the ques-tion mark for “ToCurrency” with the symbol for US Dollar, USD. The result should look like this:



```
<soapenv:Envelope xmlns:soapenv=”http://
schemas.xmlsoap.org/ soap/envelope/”
xmlns:web=”http://www.webserviceX.
NET/”>
<soapenv:Header/>
    <soapenv:Body>
        <web:ConversionRate>
            <web:FromCurrency>EUR</
web:FromCurrency>
            <web:ToCurrency>USD</
web:ToCurrency>
        </web:ConversionRate>
    </soapenv:Body>
</soapenv:Envelope>
```

We should get the conversion rate between Euro and US Dollar when we send this to the server.

The response will be a similar XML docu-ment, where we will locate an ele-ment that contains the answer. It will look like this:

```
<soap:Envelope xmlns:soap="http://
schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/
XMLSchema-in- stance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
    <soap:Body>
        <ConversionRateResponsexmlns="h
        ttp://www.webser-viceX.NET/">
            <ConversionRateResult>1.0997
            </ConversionRateResult>
        </ConversionRateResponse>
    </soap:Body>
</soap:Envelope>
```

The element "ConversionRateResult" contains the value 1.0997, which should be interpreted to mean that it will cost you 1.0997 US Dollars to get one Euro. I will add an assertion later that will verify this value.

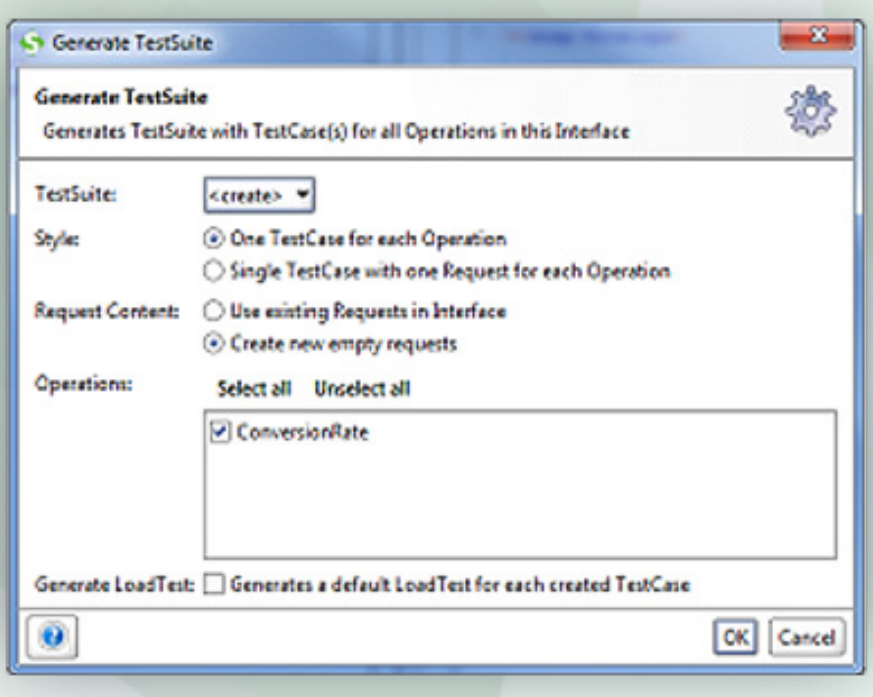
Adding an assertion in this case may be a bit risky, as the exchange rate between Euro and US Dollar isn't fixed. But let's ignore that problem for now.

Add a test suite

The sample request we made earlier should be included in a test suite that contains one or more test scenarios. Each scenario should also contain one or more assertions. We will exe-cute this test suite to do a functional test of an API.

Let's add the test suite.

- Right click on "CurrencyConverter Soap."
- Click on "Generate TestSuite."
- A dialog box will show up where you can customize the generation. Leave the de faults as they are and click "OK" to gene-rate a test suite.



Call the test suite “Euro to Dollar TestSuite” and click “OK.”

A test suite is generated for us. It has been populated with one test case. Let’s move on and have the test case do something useful.

Add a test

Expand the newly created test suite. A test case has been created for us. Our next step is to tweak it so we can verify the conversion rate.

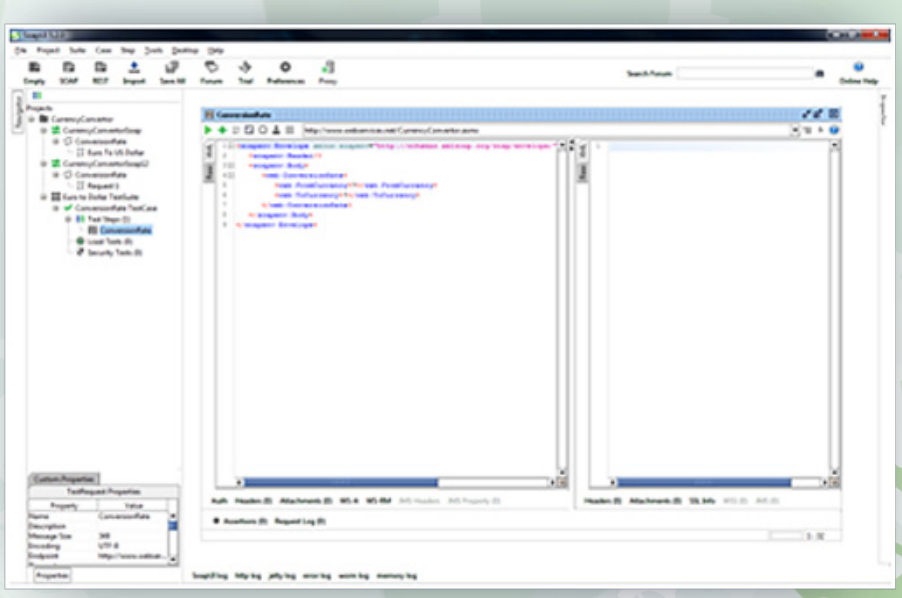
- Expand the tree until the test steps have been unfolded and the test step “ConversionRate” is visible.
- Double click on the test step. A sample request should appear in the request editor.

Replace the two question marks in the XML with EUR and USD. The result should look like this:

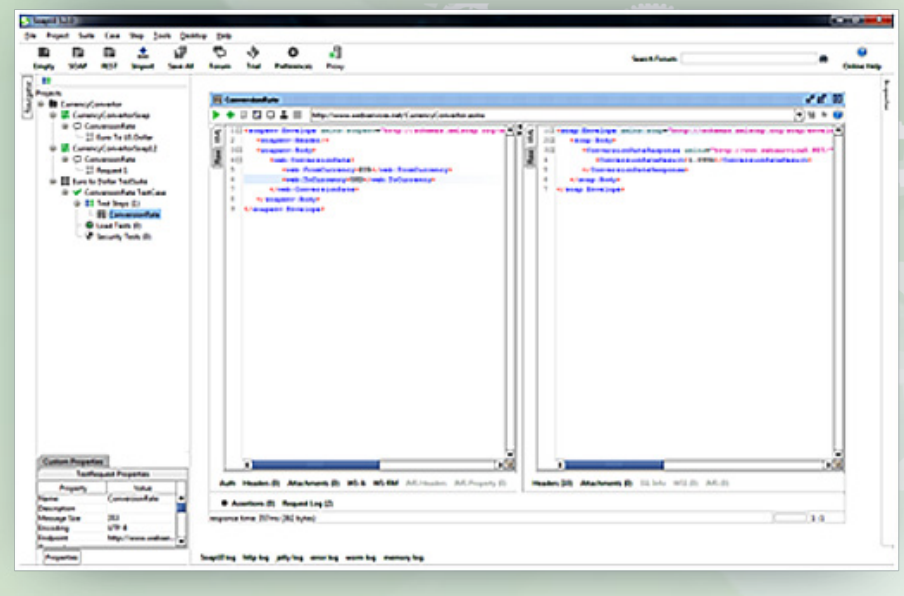
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/ soap/envelope/"
xmlns:web="http://www.WebserviceX.NET/">
<soapenv:Header/>
    <soapenv:Body>
        <web:ConversionRate>
            <web:FromCurrency>EUR</web:FromCurrency>
            <web:ToCurrency>USD</web:ToCurrency>
        </web:ConversionRate>
    </soapenv:Body>
</soapenv:Envelope>
```

- The “web:FromCurrency” tag should contain EUR and the “web:ToCurrency” should contain USD.

This is all we need to do to create our first test case. We can execute this test case now. Click on the green “Play” arrow to execute the test.



The result after an execution will look similar to this:

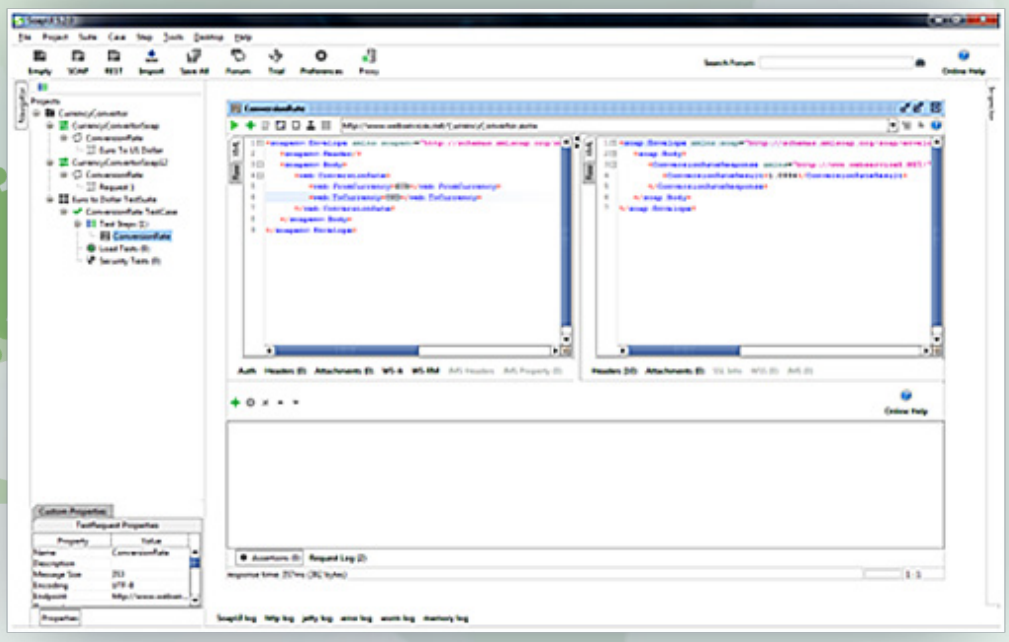


Unfortunately, we will not get any indication if it failed or passed if we execute it now. We need to add at least one assertion that will warn us if there was a problem that failed this test. Let's do this as the next step.

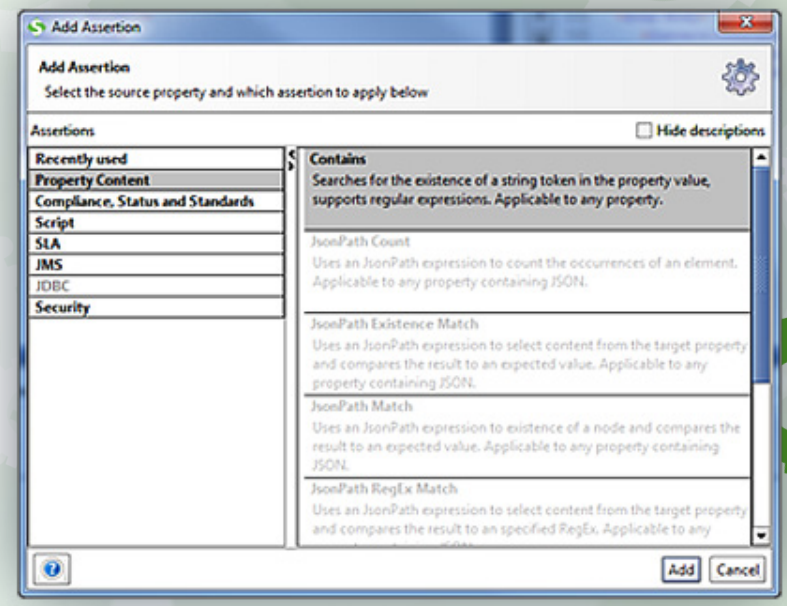
Add an assertion

Add our first assertion.

- Click on the "Assertions" tab at the bot tom of the request editor.
- This will expand the assertions editor. There aren't any assertions right now. We'll add them next.

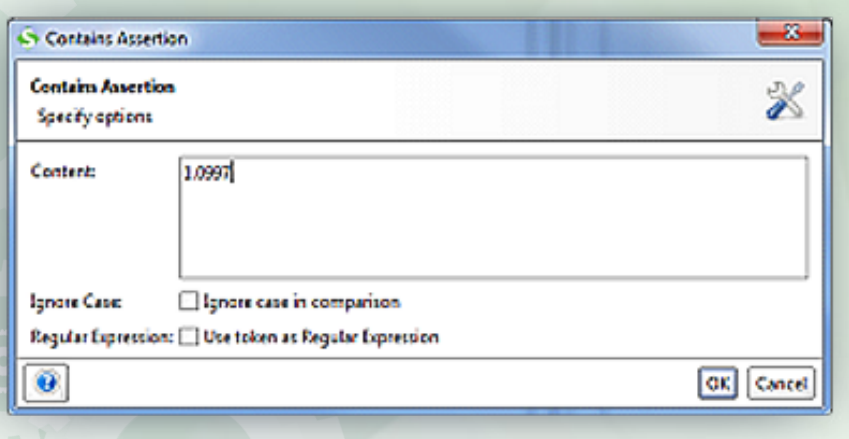


Click on the small +-sign at the top of the assertions editor.



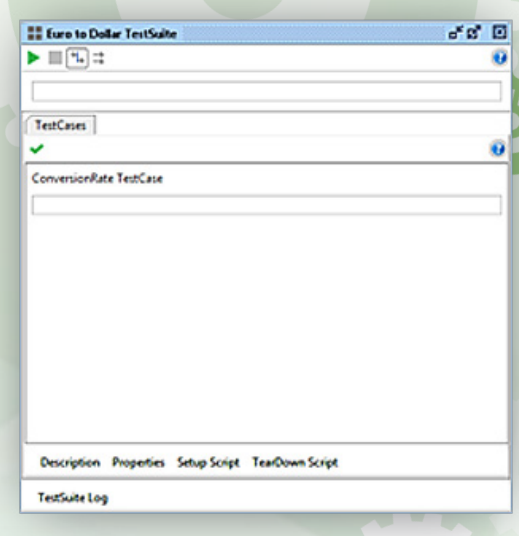
Select "Property Content assertions." The first one in the list is a Contains assertion. Let's use that one. Click on the "Contains" box and then click on "Add" to add it to the test case.

Add the current conversion rate, in this case 1.0997, in the next dialog box.



Click "OK" to add the assertion to this test.

We have one assertion to start with. We will probably want to add more assertions later, but let's start small and just have one. We will add more moving parts when this first case has been verified. Hardcoding the conversion rate between two currencies like this is brittle. We will have to do something about that later, also.



Run the test

The first thing we want to do is to run a single test case.

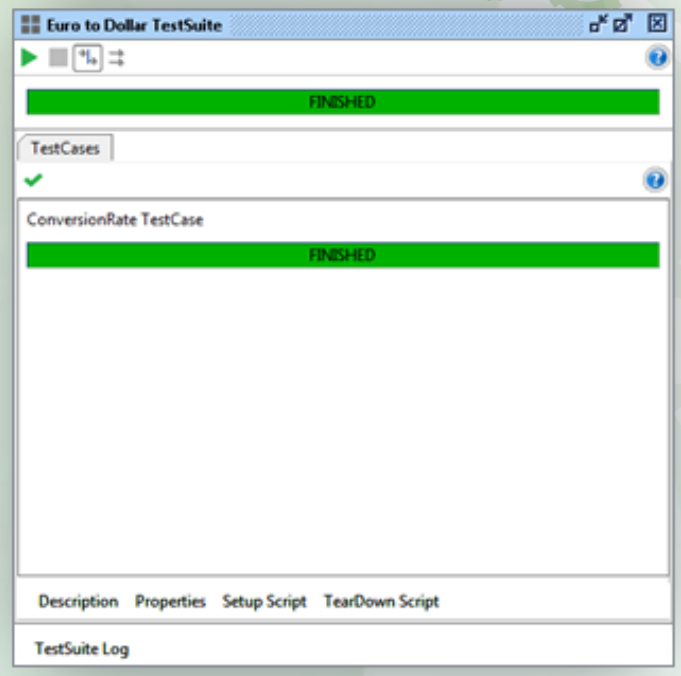
- Click on the small, green arrow at the top left corner of the request editor.
- This will send the request to the server. The assertion will be green if the value 1.0891 is available in the response.

Next step is to run the entire test suite. It doesn't contain more than a single test at the moment. The result should be exactly the same as run- ning a single test.

Run the test suite

Executing the entire test suite can be done like this:

- Double click on the test suite "Euro to Dollar TestSuite."



Click on the small, green arrow at the top of the test suite editor.

All test steps will be executed and they should be green. There is only one in this case, but one is enough to verify the execution.

Further exploration

Now that you have created a test suite, it's time to add more test steps or add more assertions to the step we just created. The basic infrastructure is in place. Now it's just more of the same.

Create tests

Creating tests is mostly about adding proper assertions to a test case. We have our test step that will verify the conversion rate between Euro and US Dollar. The verification we added earlier is weak. The conversion rate between two currencies is seldom the same for a long period of time.

Other verifications we can add are verifying if a response is received fast enough and thus answer the question if the service we are testing is following a given service level agreement, or SLA.

Let's add verifications to our test case and discover more about how to verify the response.

Verify a range

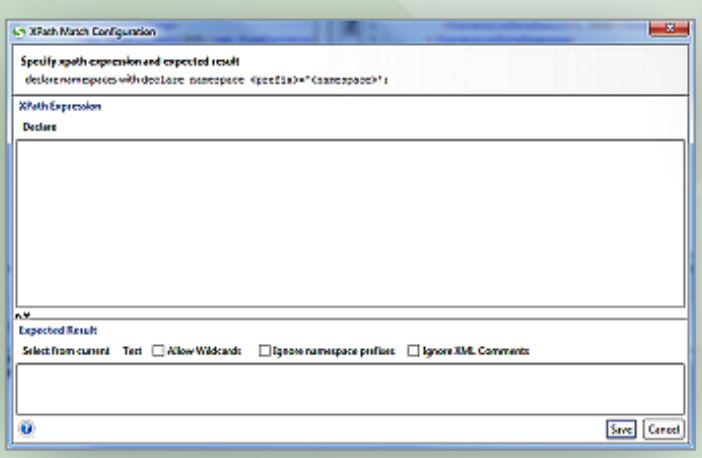
It's easy to verify that an answer contains a value. This may be great if you want to verify a value that is not expected to change. Some values change. We have to create a test that can handle a range instead of a fixed value.

Suppose that we can assume that the response will contain a value that is greater than something else. With this, we can verify a range.

Let's assume that the conversion rate between Euro and US Dollar is larger than 1 and lower than 2. How could that be expressed?

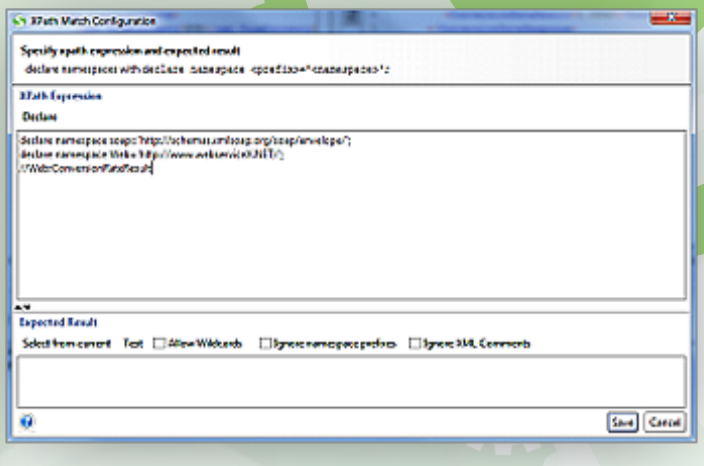
We need to start with locating the element that contains the value. When this is done, we need to compare the value with our assumption.

- Click on the “Assertions” tab at the bottom of the request editor.
- Click on the small +-sign at the top of the assertions editor.
- Select “Property Content.”

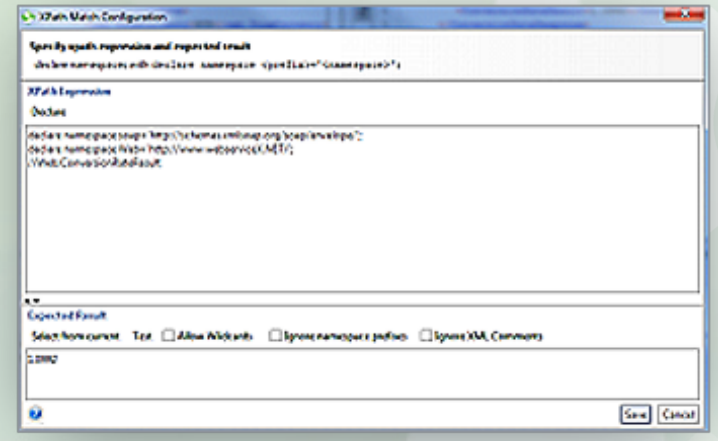


- Select “XPath match” and click “Add.”
- Click Declare icon in the XPath editor. SoapUI will declare two name spaces for you. They can be called anything. The two name spaces that were declared are called soap and ns1.
- Rename ns1 to something more descrip tive. I chose Web, as it’s the same name space that is used in the request. Web is perhaps not much more descriptive than ns1, but at least it is not an abbreviation for Name Space 1.
- The next step is to add an XPath[3] expres sion that will search for the element that contains the conversion rate.

```
//Web:ConversionRateResult
```



- Verify that the syntax is correct by pres- sing the “Select from current” button. You should get the value available in the res- ponse. I got 1.0992 when I tried.

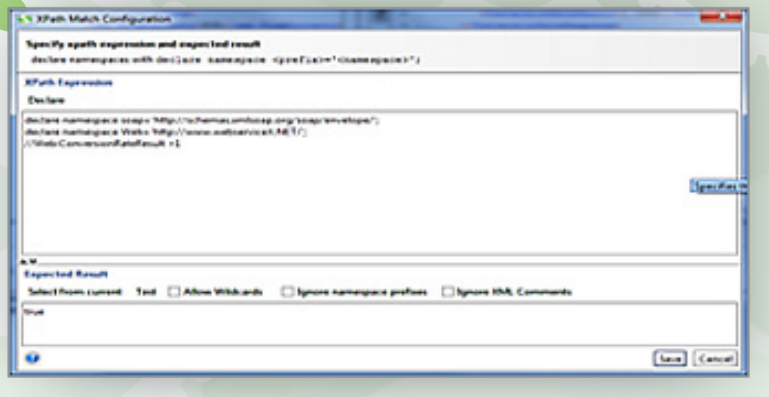


- When we are able to locate the element, let’s continue with verifying that it is lar- ger than one. Rewrite the XPath expression like this:

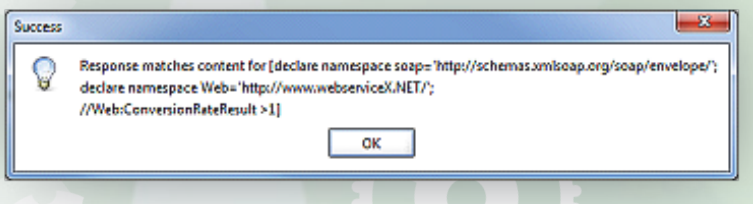
```
//Web:ConversionRateResult > 1
```

3 http://www.w3schools.com/xpath/

- Verify the result again by pressing “Select from current.” I got the result true.



- Press the Test button to execute the test. I got a dialog box that told me that the test was a success.



- Save the current verification by pressing “Save.”

To verify that the conversion rate is lower than 2, create a similar assertion and change the XPath to

```
//Web:ConversionRateResult < 2
```

An easy way to do this is to clone the XPath assertion. Right click on it, select “Clone” and make the appropriate changes.

Verify response time

Verifying the response time is often important. A slow API is a problem waiting to emerge.

Customers will probably start to complain when you have a lot of traffic and they don’t get their response quickly enough. Let’s prepare for this by requiring the service to respond within 1 second.

- Add a new assertion.
- Select “SLA” and “Response SLA.”
- Add it.
- Specify the desired response time. The default value is 200 as in 200 ms. We want to allow one second, so set the new value to 1000 ms and press “OK.”

Adding a minimum response time like this is easy. A more difficult problem may be to actually deliver such short response times.

But that is not our current focus; we are just verifying that the service is fast enough.

Verify schema compliance

APIs should follow their own schema definition. We used the schema definition when we created the request to the service. We need to add verification if we want to be sure that the response follows the schema it should be following.

- Add a new assertion.
- Select "Compliance, Status and Standards."
- Select "Schema Compliance."
- Add it.
- Define the schema, the response should follow. The suggested schema is the same schema as we used to create the service. This is the schema we want to use in this example.
- Click "OK."

All requests now have to follow its schema definition. If they don't, the test will fail. This is great if we want to catch a bug introduced to the service.

Many more tests can obviously be added, but the ones I have added here should get you started.

Go pro: Point and click testing

SoapUI NG Pro has been extended to support point-and-click tests and assertions. This is a real time saver and is something you will really appreciate when you're working under time constraints.

Creating an XPath assertion can be tricky when you need to get the correct data. It's very easy to do using SoapUI NG Pro, where you can click on the data element in the response and get a proper XPath for it.

SoapUI NG Pro also makes moving data from a response to the next request easy by assisting with the XPath expression you want to use to retrieve the correct data.

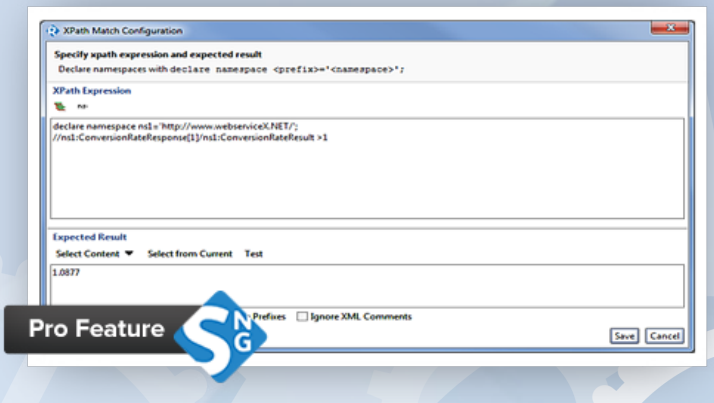
Verify a range

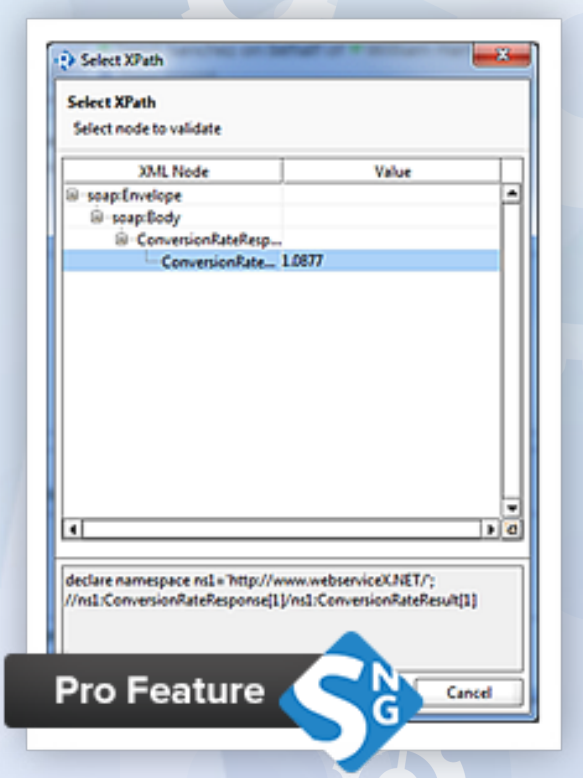
Let's revisit verifying that the response is within a certain range. We did that by writing an XPath validation manually. SoapUI NG Pro has implemented a wizard that will remove a lot of the details you have to remember and allow you to focus on the actual problem: verifying that the conversion rate between Euro and US Dollar is larger than one.

- Add a new assertion.
- Select "Property Content."
- Select "XPath Match."
- Add it.
- Click on the small node selector to the left of the "ns": button used in the previous step.

Select the node you are interested in verifying. An XPath expression will be created for you and will locate that specific element. This is a read only field.

- Click “OK.”
- Change the XPath created for you from.





- Click “Select from current” and notice that the “Expected Result” area gets set to true.
- Run the test by clicking the “Test” button. It should be a success.
- Save the test.

We just added the same verification, but we didn’t have to understand or know how to write the XPath to locate the value we wanted to verify.

Spending less time writing XPath expressions and more time focusing on what really needs to be tested is a good reason for going pro.

//ns1:ConversionRateResponse[1]/
ns1:ConversionRateResult[1]

to

//ns1:ConversionRateResponse[1]/
ns1:ConversionRateResult > 1

Data driven testing

Verifying that an API is functioning correctly using different test data is an important aspect of testing because many bugs are not elicited by a system using one single static "happy path" set of conditions. Likewise, the data that comes back from an API often needs to be checked for accuracy and consistency to the data that was used as input, like say the current balance of a specific account after completing a tran-saction, or that payment details of a specific account are not provided when logged in as another account.

In this example, a service should perhaps be able to translate between many different currencies. Verifying each combination by hand can very quickly be too much to cope with.

SoapUI doesn't have any built in support for reading test data from a source and using it as a parameter to a test. It is possible to implement using a Groovy script that reads from the data source and passes parameters to a test case. This is, unfortunately, out of scope for an introduction to SoapUI, so it will not be covered here.

Another solution is to use SoapUI NG Pro that has support for reading from a data source and passing the result to a test case out of the box.

Go pro: Data source out of the box

Setting up a test to use a data source is available out of the box for SoapUI NG Pro, part of the Ready! API platform. The setup is done in the following steps:

- Create a data source.
- Create a test that uses values from the data source.
- Add a loop to iterate over the values found in the data source.

Let's add a data source to our example.

Create a data source

The data source is created as a test step. We need some kind of data before we can make any use of it. SoapUI NG Pro supports these types of data sources:

- Data connection – connect to a data source and use SQL to extract your test data.
- Data generator – generate data directly from the test suite without having to create an external set of data.

- Directory – read a set of files from a direc tory and use their content as test data.
- Excel – read an Excel sheet and use its content as test data.
- File – read a separated file and use as test data.
- Grid – define a grid in SoapUI NG Pro that will hold all test data.
- Groovy – create a Groovy script that ge- nerates the test data.
- JDBC – connect a data source and use SQL to extract the test data. This one is very similar to the Data connection above.
- JSON – read test data from a previous test step using a JSONpath expression.
- XML – read test data from a previous test step using an XPath expressions.

I want to keep it simple, yet useful, so I will use a comma separated text file where each row will be test data for one test.

Create a text file with this content:

EUR, USD

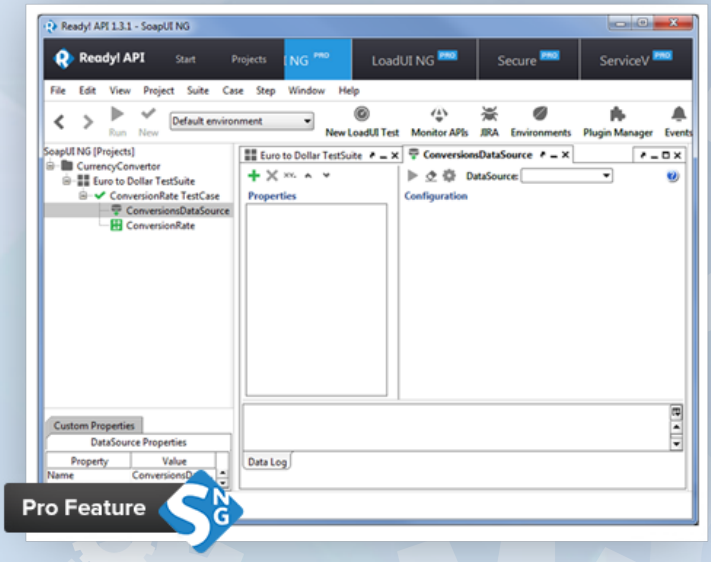
EUR, GBP

SEK, PLN

I saved it as test-data.txt. This will be our first data source. Our goal is to iterate through it and verify that we get conversion rates for the three different cases.

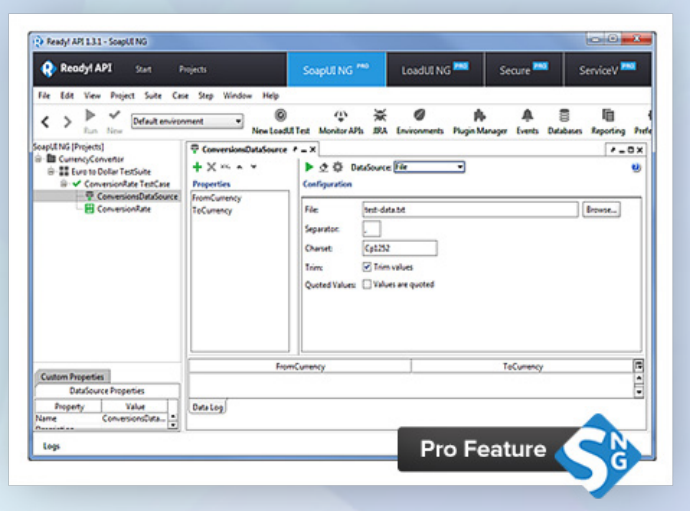
The next step is to connect to it.

- Add a new test and select the type "Data Source."
- Specify a name for it, I used Conversions DataSource.
- Define the type of Data source in the drop down Data Source. Select "File."



- Browse to the file you just created and select it.

- The default setting is that the file will be comma separated, leave that as it is.
- The default value for “Charset” is also ok.
- The checkbox Trim values is set and that is also something we want to keep. The values we read from the file should be stripped of whitespace.
- Our values are not quoted, so leave that checkbox unchecked.
- Name the two columns in the data source. Click on the +-sign in “Properties” editor to the left.
- Call the first property “FromCurrency” to indicate that the first column in our test data file will be the currency we want to get the conversion from.
- Call the next column “ToCurrency” to indi cate that the second column in the test data file is the currency we want to get the conversion to.



For now, this is all we want to set for our data source. Next step is to use it in a test case.

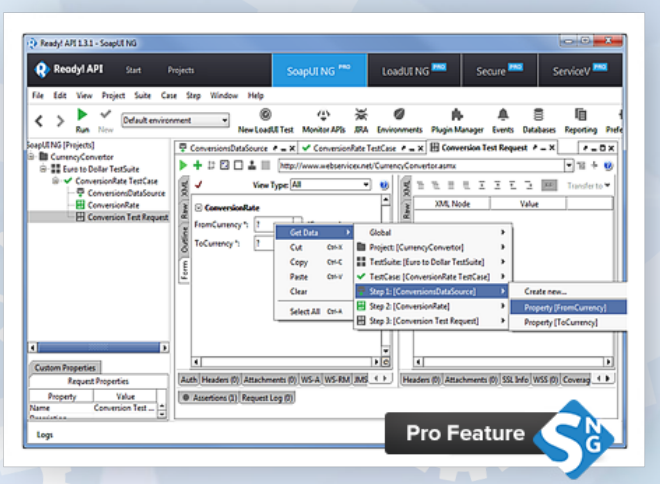
Create a test with parameters

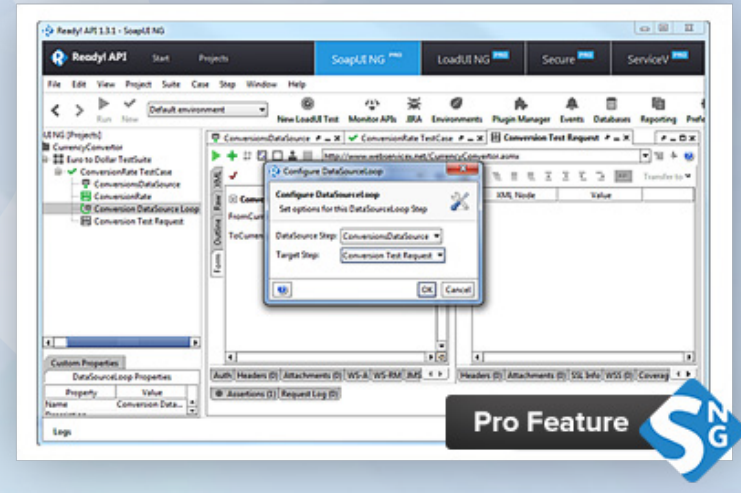
The test step is similar to the one we have created before. The only difference will be that instead of defining the currencies, we will use variables. These variables will be set with values from the data source and we will get many different tests. This is how to do it:

- Add a new test step to test steps, select the type “SOAP Request” and call it “Conversion Test Request.”
- Select the operation to invoke for this request. There is only one available.
- Define some settings in the next dialog box.
- “Add SOAP Response Assertion” is selected, leave it selected. It will verify that we actually receive a SOAP response.
- Leave the rest of the suggested assertions unchecked and Click “OK.”
- Connect the parameters by right clicking in the “FromCurrency” drop-down and selecting “Get Data”Select the step “ConversionDataSource.”
- Select “Property [FromCurrency].”
- Do the same for “ToCurrency.” Right click -> Get Data -> ConversionDataSource -> Property [ToCurrency].

That was all we had to do to connect the data source with the test case. Next step is to ite-rate, or loop if you want, over the data source, so we can execute the test case for each row that is found in the data source.

The data source and the test request have been connected. The next step is to execute a test run and verify that we send three different requests and get valid answers back.





Loop over the data source

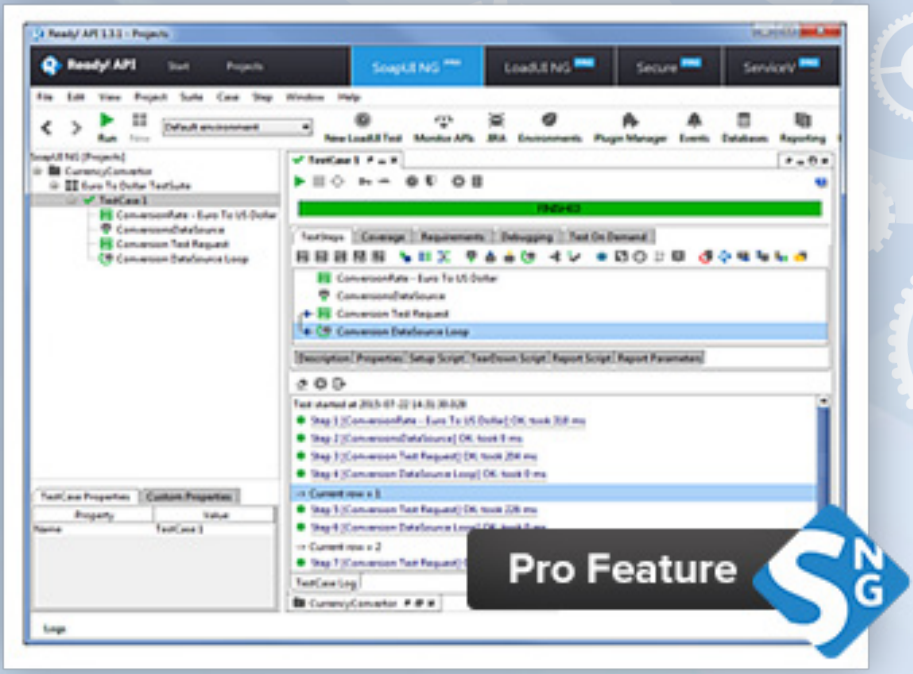
Iterating over the data source is done using a DataSource Loop.

- Add a new test step to test steps, select the type “Data Source Loop” and call it “Conversion DataSource Loop.”
- Double click on the new test step to bring up the editor where we can define which step to use as the data source for this loop, and define which step will be executed.
- Select “ConversionDataSource” as Datasource Step.
- Select “Conversion Test Request” as Target Step.
- Click “OK.”

Execute the test case

We have prepared a test case. Let’s execute it and see if we get something interesting.

- Double click on the “ConversionRate Test Case” to bring up the TestCase overview.
- The test steps should be listed for execution.
- Click on the small green arrow in the left, upper corner, to execute the steps.



Eight steps were executed. The first was our previous test step, the second read the data source, the next executed the test request, and the fourth read the next value from the data source. The last two steps where repeated until all rows in the data source were read. Try adding one more row in the file test-data.txt and execute the test case again. I added USD, EUR.

Adding one row resulted in one more test case that was executed. Verifying more combinations is just a matter of adding them to the data source file.

Further exploration

A simple file was used as a data source in this case. It is common to have other data sources. Many of them can be connected out of the box with SoapUI NG Pro.

Debugging

Writing a test perfectly the first time is difficult. Debugging and testing are common activities that you usually will spend some time doing until the test you are creating is perfect, or at least good enough.

Manual

Problem: You have a test suite with some steps, and you are not sure that the steps are correct.

Solution: Disable all steps, except the first step. Rerun the suite and verify the outcome. When the first step delivers the expected result, enable the next step. Adding one step at a time, in a controlled way, is a great way to build up confidence for the suite.

A general problem solving technique to debug anything is half-interval search[4]. You remove half of the steps and check to see if the problem you are trying to locate is present. If it isn't, then you know that your problem is in the other half. Dividing the steps into smaller and smaller parts will eventually, often quite soon, tell you exactly where the problem is.

Go pro: Test debugging

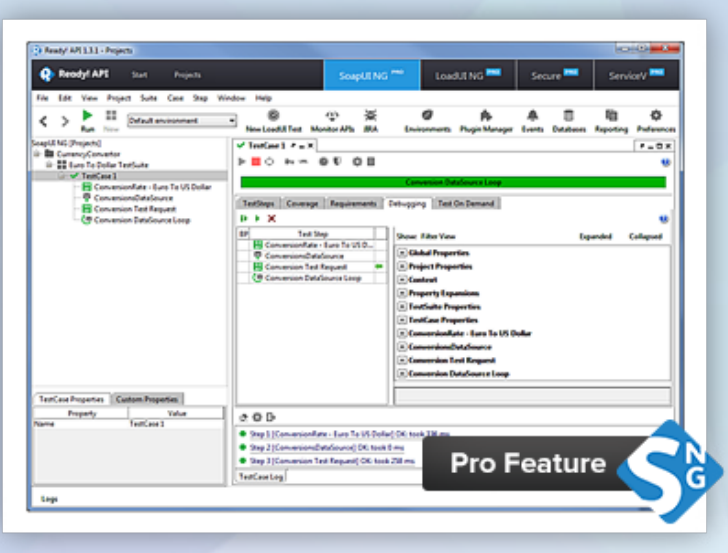
Easily follow your test flows step by step, and improve the quality of your tests with Test Debugging. There are times when you want to examine the behavior of your tests in detail – this is called debugging.

If you suspect errors in your tests or the services you're testing, Test Debugging will help you diagnose it. With the debugging support, you can execute your Test Steps one by one. Alternatively, you can add breakpoints and then run the test to those set breakpoints and view the current value of the SoapUI NG properties.

The Test Debugging Interface simplifies following Test Flow, Variables, Properties, Requests, Context, and much more, making test creation and improvement more streamlined.

Let's walk through a small debugging session to show how it can be done using SoapUI NG Pro.

- Double click on the "ConversionRate Test Case" in the Navigator.
- Select "TestCase Debugging" in the test case editor to the right.
- The steps are now listed and can be executed by clicking on the small, green arrow above the test step list. Doing so now will execute them in a sequence. You must add breakpoints to be able to step through them.
- Add breakpoints. Click just to the left, in the column named BP, of each test step. That will toggle a breakpoint on or off. Set breakpoints on all steps.
- Run the tests and notice that the execution stops at each breakpoint.



4 http://en.wikipedia.org/wiki/Binary_search_algorithm

- The current step is indicated with a small, green arrow, just to the right of the step name.
- Click on the solid green arrow to run one step at a time.

Debugging a simple case like this may not be very interesting, but it prepares you for more complicated cases.

In practice, debugging test cases saves valuable time during diagnostic analysis when either test data gets out of sync or the system being tested changes schema, a must-have for professional teams using SoapUI NG Pro.

Multiple environments

The test you have created should be possible to use for a system test environment as well as an acceptance test environment. The only thing you should change is the address, also known as the endpoint, to the server where the system is hosted.

Change the target environment

The environment you test against is defined in the service endpoints. Change it like this:

- Access the Projects tab.
- Double click on the Service definition, "CurrencyConverterSoap."
- Select the "Service Endpoints" tab.
- Click on the endpoint you want to change in the list and change the address.

http://www.webservicex.net/CurrencyConvertor.asmx to http://acc-www.webservicex.net/CurrencyConvertor.asmx if you want to test the same service on the host acc-www.

- Press enter when you are done with your change.
- Execute the test step. My execution failed, there is no host called acc-www.webservicex.net.

It is possible to use properties to define the endpoint, and this could be another way of setting the service endpoints in such way that they are quick to change. More information about this is available at SoapUI.org.[5]

5 http://www.SoapUI.org/Functional-Testing/structuring-and-running-tests.html

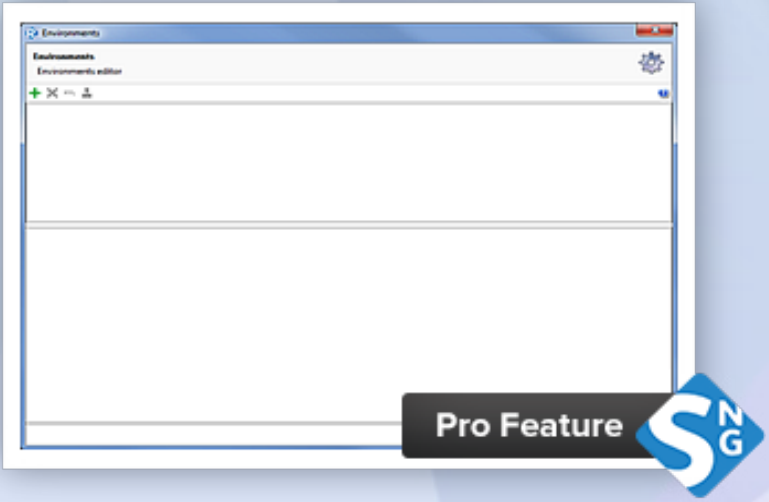
Go Pro: Multi-Environment Support

If you're testing different environments, such as Staging and Production, this feature will be a major time-saver. Manually changing all the different connections and end-points of your tests when you need to switch environment is cumbersome. With SoapUI NG Pro you can choose which environment you want to apply through a simple drop-down menu.

SoapUI NG Pro then automatically changes the necessary parts of your tests to enable it to run in the new environment. All that's needed is a one-time setup of your different environments, and you are good to go.

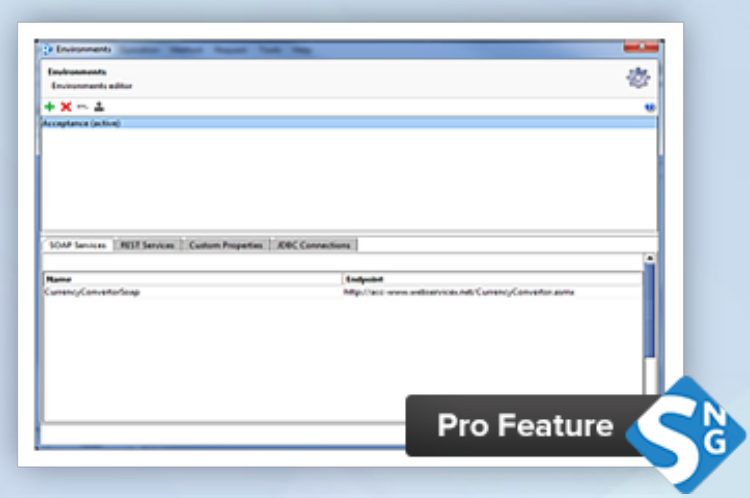
Let's walk through an example on how you can set up different environments so you can switch between them quickly.

- Select the project.
- Click the "Environments" icon below the SoapUI NG Pro tab.



- Click on the small +-sign to add a new en vironment.
- Name it "Acceptance."
- Keep the default values and do not copy from the test case. We will add a good end point manual in a short while.
- Click "OK."
- All requests and their endpoints are listed in the dialog box below the Environ ment name.
- Double click on "CurrencyConverterSoap."
- Add a good endpoint for it. I use:

http://www.webservicex.net/CurrencyConvertor.asmx



- Test it by executing the test case and se lecting "Acceptance" as environment.
- It failed, there is no host called http://acc-www.webservicex.net.
- Create another environment by following these steps again, call it "Production" and use the endpoint.

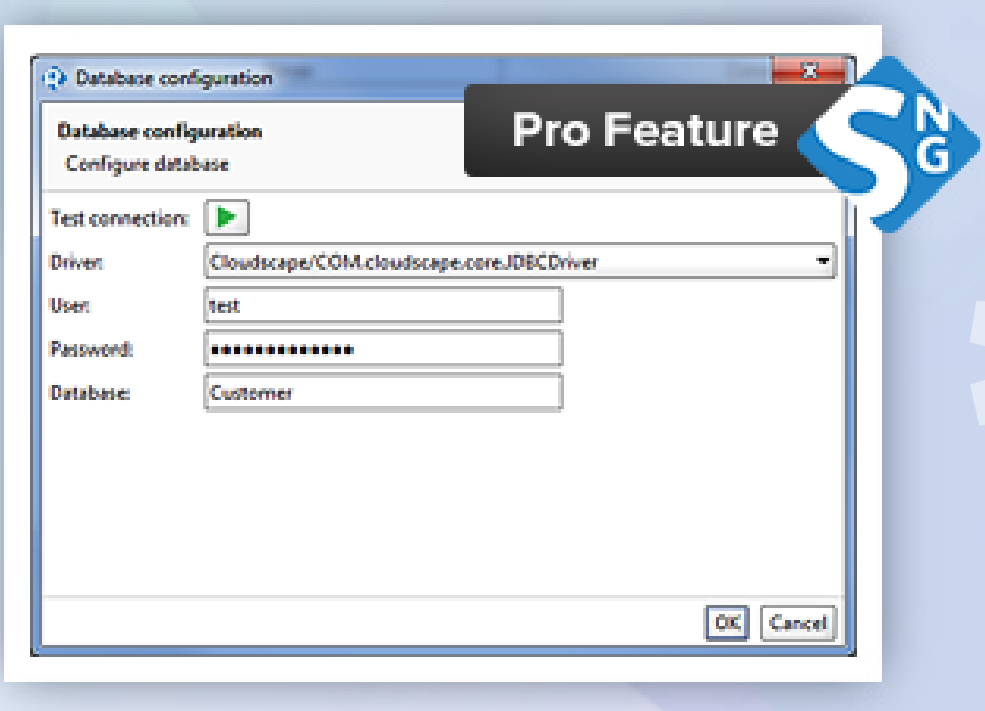
http://www.webservicex.net/CurrencyConvertor.asmx

If all was done correctly, things should work when you execute the test and use the environment "Production".

When you change the environment you test, a database hol-ding the current data set for the environment will often have to be changed. This can be done at the same time. Let's extend the example a little bit by adding support for different databases at the same time as you change environment.

We don't have a database connection in our example, so let's add one.

- Select the project.
- Click on the "Databases" icon below the Secure tab.
- Add a new data connection by clicking on the small +-sign in the left corner.
- Specify a name for the data connection. I use "Customer" to indicate that this is our customer data connection.



- Click "OK."
- Select the correct database and connection details. I set up a MySQL connection.

We just created a data connection. Now let's make sure that we connect to different databases when we change environment.

Open up the "Environments" section again.

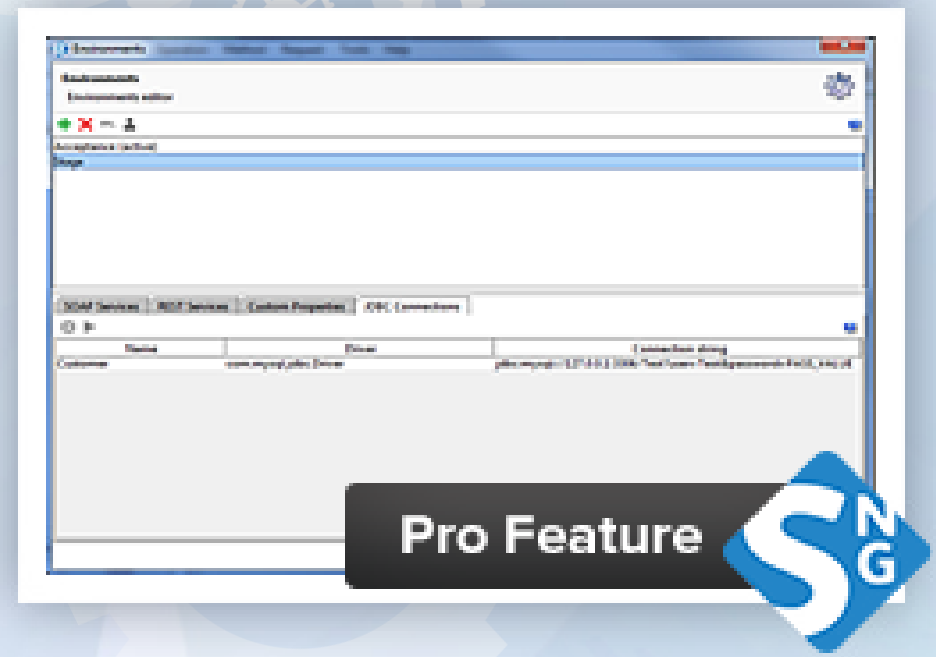
We have an environment called "Acceptance", let's add another one called "Stage." Click on the small +-sign above the environment list and call it "Stage."

Keep the setup defaults empty and click "OK."

Click on "Acceptance" and notice that we have an endpoint set.

Click on "Stage" and notice that we haven't an endpoint set. Set it to something appropriate.

Click on the "JDBC Connections" tab and check the value set for "Ac-acceptance" and then the value set for "Stage."



Acceptance has the same connection details as we specified earlier. Stage doesn't have any Connection string defined yet.

Double click on the "Customer JDBC" connection for Stage and set up the connection string. The result will be that when we test against Acceptance, we will use the endpoint defined for SOAP Services and the data connection defined in JDBC Connections for Acceptance. When we change environment to Stage, both of these settings will be changed. This allows us to change between different environments very quickly. Using different environments could be a good reason to go pro.

Reporting

Reporting the status of your test runs is important. A test report is often required before you are allowed to deploy an API into production.

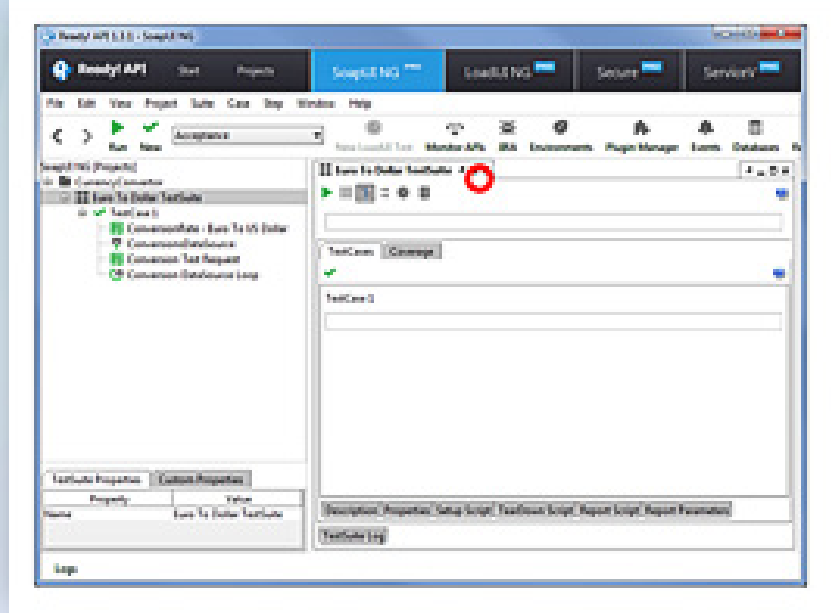
There is no built-in support in SoapUI to generate reports from a test execution. This is only available in SoapUI NG Pro and part of the Ready! API platform.

Create a test report

A test report is created based on a test suite execution. This means that we must start by running a test suite before a report can be generated.

- Execute the test suite Euro To Dollar Test Suite.

Click on the "report" button in the toolbar to generate a report. It's the last icon below the test suite name.

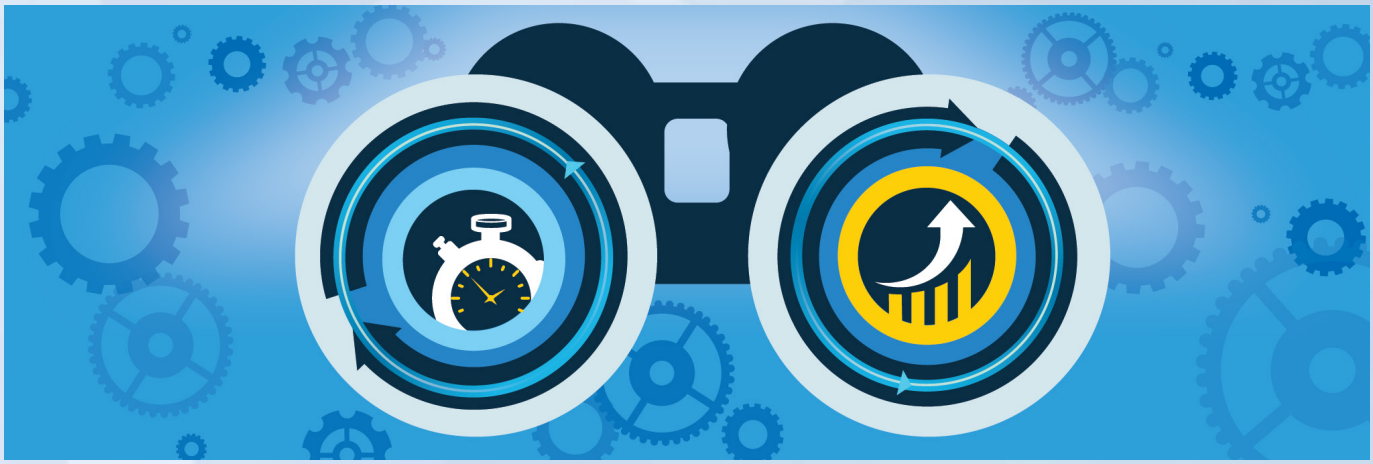


- Select a format TestSuite Report is the format you'd use if you want to visually inspect the data. The other types are intended for computers.
- Click "OK."
- A preview of the report shows up.
- Save it by clicking on the floppy icon to the far left.
- A "save" dialog box shows up where you can define the name for the report as well as the format. I chose RTF, Rich Text Format, because I want to be able to edit it later, using Microsoft Word.
- PDF – Portable Document Format.
- RTF – Rich Text Format.
- ODT – Open Office.
- HTML.

SoapUI NG Pro has an extensive set of prepared reports that supports these formats:

- Single sheet XLS – Excel.
- Multiple sheets XLS – Excel.
- CSV – Comma-separated file for import to Excel.
- XML.

You can change the content of the reports, but that is out of scope for this SoapUI NG introduction.



Organize your work

How you organize your work is obviously not something that should be dictated by a tool. At the same time, the tool you are using may support working in different ways. SoapUI NG Pro will allow you to organize your work using workspaces. It will give you a choice on how to save your project so it can be shared with other testers. Let's start with workspaces and then discuss version control and sharing.

Workspaces

SoapUI NG Pro uses workspaces[6] where you do all your work. A workspace contains all your current projects. This means that you can work on more than one project in the same SoapUI NG Pro session. When you want to change to another project either add it to the current workspace or switch to another workspace.

Workspaces can be shared by a team. Sharing workspaces will force everyone to work with the same set of projects all the time. This may not suit how you work.

Version control

Storing your projects in a version control system is considered a best practice. It will enable you to go back to a known version and make it easier to share files with other testers.

Why?

A shared resource on a file system is a fragile way to share your work. People can overwrite files that another person is working on and changes can easily get lost. File history is hard to keep track of. Tracking who made which changes to the project is not possible. Therefore, sharing files through a version control system is often used in development projects.

The tests you have created with SoapUI NG Pro have a significant value and should be protected as much as possible. Storing them in a version control system is therefore something you'll want to do.

What?

What should be stored then? Your workspace may or may not be relevant to other testers. The workspace reflects how you personally group different projects. Sharing them with the other testers is something you need to discuss and then decide upon. You should store your project files. They are the most important outcome from your testing work. They should be used and maintained for all testers in your team.

SoapUI NG Pro saves the project as a composite. This means that each part of the project will be saved in separate files instead of saving the entire project in the same file. This is a useful feature to help team members cooperate while testing by allowing more than one person to work on the same project. Cooperating and sharing are the keys to scalability. One person can only do so much; two people can do a lot more.

Combining composite projects and storing them in a version control system allows the testing effort to scale and more tests can be created and maintained, compared to not using composite projects.

How?

There are many different version control systems. Some of them are large enterprise solutions, others are open source and freely available. They all solve the same business problem by providing a single repository for your files and versioning the changes.

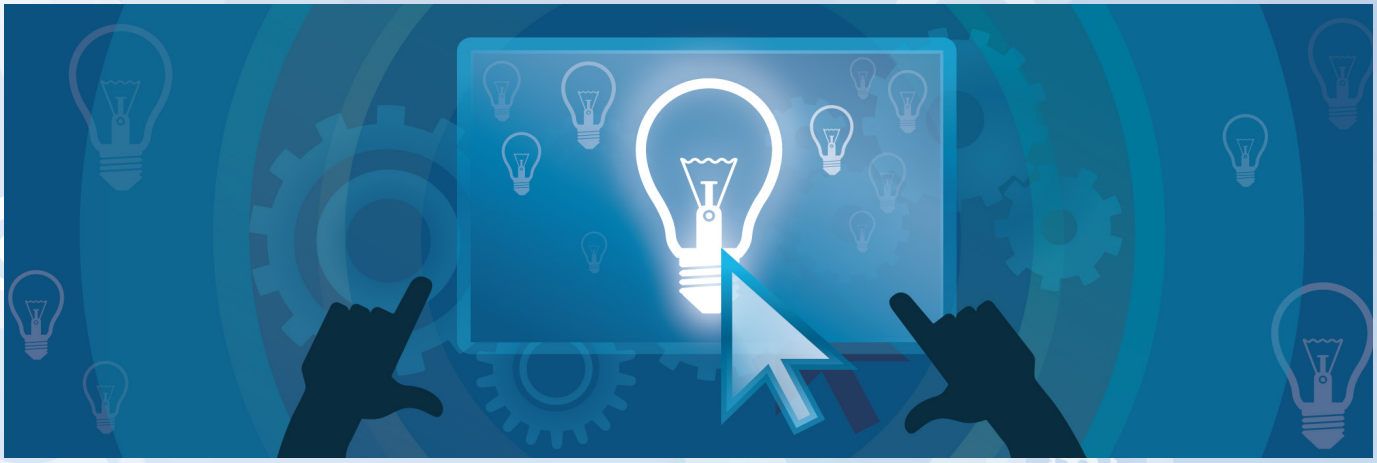
Open source or an enterprise system?

This is a question that depends a lot on the organization you work with.

Some organizations want to buy tools that are officially supported; others rely on open source tools and just buy support they need

6 http://www.SoapUI NG.org/Working-with-SoapUI NG/mana-ging-workspaces.html

when they need it. Open source solutions are used to store code and data related to some very high profile systems. The Linux kernel, for example, is stored using an open source version control system called Git. SoapUI open source and SoapUI NG Pro is both also version-controlled using Git.



Automation

Automated testing is an area where many companies put in a lot of effort. They want to eliminate the mundane, expensive and manual testing that used to be the only way you could verify a system. They also want the developers to get fast feedback as soon as they have made a change to a system.

How can SoapUI NG Pro be used for test automation? The trick is to execute test suites developed using either SoapUI or SoapUI NG Pro, without using a user interface, and instead trigger it from some kind of automation build tool. There are many different tools that could trigger the execution. Most of them have a way to trigger things from a command line prompt. This could be a batch script in Windows, a shell script in Unix, or it could be Maven pro-ject in a Java build environment.

Run SoapUI NG Pro test suites in Continuous Integration (CI) Systems

The first example will show how SoapUI NG Pro can be exe-cuted from a command line. The second example will show how Maven can be used to run SoapUI NG Pro.

Command line

Command line tools are good if you want to be able to exe-cute something from a script that you have created yourself. SoapUI NG Pro has support for executing from a command line.

Execute it like this:

testrunner project_file

where project_file is a SoapUI NG Pro project file that could look like this c:\projects\my-SoapUI-project.xml if you are on Windows.

The command line tool takes a lot of parameters that will allow you to control the execution. They are described in the online documentation.

7 http://www.SoapUI NG.org/Test-Automation/functional-tests.html

It turns out that it is actually the command line tool that is used when a test is executed from the TestRunner. If you want help with running the exact same execution from a command line that you executed from the TestRunner, then try this:

- Execute the test you want from the test runner.
- Scroll back in the execution log until you find the launch of the test runner in the beginning of the log.
- Copy the command and use it in your own script or from a command line.

Maven

Maven is a popular tool for building Java. Running a SoapUI NG Pro test from Maven is similar to running it from a command line. The difference is that you need to set up a Maven project and then use Maven. This is a good alternative if Maven is already building the rest of the project. Many Java projects are using Maven. Let's set up a Maven project that will run SoapUI NG Pro.

Maven is very good at handling dependencies. There are repositories on the Internet where dependencies that could be used in a Maven project are stored. When Maven realizes that it doesn't have a local copy, it will download the resources it misses. This is a feature we can use when retrieving the SoapUI NG Pro plugin.

Maven uses a file called pom.xml for all settings. We need to extend an existing pom file for your project or create a new one. Going through all the details for a Maven project is out of scope here. I will just focus on the parts that are SoapUI NG Pro specific and leave the rest of the details to be explained by other resources [8] [9] on the Internet.

8 http://maven.apache.org/

9 https://thomassundberg.wordpress.com/2013/02/24/maven-the-simplest-possible-introduction/

SoapUI NG Pro is a tool in the Ready! API platform, which includes other tools for load and security testing as well as lightweight API service virtualization, and also includes tool-agnostic features like reporting, events, and plugins. Plugins are written at the platform level to perform various tasks or act as integrations, sometimes specific to a single tool and other times applying to other tools as well.

As such, to run a SoapUI NG Pro script in Maven, you need the Ready! API Maven plugin (available online). Additionally, there are Ready! API specific things that must be added to your Maven pom are.

- Specify a repository where the Ready! API plugin can be found.
- Add and configure a plugin that will execute the test.

Add a repository for Ready! API

The Ready! API plugin for Maven must be found and downloaded. It lives in a repository that must be specified specifically. Add this to your pom:

```
<pluginRepositories>
    <pluginRepository>
        <id>eviwarePluginRepository</id>
        <url>http://smartbearsoftware.com/
        repository/maven2/com/smartbear/
        ready-api-maven-plugin/</url>
    </pluginRepository>
</pluginRepositories>
```

The value of the id tag could be set to almost anything, as long as it is unique. A descriptive name is, as always, preferable.

Now that we have located the Ready! API plugin, let's continue with adding and configuring it.

Add and configure the SoapUI NG Pro plugin

Add the following snippet to the build section of your pom:

```
<plugins>
    <plugin>
        <groupId>eviware</groupId>
        <artifactId> ready-api-maven-plu
        gin</artifactId>
        <version>1.3.1</version>
        <configuration>
            <projectFile>simple-test-soapui-
            ng-project.xml</projectFile>
        </configuration>
    </plugin>
</plugins>
```

This will specify that we want to use a plugin called 'ready-api-maven-plugin' and that we want to use version 1.3.1 of the plugin.

The project we want to execute is specified in the configuration section and it is called 'simple-test-soapui-ng-project.xml'.

You can specify more things, but this is a small start with just a few moving parts.

The next step would be to execute a test.

Execute a test from Maven

Maven is a command line tool that is executed with the command

mvn

You want to specify at least one goal that Maven should execute. The simplest possible could be

mvn validate

It will validate that the format of your pom is correct. To actually execute the test we specified above, run this instead:

mvn eviware:ready-api-maven-plugin:test

We specify that we want to execute the Ready! API plugin and the goal test. The plugin has other goals that you could use. Executing load test is one of them.

These were just a few snippets. Let's put them in context in a complete example.

Properties

You can define properties in your test. You need to add a section called 'projectProperties' in the configuration section. Then add the values you want as name value pairs.

```
<configuration>
    <projectFile>simple-test-soapui-ng-
    project.xml</project-File>
    <projectProperties>
        <value>message=Hello World!</
        value>
    </projectProperties>
</configuration>
```

There should not be any space after the equals sign. It would place a space in front of your value. That could be a subtle bug that you would like to avoid.

A complete example

A small example can look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.smartbear</groupId>
    <artifactId>soapui-ng-example</artifac
    tId>
    <version>1.0-SNAPSHOT</version>
    <pluginRepositories>
        <pluginRepository>
            <id>eviwarePluginRepository</id>
                <url>
                http://smartbearsoftware.com/
                repository/maven2/com/smart
                bear/ready-api-maven-plugin/
                1.3.1/</url>
        </pluginRepository>
    </pluginRepositories>
    <build>
```

```
            <plugins>
                <plugin>
                <groupId>eviware</groupId>
                    <artifactId>ready-api-
                    maven-plugin</
                    artifactId>
                    <version>4.5.1</version>
                    <configuration>
                        <projectFile>simple-
                        test-soapui-ng-pro-
                        ject.xml</projectFile>
                    </configuration>
                </plugin>
            </plugins>
        </build>
</project>
```

I have defined the necessary plugin repository and configured the Ready! API plugin. I used the pro version here. If you would like to do the same thing for SoapUI open source, then use the plugin called maven-soapui-plugin. To execute this example, run Maven like this:

mvn eviware:maven-soapui-plugin:test

from a command line prompt.

You can define properties that will be set from the pom and you can run the tests as an integrated part of the Maven build. That is more of a Maven discussion than a SoapUI discussion, so I will leave that out to minimize the number of moving parts.

More details

The Maven plugin can be used to do more than what I described here. The details can be found in the Ready! API documentation online. [10]

Automate the execution

After creating your SoapUI NG functional tests with test suites and connecting SoapUI NG tests to your build tool, it's time to execute them, as soon as a change has occurred.

There are many different build tools available. One popular tool is Jenkins. Jenkins will be able to listen for a change on a URL and kick off a test job. This would make it possible for you to start testing as soon as there is something new to test.

All build servers handle Maven projects out of the box, so setting up a build that executes SoapUI NG tests is just a matter of setting up a Maven build. Describing that is, however, out of scope here.

10 http://readyapi.smartbear.com/readyapi/integration/maven/start 11http://jenkins-ci.org/

Conclusion

After walking through these example, you are now able to verify and test APIs using SoapUI.

In contrast to using a homegrown or purely script-based testing approach, your chances of securing the delivery of high quality APIs will be drastically improved.

You will save a lot of time and be able to focus on the real problem, testing, instead of the problem of creating your test tool.

SoapUI NG Pro gives you critical testing capabilities beyond the basics in SoapUI and as part of the Ready! API platform, opens your options up to perform comprehensive API load (LoadUI NG) and security testing (Secure) as well as API vir-tualization (ServiceV).

You will be able to save even more time, money, and heartburn by using the professional features of these tools in the Ready! API platform to speed up common tasks.

5 Reasons to Go Pro

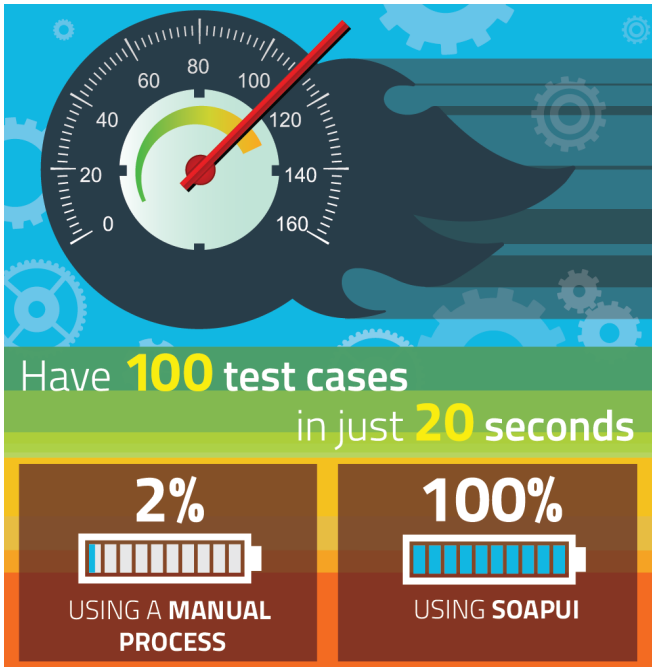
1. Get home faster, but still ship even better APIs

"Using a manual process to run 100 test cases previously required close to six hours of resource time. With SoapUI, testers can conduct the same number of tests in just 20 seconds. "

— **IMAD ALASSI**, Sr. Test Automation Developer

- Data-driven input and validation.
- Test and assertion coverage.
- Multi-environment support.

SoapUI NG Pro is the easiest way to simplify and speed up the process of shipping better APIs.

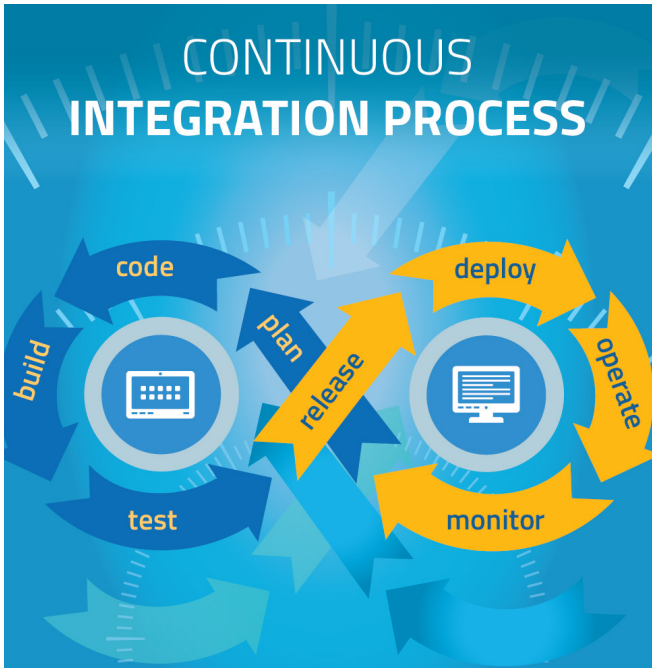


2. Works with your Continuous Integration process

Your CI process is critical to delivering changes to your APIs on time. Getting each component in the chain working seamlessly is important, which is why SoapUI NG Pro is built to easily fit in to the process.

- Connect Jenkins, TeamCity, Bamboo.
- Logging & reporting via command-line.
- Git storage / JIRA issue tracking.

Your CI pipeline is critical to delivering changes to your APIs on time. SoapUI NG Pro is built to easily fit in to your process.



3. Less time training, more time doing

"We experienced a first-class customer engagement with SmartBear. From our account rep to the project manager and the top-notch engineering team, we always received prompt and professional service."

— **FATIH COMLEKOGLU**, Chief Software Architect

- Visualize your REST & SOAP calls
- Test debugging and rapid diagnosis
- Javascript & Groovy scripting

Shipping software on time gets a whole lot easier when your team shares testing skills and use the best tools for the job.

4. Deliver confidence all along the way

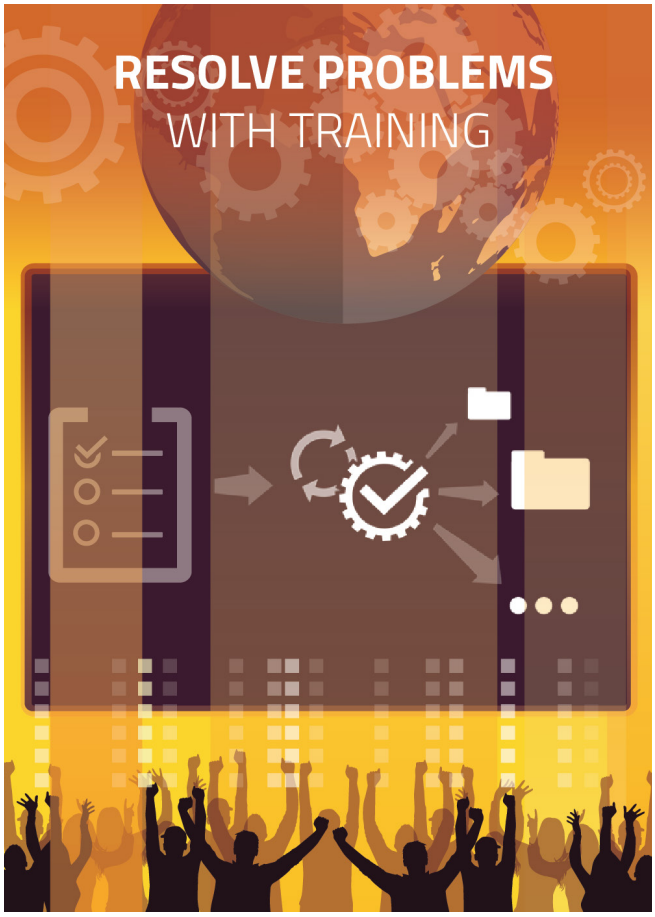
"All the stats are in one place as we manage application tests in real time. We can now more easily track improvements and find new bottlenecks."

— **ERAN KUX**, Performance Testing Engineer

- Performance and service virtualization
- Security considerations for REST & SOAP
- Extensible

SoapUI NG Pro is an API functional testing tool in the Ready! API platform, which also includes load testing, security testing, and service virtualization.

5. Resolve problems and rapidly on-board with training



If your team runs in to issues testing your APIs, our SoapUI NG Pro support team is ready to help get you back on track. SmartBear also partners with professional service providers to deliver training and certification in SoapUI NG Pro.

- Fast turnaround time on issues
- Comprehensive, professional training
- Weekly Q&A webinars

Got a technical question? Get it answered in a 14-day trial of SoapUI NG Pro by one of our professional technical staff.


SoapUI