

Raj Sewa Dwaar.

Customer API Testing Document

TABLE OF CONTENTS

Document Change Control	2
Table of Contents	3
1. Purpose.....	4
2. Prerequisite	4
3. Steps to Test Rest Service	4
3.1 Working with REST Requests	8
3.1.1 Request Editor Views	9
3.1.2 Request Message Tabs.....	11
3.2 Response Message Views.....	16
4. Testing Soap Service	21
4.1 Operations	23
4.2 Custom HTTP Headers	24

1 . PURPOSE

Here is the walkthrough of the usage of Soap UI Client tool which can be used for the testing for the services published on RSD. Priority of this document is to observe the header value passed.

Eg: X-IBM-Client-Id : 044b3c5b-4d42-4cb5-b757-ec509d18c483

2 . PREREQUISITE

SOAP UI to be installed in testing machine. You can download from below link.

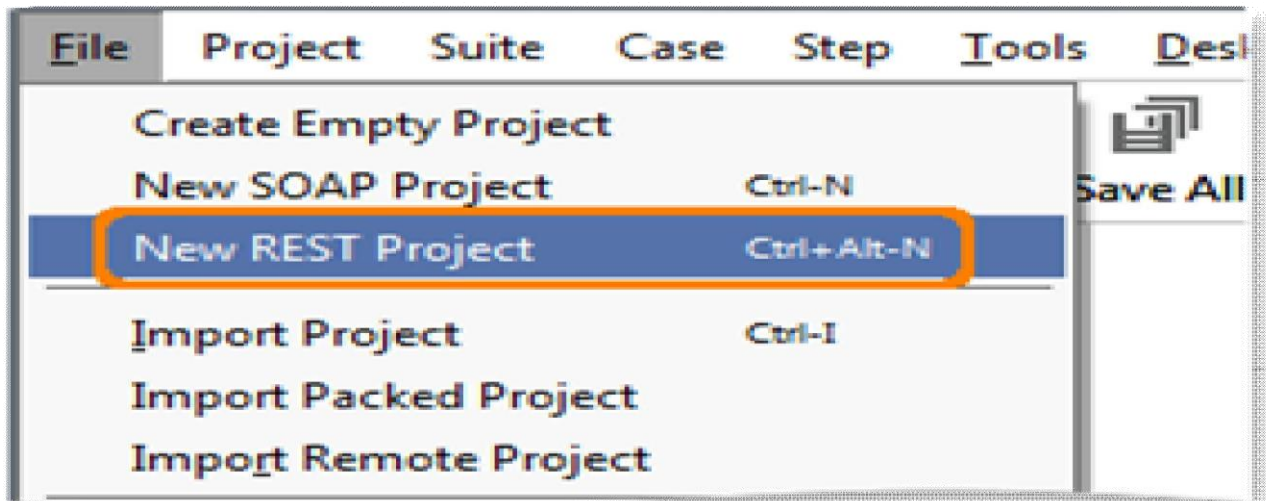
<http://www.soapui.org/downloads/soapui-open-source.html>

3 . STEPS TO TEST REST SERVICE

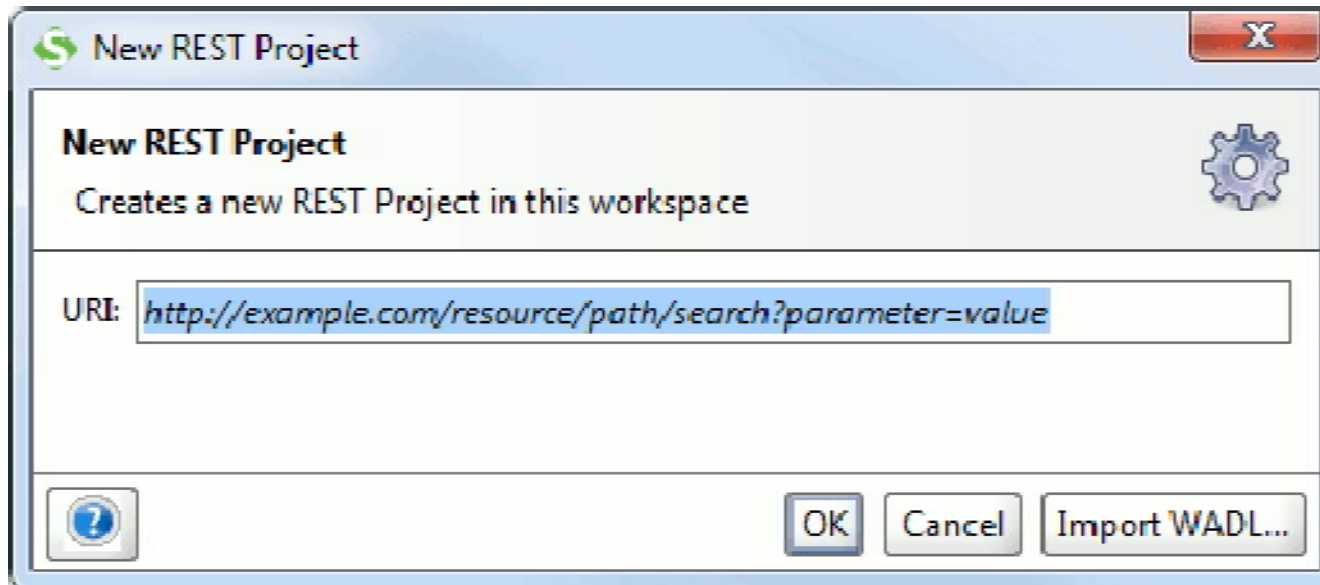
Let's have a detailed look at how SoapUI models REST Services.

Start by creating a new REST project:

Select **File | New REST Project** from the main menu:

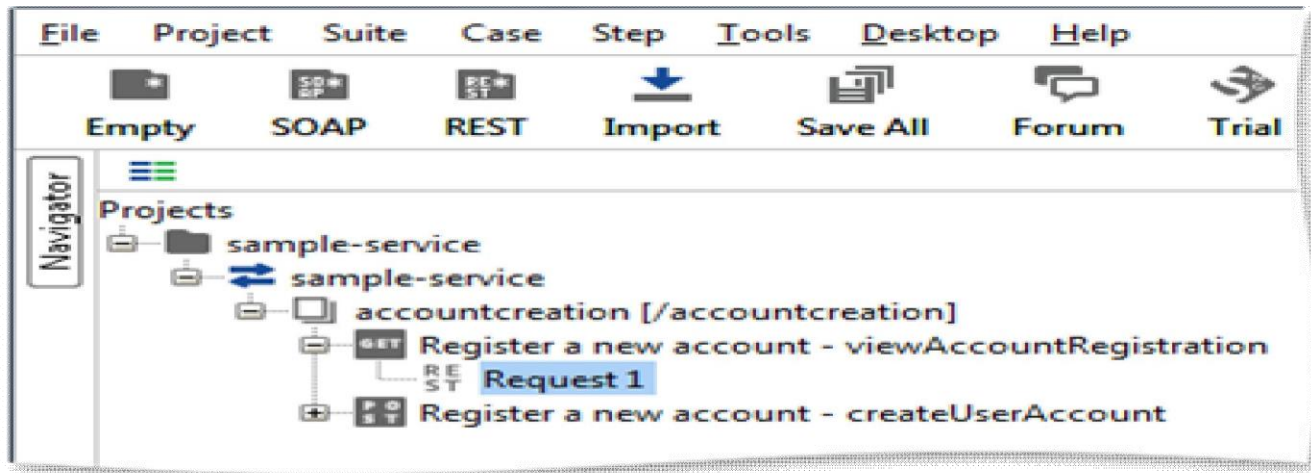


Just provide an URI:



Click **OK**.

SoapUI creates the specified project, resulting in the following object hierarchy in the navigator:

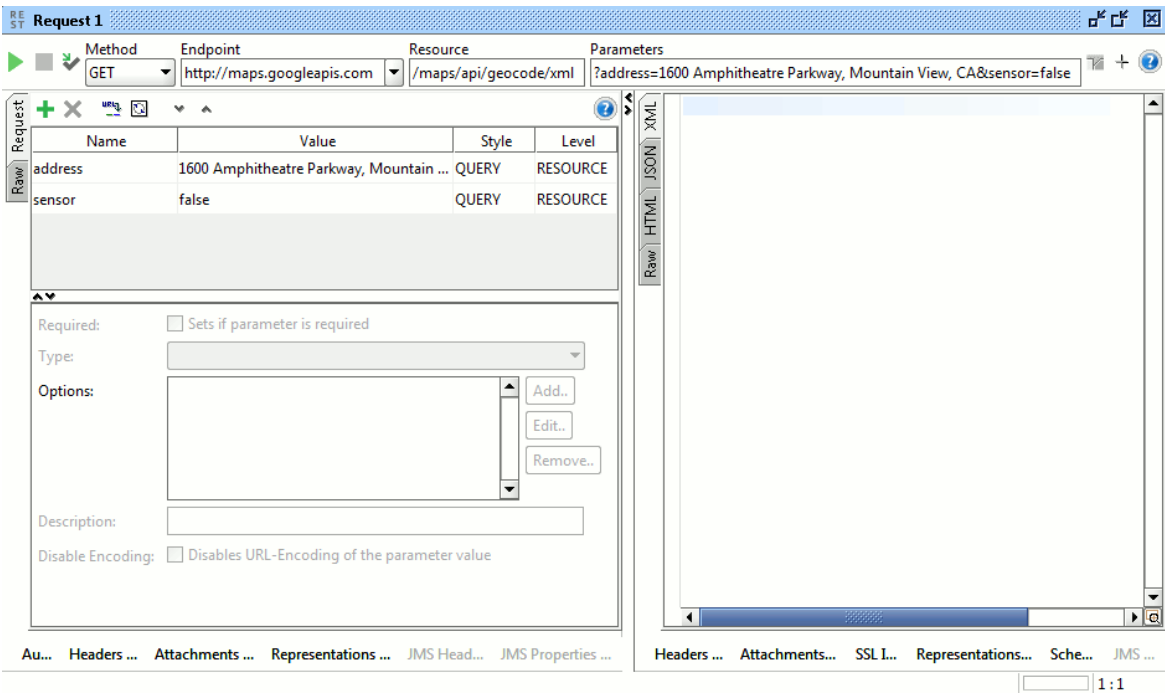


Here you can see the core items that make up a REST Service project:

- The project (sample-service)
- The REST Service (sample-service)
- A resource (accountcreation)
- A GET method for accessing the resource (Register a new account)
- A default request generated by soapUI for this method ("Request 1")
- ... and soon...

31 1 WORKING WITH REST REQUESTS

Double-clicking a REST Request in the navigator opens the REST Request editor window:



-
- A toolbar at the top with standard actions and the endpoint dropdown for easily changing the service endpoint
 - A Request Editor to the left with corresponding editor views along the left border and editor tabs at the bottom
 - A Response Editor to the right with corresponding editor views along the left border and editor tabs at the bottom

Let's have a look at the request and response editor views and tabs.

3.1.1 REQUEST EDITOR VIEWS

The Request editor has the following editor views available along the left border:

Request (shown above): shows a tabular view of all the parameters defined for the Request, these are the aggregate of the containing Method and its Resource and any parent Resources available. Table also contains the parameter style and level (RESOURCE or METHOD) at which parameter exists. If a parameter is added at RESOURCE level then it is used by all requests under that resource but if the parameter is at METHOD level then it will be used only by the requests under that method.

You can Add/edit/remove the parameters from the request editor. A new parameter is added at RESOURCE level by default but level can be changed by selecting other value in the drop down and it will propagate to all the requests in navigation tree of the level. A good thing to keep in mind is that this will affect other requests also in the navigation tree since parameter level is either RESOURCE or METHOD. The parameter value entered/edited in the request editor is local to request only and hence is not propagated to other requests in the navigation tree. Add your desired parameter and corresponding values in this table (property-expansion is supported also)

If the Method uses an HTTP Verb that sends a request body (POST or PUT), a corresponding editor for the message content is made available under the table of parameters:

+

×

url

?

Request

Raw

Name	Value	Style	Level
address	1600 Amphitheatre Parkway, Mountain View...	QUERY	RESOURCE
sensor	false	QUERY	RESOURCE

^

v

Required:

☐ Sets if parameter is required

Type:

Options:

▲

▼

Add..

Edit..

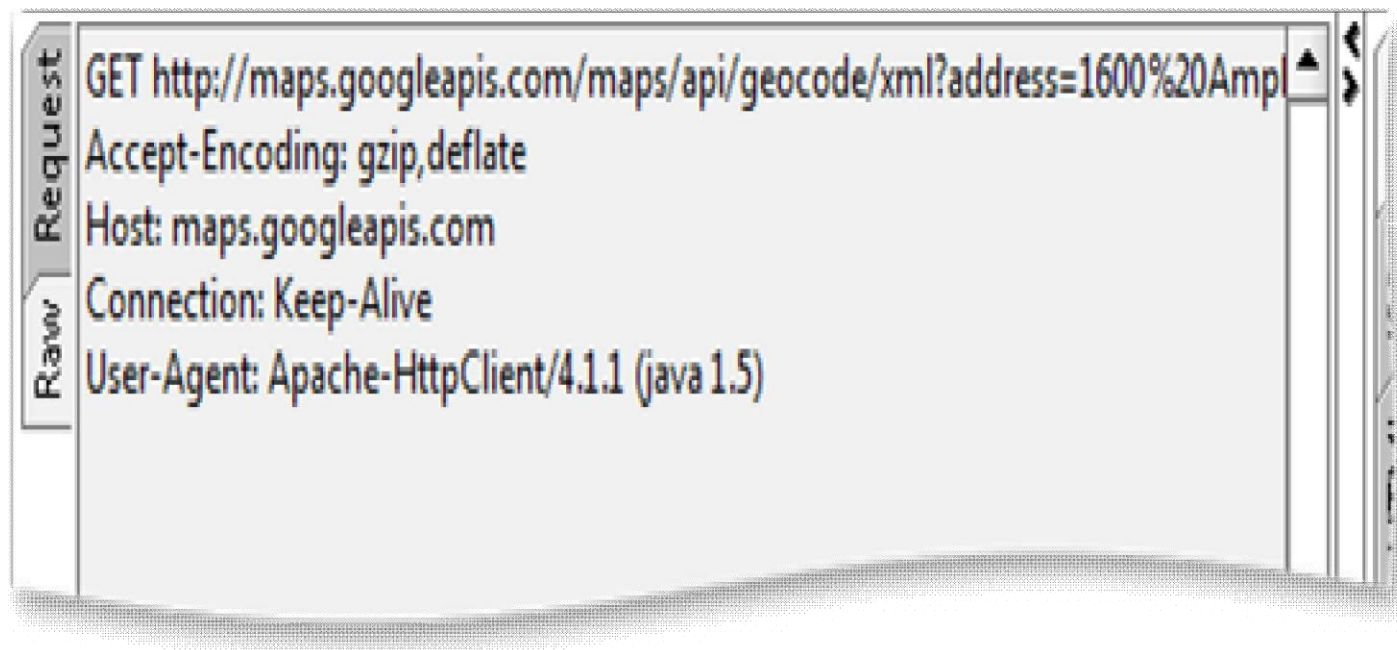
Remove..

Description:

Disable Encoding:

☐ Disables URL-Encoding of the parameter value

RAW: just as for SOAP Requests this shows the Raw bytes sent for the last request. After submitting a request it will contain something like:



312 .2 REQUEST MESSAGE TABS

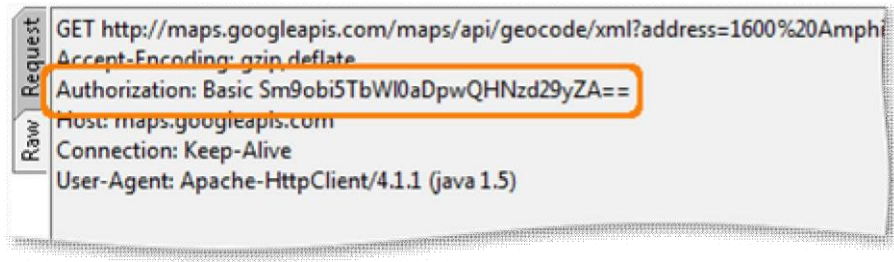
A number of tabs are available along the bottom of the Request Parameters View; let's have a look at them in order to see how they can be used:

- **Auth:** Allows you to specify HTTP Authentication information:



The screenshot shows the 'Authorization' dialog box in soapUI. At the top, there is a dropdown menu labeled 'Authorization:' with 'Basic' selected. Below this, there are four input fields: 'Username:' containing 'John.Smith', 'Password:' containing a series of dots, 'Domain:' which is empty, and 'Pre-emptive auth:' with two radio buttons. The first radio button, 'Use global preference', is selected. The second radio button is 'Authenticate pre-emptively'. At the bottom of the dialog, there is a tab bar with several tabs: 'Auth (Basic)', 'Headers (0)', 'Attachments (0)', 'Representations (0)', 'JMS Headers', and 'JMS Property (0)'. The 'Auth (Basic)' tab is highlighted with an orange border.

Specifying username and password will allow soapUI to authenticate with the service using Basic HTTP Authentication (if challenged by the server). If you want soapUI to send credentials directly without a challenge, then select the “Preemptive Authentication” option in the global HTTP Preferences. In this case you can see the credentials in the Raw message tab after sending:



Specifying a value in the Domain fields also allows soapUI to authenticate with NTLMv1 servers (NTLMv2 is not directly supported but can be accessed by using a third party tool like...).

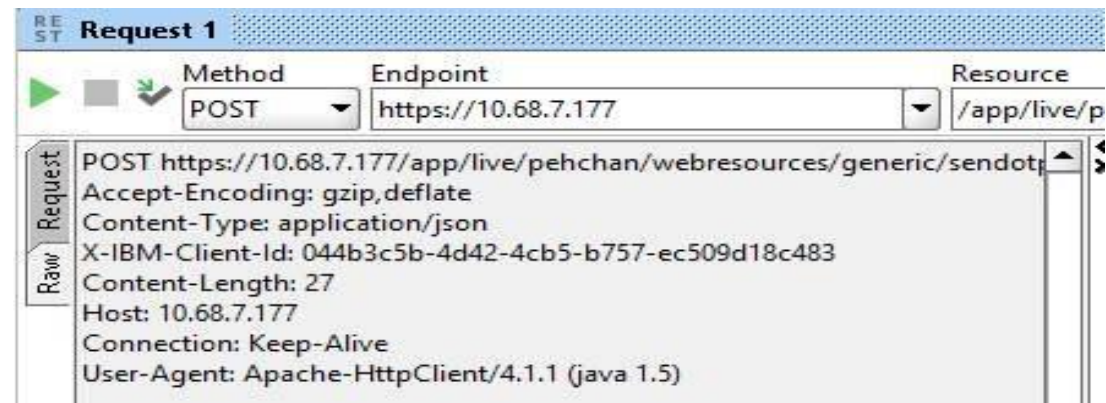
- **Headers:** Allows you to add arbitrary HTTP Headers you might want to include with your request in two ways, for example



Or



results in the Raw request tab.



- **Attachments:** Contains any files that should be attached to the request as MIME attachments. If you want to associate the content of a file with any of the parameters when simulating a HTML multipart/form-data form then specify that parameters value as "file:". For example:

The screenshot shows a web client interface with a 'Request' tab. It displays a table of request parameters and a section for attachments.

Name	Value	Style	Level
File1	file:hermes.log	QUERY	RESOURCE
File2	file:error.log	QUERY	RESOURCE

Below the table, the 'Media Type' is set to 'multipart/form-data' and the 'Post QueryString' checkbox is checked.

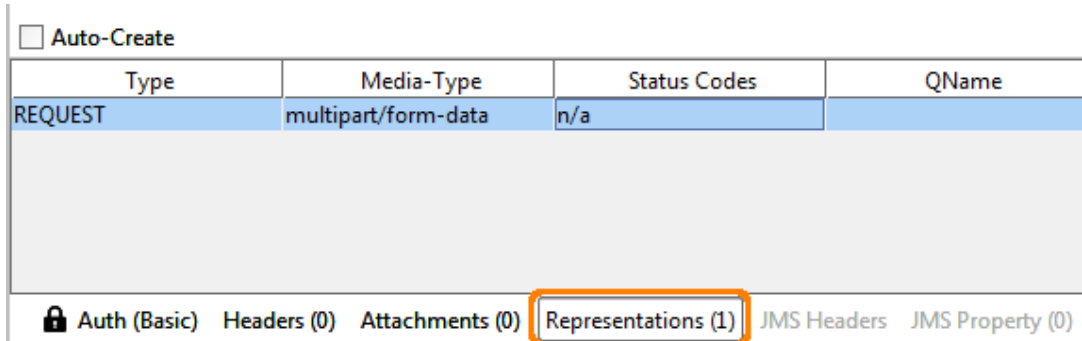
Name	Content type	Size	Part	Type	Conte...	Cached
error.log	application/octet-stream	38304		UNKNOWN	error.log	<input checked="" type="checkbox"/>

At the bottom, there are tabs for 'Auth', 'Headers (0)', 'Attachments (1)', 'Representations (0)', 'JMS Headers', and 'JMS Property (0)'.

Here you can see the request has two parameters both specifying file. The first just refers to a file in the file system ("hermes.log"), and

the second refers to an attachment (“error.log”). Setting the media -type to multipart/form -data and posting the query string will send the request as if it were a HTML Form with corresponding File input fields.

- **Representations:** Shows the defined request representations for the underlying REST Method:



The screenshot shows the SoapUI request editor interface. At the top left, there is a checkbox labeled 'Auto-Create'. Below it is a table with four columns: 'Type', 'Media-Type', 'Status Codes', and 'QName'. The first row of the table is highlighted in blue and contains the values 'REQUEST', 'multipart/form-data', 'n/a', and an empty cell. Below the table is a large, empty light gray area. At the bottom of the editor, there is a horizontal tab bar with several tabs: 'Auth (Basic)', 'Headers (0)', 'Attachments (0)', 'Representations (1)', 'JMS Headers', and 'JMS Property (0)'. The 'Representations (1)' tab is currently selected and is highlighted with an orange border.

Type	Media-Type	Status Codes	QName
REQUEST	multipart/form-data	n/a	

That's it for the request editor; let's have a look at the response editor as well.

3.2 RESPONSE MESSAGE VIEWS

The response editor contains several views for visualizing different types of responses;

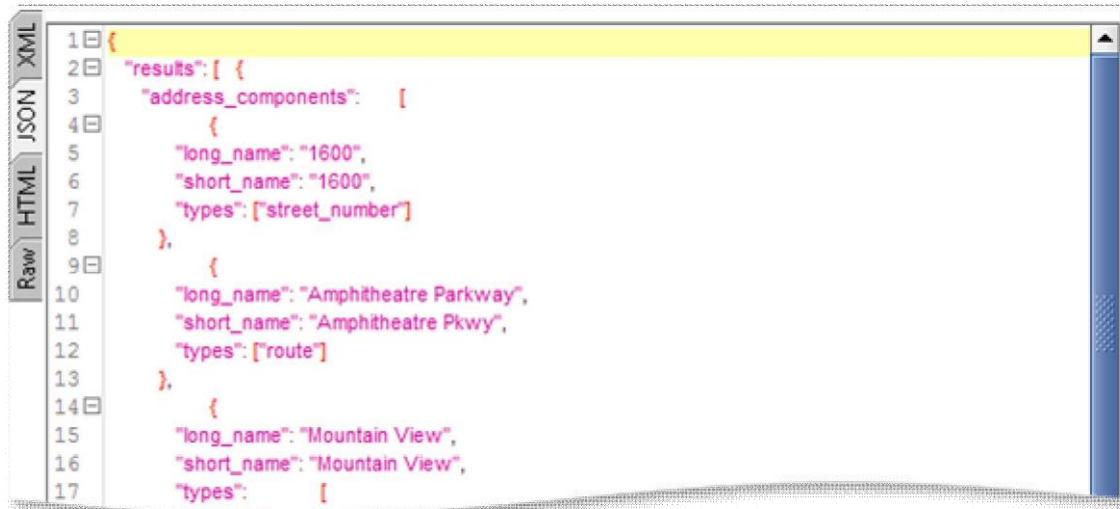
- **XML:** shows the current response in XML format. SoapUI will attempt to convert HTML responses to XML so they can be used in property transfers and scripts just like any other response messages in your functional tests. --> For example, here comes a HTML response rendered as XML:



The screenshot shows a web browser's developer tools window with the 'Raw' tab selected. The HTML source code is displayed, starting with the <html> tag. The <head> section includes a meta tag for 'HTML Tidy for Java', a title 'SoapUI - The Home of Functional Testing', and several meta tags for content type, charset, and caching. A JavaScript snippet is also visible, initializing a Google Analytics tracking object.

```
<html>
<head id="head">
  <meta content="HTML Tidy for Java (vers. 26 Sep 2004), see www.w3.org" name="generator">
  <!--Head Open-->
  <title>SoapUI - The Home of Functional Testing</title>
  <meta content="SoapUI, is the world leading Open Source Functional Testing tool for SOAP, REST, and Web Services" name="description">
  <meta content="text/html; charset=UTF-8" http-equiv="content-type"/>
  <meta content="no-cache" http-equiv="pragma"/>
  <meta content="text/css" http-equiv="content-style-type"/>
  <meta content="text/javascript" http-equiv="content-script-type"/>
  <meta content="soapui, soap ui, testing, soap, api, rest, soa , web services, soap t" name="keywords">
  <script type="text/javascript">var _gaq = _gaq || [];
    _gaq.push([['_setAccount', 'UA-92447-1']]);
    _gaq.push([['_setDomainName', 'none']]);
    _gaq.push([['_setAllowLinker', true]]);
    _gaq.push([['_setAllowHash', false]]);
    _gaq.push([['_trackPageview']]);
  </script>
</head>
```

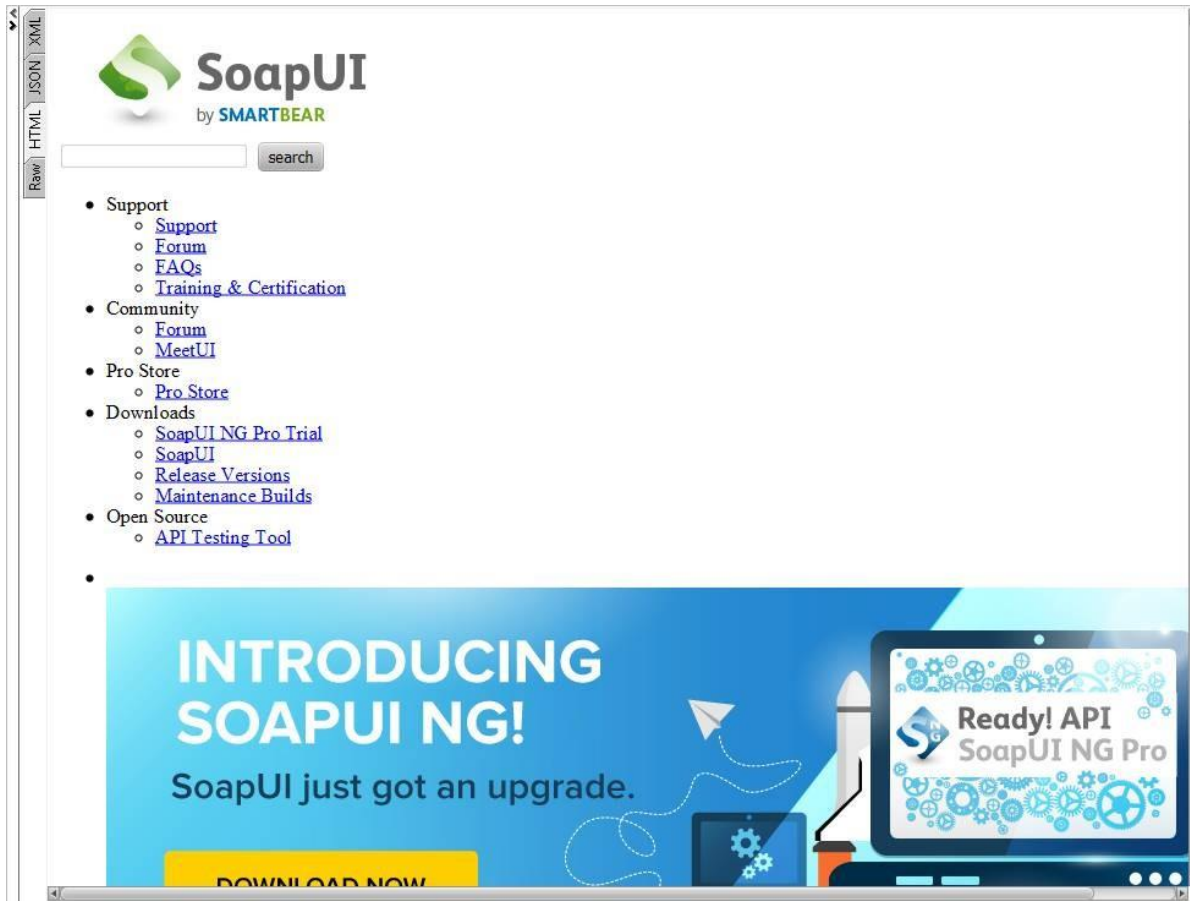
- **JSON:** shows a nicely-formatted rendering of the returned JSON response (if applicable):



The image shows a web browser window with the 'Raw' tab selected in the top-left corner. The tab bar also includes 'HTML', 'JSON', and 'XML'. The main content area displays a JSON response, which is a list of three address components. The first component is '1600' with type 'street_number'. The second component is 'Amphitheatre Parkway' with type 'route'. The third component is 'Mountain View' with type 'locality'. The JSON is formatted with syntax highlighting and line numbers on the left.

```
1 {
2   "results": [ {
3     "address_components": [
4       {
5         "long_name": "1600",
6         "short_name": "1600",
7         "types": ["street_number"]
8       },
9       {
10        "long_name": "Amphitheatre Parkway",
11        "short_name": "Amphitheatre Pkwy",
12        "types": ["route"]
13      },
14      {
15        "long_name": "Mountain View",
16        "short_name": "Mountain View",
17        "types": [
```

- **HTML:** renders the contents of an HTML response (if applicable), for example if we create a REST service for the SoapUI website and define the root page ("index.html") as a resource with a GET method we can see the results:



You can click in the page just as usual, pages that are shown in a new window will be opened with the current system browser instead.

Please note that the Browser component used by soapUI does not work on all platforms, in which case you will get a "Browser Component Disabled" message instead.

- **Raw:** shows the raw bytes received for the response. For the above HTML page this contains:

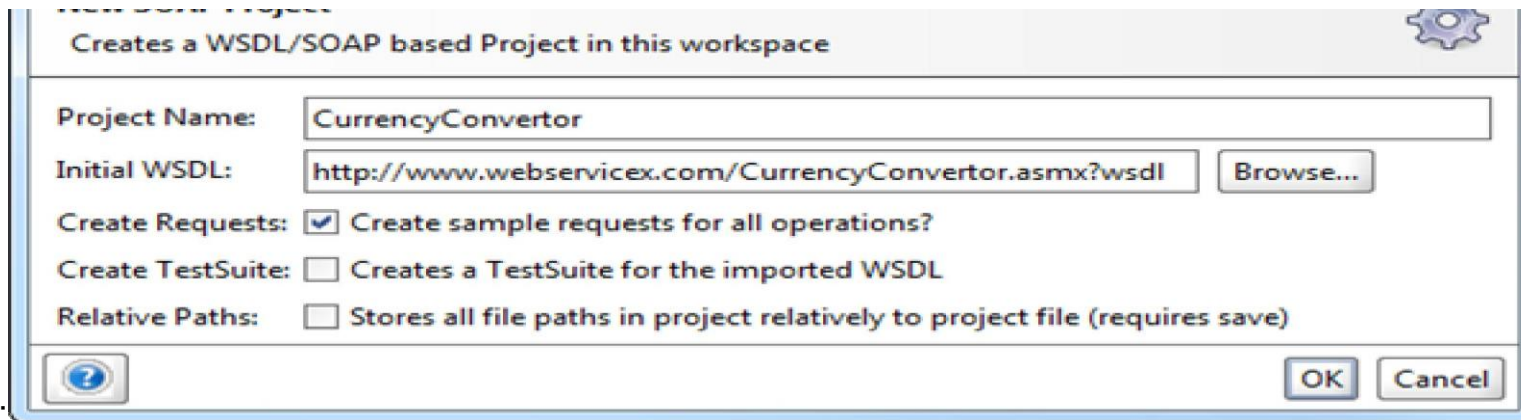


```
HTTP/1.1 200 OK
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Content-Type: text/html; charset=utf-8
Expires: -1
Server: Microsoft-IIS/7.0
Set-Cookie: CMSPreferredCulture=en-US; expires=Wed, 15-Jun-2016 13:47:06 GMT; path=/; HttpOnly
X-Frame-Options: SAMEORIGIN
Date: Mon, 15 Jun 2015 13:47:05 GMT
Content-Length: 189729

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="head"><!-- Head Open --><title>
    SoapUI - The Home of Functional Testing
</title><meta name="description" content="SoapUI, is the world leading Open Source Functional Testing tool for API
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
<meta http-equiv="pragma" content="no-cache" />
<meta http-equiv="content-style-type" content="text/css" />
<meta http-equiv="content-script-type" content="text/javascript" />
<meta name="keywords" content="soapui, soap ui, testing, soap, api, rest, soa , web services, soap testing, soa testing
<script type="text/javascript">
```

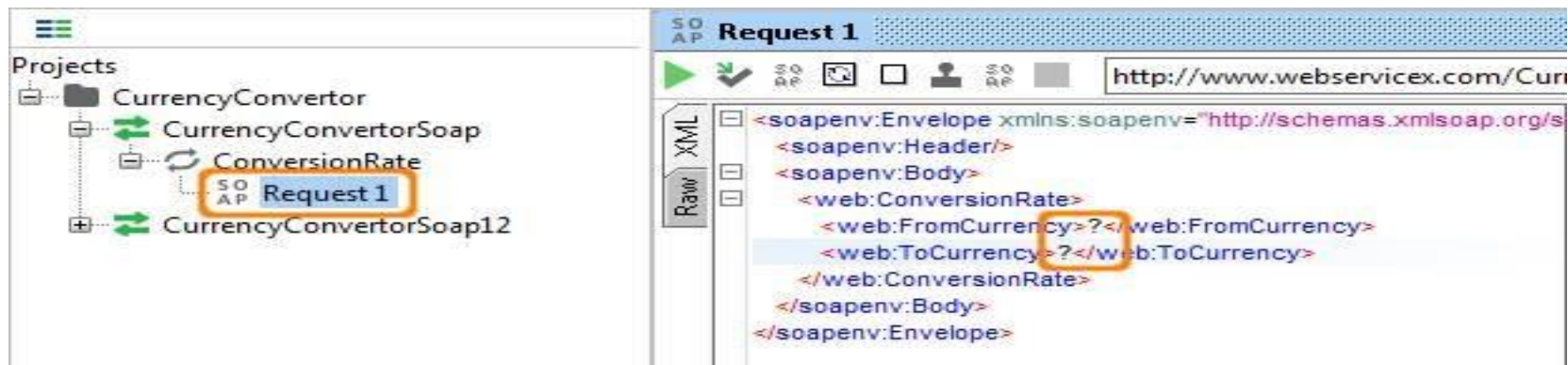
4 . TESTING SOAP SERVICE

Getting started with some ad-hoc testing of a SOAP service is straightforward; select the “New Project” option from the File menu, which will prompt the following



dialog:

Paste the WSDL path `http://www.webservicex.com/CurrencyConvertor.asmx?wsdl` into the Initial WSDL/WADL field (the Project Name will be extracted from this) and press OK. SoapUI will work a bit and create the project with the imported WSDL available in the navigator. Go straight to the first “Request 1” request generated for the ConversionRate operation and double-click it, which opens the following window:



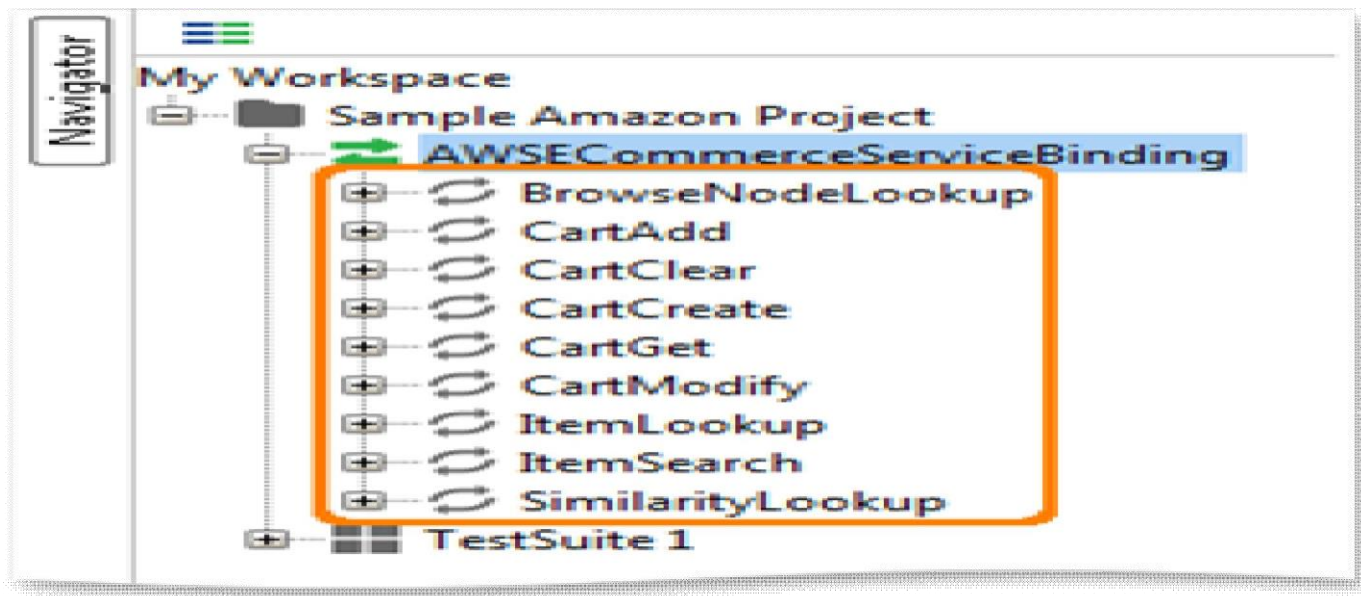
Now all you have to do is enter the codes for the desired currencies and press the green arrow on the top left to submit the request to the target service, which will return a nice response for



you:

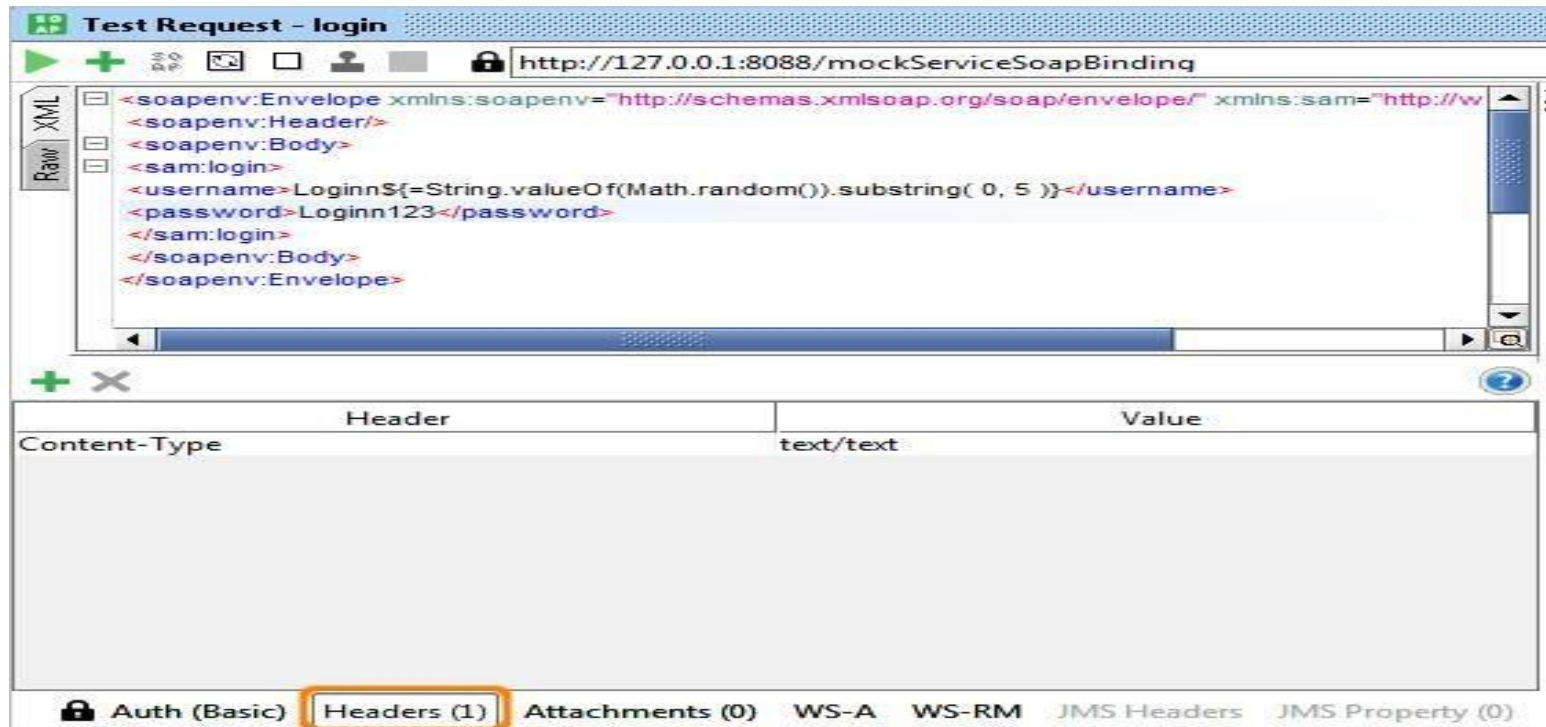
41 1 OPERATIONS

Each WSDL-based Service exposes a number of operations (conveniently named “operation” in the WSDL) that each have a request and response message format (both optional). In soapUI, the operations for a Service are shown as nodes under the Service node in the project navigator:

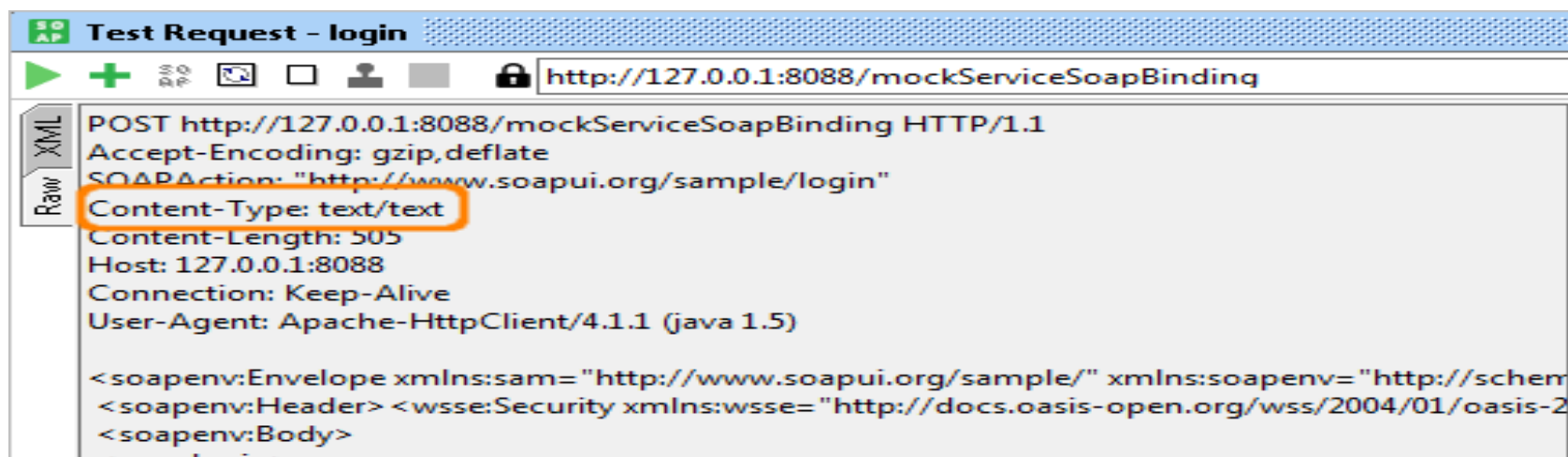


42 2 CUSTOM HTTP HEADERS

Adding custom HTTP Headers is straight -forward; the Headers inspector at the bottom of the XML editor allows for this:



Here we've added a custom Content-Type header which will override the standard Content -Type used for the SOAP Request ("text/xml; charset=utf -8"). Sending the request and looking the Raw Request Viewer reveals



You can of course add as many headers as required, and their value can contain property expansions as usual. The corresponding Headers tab for the response message not surprisingly shows all HTTP Headers in the response

The screenshot shows a web browser's developer tools window. The top pane displays an XML SOAP response. The XML structure is as follows:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsa:Action>asdasd</wsa:Action>
    <wsa:RelatesTo RelationshipType="http://www.w3.org/2005/08/addressing/related-to">
      <wsa:To>asdas</wsa:To>
      <wsa:MessageID>uuid:d8353dc2-eb07-4188-a73b-dae7e3</wsa:MessageID>
    </wsa:RelatesTo>
  </soapenv:Header>
  <soapenv:Body>
    <sam:loginResponse>
      <sessionId>43743284089152157</sessionId>
    </sam:loginResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The bottom pane shows the HTTP headers for the response:

Header	Value
Content-Length	313
#status#	HTTP/1.1 200 OK
Content-Encoding	gzip
Content-Type	text/xml; charset=utf-8
Server	Jetty(6.1.26)

At the bottom of the developer tools, there are tabs for "Headers (5)", "Attachments (0)", "SSL Info", "WSS (0)", and "JMS (0)". The "Headers (5)" tab is currently selected and highlighted with an orange border.