This File contains details about the CustomerAPI, SoapUI tool’s installation and about the automated test cases that have been written for the API.

**About the API**

This Restful API called “CustomerAPI” stores customer data which includes customer’s first name, last name, email address, date of birth and Phone number. This API has 4 methods - GET, POST, PUT and DELETE. Used Entity framework core as an In-memory database for .Net core 2.0

**Swagger URL for the API**

<http://appsvr.southcentralus.cloudapp.azure.com:8080/swagger/#/>

**How to run the Test Automation**

1. Download & install SOAPUI tool. Latest version available is 5.4.0.
2. Go to file -> Import Project and import the ***Customer-soapui-project.xml*** that has been committed in GitHub repository.
3. This should import the test suite including all my test cases.
4. Now double click on customer workflow , it should open up another window , now click on icon which will run all automated test cases flawlessly.

**TEST AUTOMATION COVERAGE AGAINST THE API**

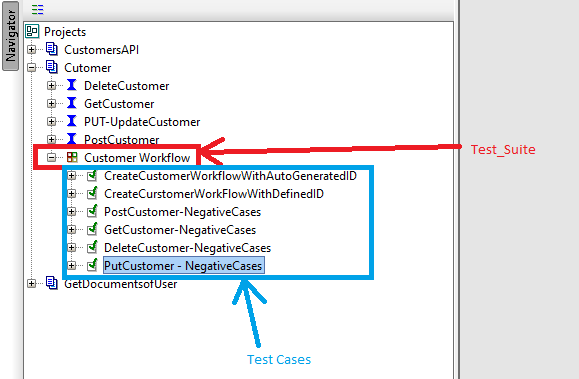
All Methods under API are converted into Parameterized Restful Interface in SOUPUI (i.e. GET, POST, PUT, DELETE), which can accept requests with different Parameters, followed by the Standards defined by its WSDL.

Test Suit Contains 6 Test Cases that covered all functionality provided by API

Test Cases are as follows:

1. CreateCustomerWorkflowWithAutoGeneratedID
2. CreateCurstomerWorkFlowWithDefinedID
3. PostCustomer-NegativeCases
4. GetCustomer-NegativeCases
5. DeleteCustomer-NegativeCases
6. PutCustomer – NegativeCases

Screen Shot reflects the above statement is as below.

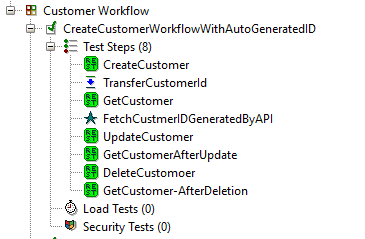


1. **Test Case 1>> CreateCustomerWorkflowWithAutoGeneratedID**

This test case verifies all the positive flows provided by the API, i.e. Create\_Customer, Update\_Customer, Get\_Customer, Delete\_Customer.

Below are the Functionality covered/verify under this Test Case.

1. **CreateCustomer :-** This Test Step send the valid request to API with id=0, so that Autogenerated CustomerId can be generated by API and returns as a response with other details.
2. **GetCustomer :-** This step sends the CustomerID returned in Above/previous Response and verifies all details with the help of Assertion.
3. **UpdateCustomer :-** this step sends the updated json as a request to update the customer details and verifies send details in received response.
4. **GetCustomerAfterUpdate :-** Again it sends the same CustomerID to verify the updated details are being returned by API response.
5. **DeleteCustomer** :- This step sends the customerid as a request to delete that entry and verifies the response code i.e. 204.
6. **GetCustomer-AfterDeletion :-** Again its sends the customerid to verify that entry is now deleted from DB, and expect the response code, 404 https code in response for this request.

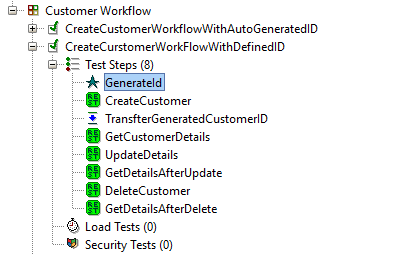


1. **Test Case 2 >> CreateCurstomerWorkFlowWithDefinedID**

Update\_Customer, Get\_Customer, Delete\_Customer but with the define CustomerID is sent in Post Request to Create a Customer on a specific customer ID passed as an request.

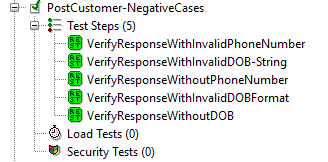
Below are the functionalities covered/verified under this Test Case.

1. GenerateId :- This step generates the CustomerID and is passed as a parameter to other followed Steps.
2. CreateCustomer
3. GetCustomerDetails
4. UpdateDetails
5. GetDetailsAfterUpdate
6. DeleteCustomer
7. GetDetailsAfterDelete



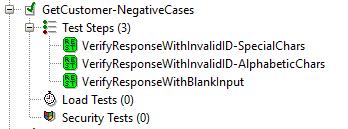
1. **Test Case 3 >> PostCustomer-NegativeCases**

This test case will verify response of API of several Negative Requests, Define below.



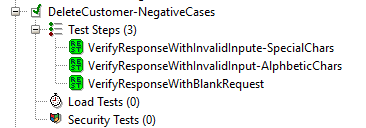
1. **Test Case 4 >> GetCustomer-NegativeCases**

This test case will verify response of API for some negative scenarios: -



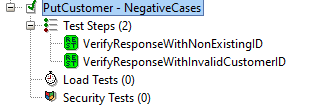
1. **Test Case 5 >> DeleteCustomer-NegativeCases**

This test case will verify response of API for some negative scenarios: -



1. **Test Case 6 >> PutCustomer – NegativeCases**

This test case will verify response of API of several Negative Requests, Define below



**Execution of All Test Suit is as below.**

Where Green Color on Test Step Reflects that Response comes for this request is as expected and written under their Assertion Logic to verify the Response content.

