API Testing: Picking the Right Strategy

Asha KR

Asha_kr@mcafee.com

Shwetha D J

Shwetha_naik@mcafee.com

Abstract

A right testing strategy for **Application Programming Interface (API)** is crucial for developing a successful product. However, many businesses fail to see a complete picture during project management phase. Changes in product and project requirements are inevitable, which can potentially disrupt the timelines or quality of the product. Eventually, more time and resources are required to accommodate the change requests. It is challenging to deliver a good quality product to the end users in such situations. Testing of API after it integrated with the UI is not an easy task especially when we deliver a quality software as a service to our customers and partners.

API driven testing is a testing framework that uses a programming interface and the application to validate the behavior under test. Typically API driven testing overcomes application user interface altogether. API driven testing is also being widely used by software testers as it serves additional benefits as compared to other testing strategies. Programmers or testers write scripts using a programming or scripting language that calls interface exposed by the application under test. These interfaces are custom built or commonly available interfaces like COM, HTTP. Present development world changes every day with new technologies and this is directly proportional to UI changes. In this era we cannot rely only on UI testing for functional verification. The test scripts created are executed using an automation framework to compare test results with expected behavior of the API. The journey was more challenging while bringing up a startup project to a stabilized API automation framework with fair number of test cases automated in it. There were many initiatives and ideas put forth for converting manual testing to stabilized automation testing.

Outcome of API automation is increased test coverage by 100% with quality deliverables. 20% of early detection of defects and scope for manual testers increased to become Test Automation Engineers. There are nightly builds running with automation for Build Validation and Functional Validation, which has helped testers pitching in early defect analyses with in less time.

Biography

***Asha KR** is a Senior Software QA Engineer at Intel Security, currently working in the Intel Security India Center in Bangalore. She has been working for the past 5+ years in different QA roles SaaS products. Asha holds Bachelor of Engineering in E&C from VTU, Karnataka, India.*

***Shwetha D J** is a Software Development Engineer for Test, currently working in the Intel Security India Center in Bangalore. She has been working for the past 1+ years on SaaS products. Shwetha holds Bachelor of Engineering in E&C from VTU, Karnataka, India.*

1. Introduction

Web services extend the World Wide Web infrastructure to facilitate a software to connect to other software applications. Web services (sometimes called as application services) are services usually including a combination of the software program and data, but possibly including human resources as well that are made available from a business's Web server for Web users or other Web-connected programs. Applications access Web services via Web protocols and data formats such as HTTP, XML, and SOAP. Web services combine the best aspects of component-based development and the Web. They are a cornerstone of the Microsoft .NET programming model.

In this era of **agile development**, we are moving to a model where UI is ever changing and the consumption of web services is taking the center stage and hence giving this layer the attention it truly deserves by shifting the onus from UI testing to Web Services based validation.

API stands for **Application Programming Interface**, which specifies how one component should interact with the other. It consists of a set of routines, protocols and tools for building the software applications. An API is similar to **user interface**, the difference is that, instead of a user-friendly collection of windows, dialog boxes, buttons, and menus, the API consists of a set of direct software links, or calls, to lower-level functions and operations. APIs can look formidable, but they're designed to be accessible to trained, knowledgeable programmers.

API testing in many respects is like testing software at the user-interface level, however instead of testing by means of standard user inputs and outputs, the testers use software to send calls to the API, get output, and log the system's response. General steps involved while performing API testing are mentioned below:

- Details of API information is found in Specification Document which lists the signatures of each API function (the input parameters, the function or method name, and the return type)
- Identify the software to be used for API testing
- Add the web service call to be tested
- API call will have request and response parameters
- Prepare the inputs for the request
- Invoke the method with all provided inputs
- Analyze the output response

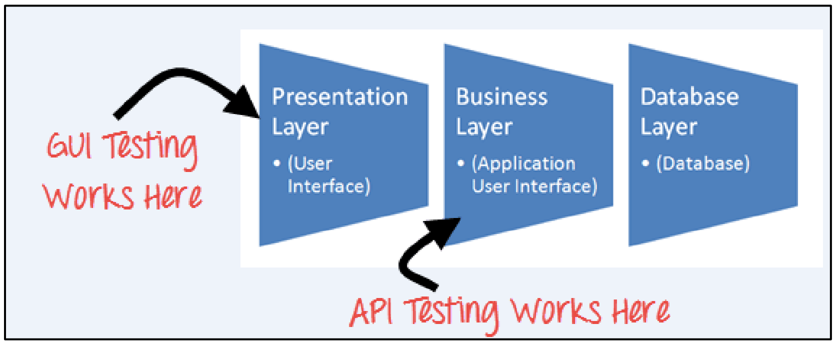


Fig 1.1: API Testing in Business Layer

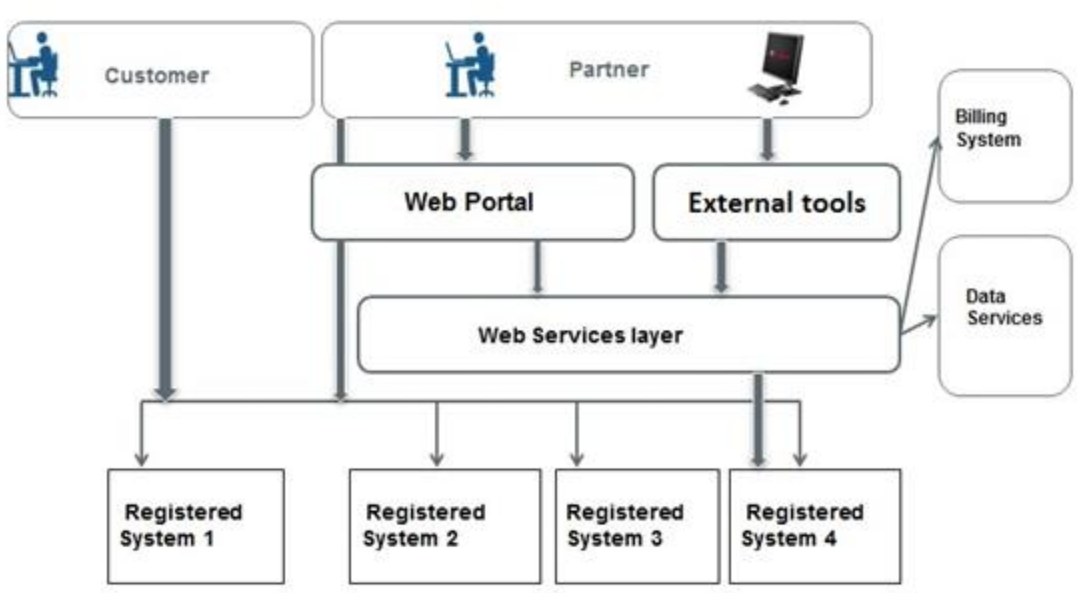


Fig 1.2: Web Services and API Representation

The above diagram Fig (1.2) represents architecture of where the Web services, API come into the big picture of the solution under test. Our team is responsible for development of Web Services. There is a **Web Services layer** that hosts all the web services. **External tools** block is one in which the methods are present and they can be accessed from Web Services layer. Web Service layer consists of two sub layers:

1. **Data Service Layer**: This layer consists of all Data related APIs which provide the business logic for the application. This layer also in turn communicates to the data source or the data layer.

2. **Management Service Layer**: This layer consists of Management related APIs which provide the management solutions to the applications that consumes the corresponding services in this layer.

User requests reaches management service layer, which communicates to data service layer, in turn it fetches data from data source. Each layer has multiple web services and each web service has multiple methods called APIs. Different systems access these API's for their business purpose. **Web portal** is the UI layer which consumes all the web services and presents the relevant data to the end user. Partners can make use of external tools to directly invoke the APIs.

2. Problem Statement

Testing of an application which integrates multiple systems is not an easy task. When we promised to deliver flawless product to customer, detailed testing of the individual components, systems and subsystems that integrates to main product is very crucial. If we are planning to test the end to end scenarios that spans across all the systems and subsystems in consideration it may lead to product which is not impeccable. If some functionality fails at UI then analysis of failure at different system is more time consuming. Having API testing one step before the UI can help in managing different systems without a glitch. However when we start to test multiple different systems as one, lot of manual effort and time is required and also maintaining the product will eventually become tiresome.

When products grow more complex and interacts with external systems, we cannot rely on manual testing because of human errors and a natural tendency to miss few regression test scenarios or some feature validations. The strategy that we followed should suffice all the requirements mentioned above with the effective outcome.

2.1 Challenges in API Testing

- Main challenges in API testing are Parameter Combination, Parameter Selection, and Call Sequencing
- There is no GUI available to test the API which makes it difficult to provide input values
- Validating and Verifying the output in different system is not very easy for testers
- Parameters selection and categorization required to be known to the testers
- Exception handling function needs to be tested
- Coding knowledge is necessary for testers

3. Approach

Considering the problems that arise during API testing, choosing an approach for an effective and efficient API testing depends on a lot of things. Will the API be a public API that will be consumed by some external users/systems, or is it a part of a larger product's infrastructure? API is a general term that is sometimes used to describe anything from a COM interface, to a DLL or JAR you can reference, to a REST web service. Different approaches can be applied to testing these different categories.

3.1 Points to keep in mind while performing API testing

Below flowchart represents the steps to consider while performing API testing.

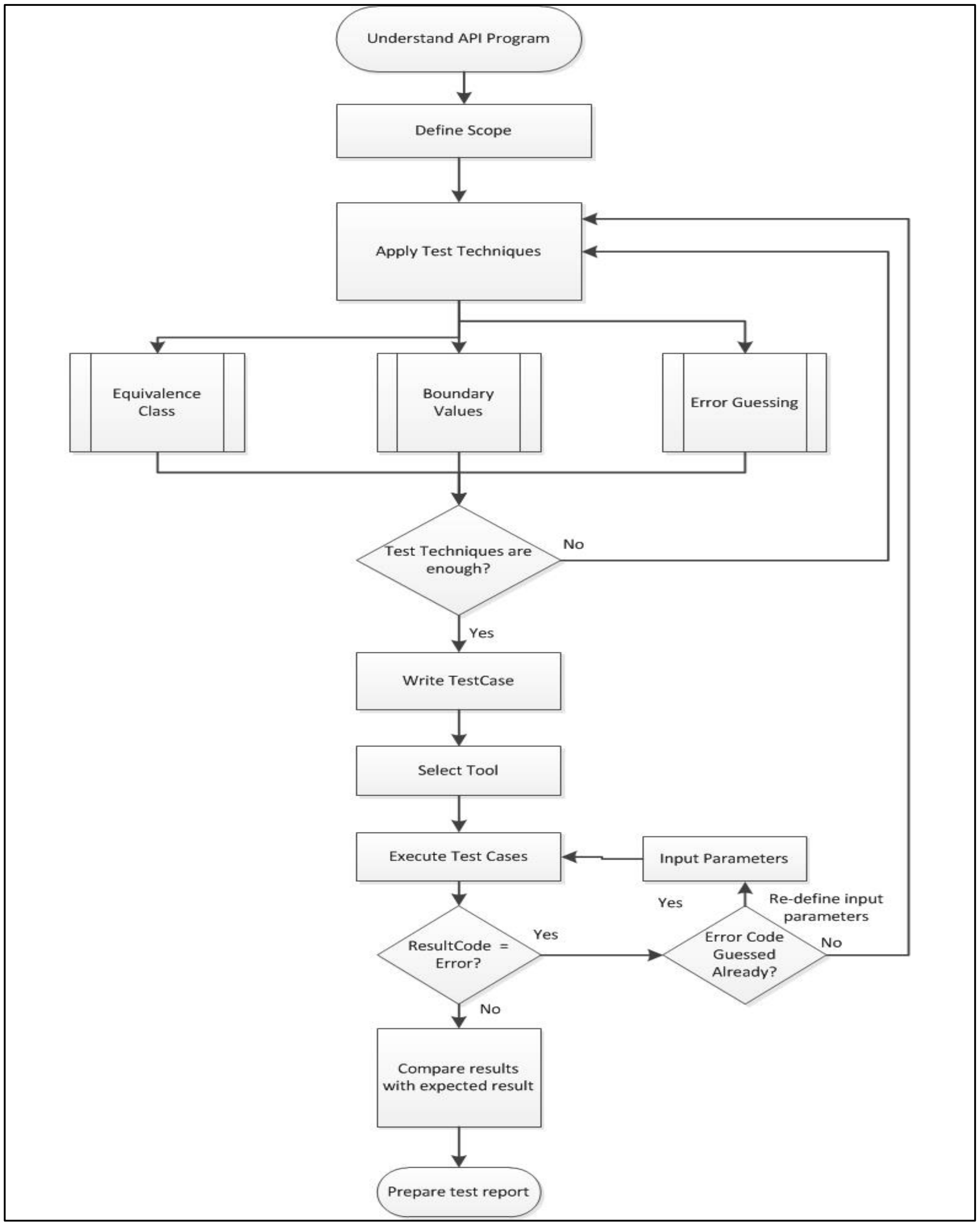


Fig 3.1: Flowchart depicting the API Testing steps

- To know how the API works, it is important to understand the functionality of an API and clearly define the scope of the program.
- Apply testing techniques such as equivalence classes, boundary value analysis and error guessing and write test cases for the API.
- Input Parameters for the API need to be planned and defined appropriately.
- Execute the test cases and compare expected and actual results.
- Use the appropriate tool to test API which gives more effective results

When we understand and realize the importance of Web Services Automation, the next challenge is to select the right Test approach and Test Type before jumping into automation design and implementation.

3.2 Test Objective

It is important to answer the basic question "Why do we want to do Web Services Testing?" and that will help us to decide the right kind of Test we should choose to automate them. Let's begin with few possible answers

- To validate the functional behavior of your application / APIs
- To validate the performance aspect of your application / APIs
- To first validate the functionality and then also test the performance aspect of it.

Once the objective of the test is clear, we need to look at the technologies that can make a difference to the way we test the Web Services.

3.3 API testing Manual

Manual API testing is one where in which testers needs to give inputs and invoke the method for response, analyze the results and report them.

E.g. AuthenticateSession API method in SessionData Service.

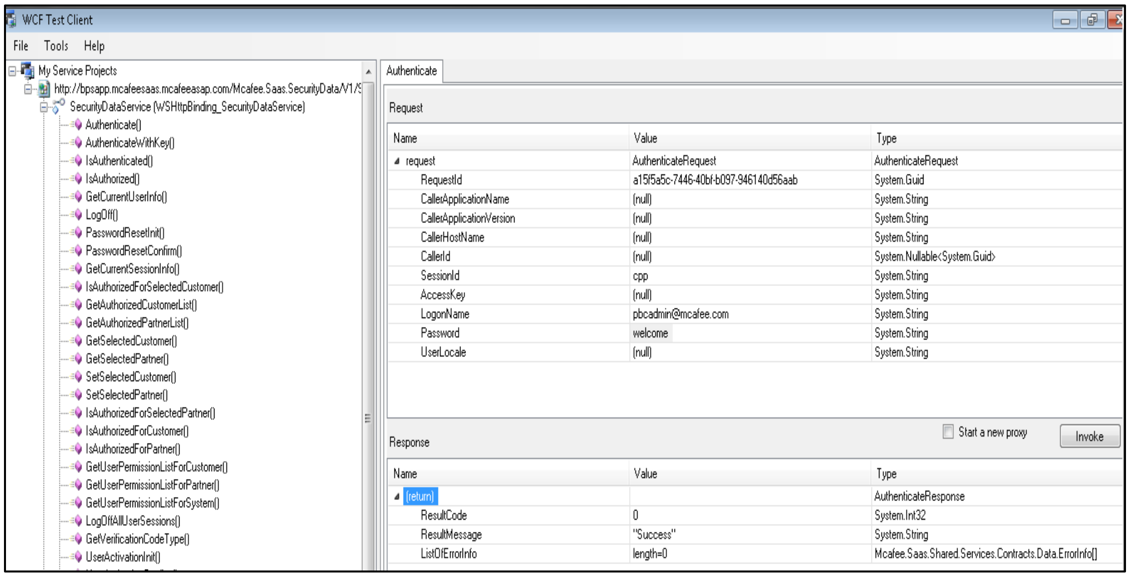
Inputs for AuthenticateSession method are: SessionId, Username and Password. Output for this call will be 'Success' or 'Fail' [With fail reason as message]
There are many steps involved in execution of API methods manually, which also consumes lot of effort and time. Identifying the tools to suit the testing is also one of the challenge. There are many tools available for executing manual testing such as

3.3.1 WCF (Windows Communication Foundation) Client
3.3.2 SOAP UI etc.

WCF is the tool which comes with Microsoft .net framework and user friendly. **Soap UI** is the open source cross platform functional testing solution with an easy to use graphical interface. Soap UI allows easy and rapid creation and execution of test cases.

3.3.1 WCF Client

Windows Communication Foundation (WCF) Test Client (WcfTestClient.exe) is a GUI tool that enables users to input test parameters, submit that input to the service, and view the response that the service sends back. It provides a seamless service testing experience when combined with WCF Service Host



My Service Projects
http://bpsapp.mcafeesaas.mcafeeasap.com/Mcafee.Saas.SecurityData/V1/S
SecurityDataService (WSHttpBinding_SecurityDataService)
- Authenticate()
- AuthenticateWithKey()
- IsAuthenticated()
- IsAuthorized()
- GetCurrentUserInfo()
- LogOff()
- PasswordResetInit()
- PasswordResetConfirm()
- GetCurrentSessionInfo()
- IsAuthorizedForSelectedCustomer()
- GetAuthorizedCustomerList()
- GetAuthorizedPartnerList()
- GetSelectedCustomer()
- GetSelectedPartner()
- SetSelectedCustomer()
- SetSelectedPartner()
- IsAuthorizedForSelectedPartner()
- IsAuthorizedForCustomer()
- IsAuthorizedForPartner()
- GetUserPermissionListForCustomer()
- GetUserPermissionListForPartner()
- GetUserPermissionListForSystem()
- LogOffAllUserSessions()
- GetVerificationCodeType()
- UserActivationInit()

Request

Name	Value	Type
request	AuthenticateRequest	AuthenticateRequest
RequestId	a15f5a5c-7446-40bf-b097-946140d56aab	System.Guid
CallerApplicationName	(null)	System.String
CallerApplicationVersion	(null)	System.String
CallerHostName	(null)	System.String
CallerId	(null)	System.Nullable<System.Guid>
SessionId	cpp	System.String
AccessKey	(null)	System.String
LogonName	pbcadmin@mcafee.com	System.String
Password	welcome	System.String
UserLocale	(null)	System.String

Response

Name	Value	Type
(return)		AuthenticateResponse
ResultCode	0	System.Int32
ResultMessage	"Success"	System.String
ListOfErrorInfo	length=0	Mcafee.Saas.Shared.Services.Contracts.Data.ErrorInfo[]

Fig 3.3.1: WCF Client

The Fig (3.2.1) shows a WCF client window, where left pane represents the Web service requests and the API methods included. Right top part represents the request with input parameters and down part is the response of the API invoked.

3.3.1.1 Advantages

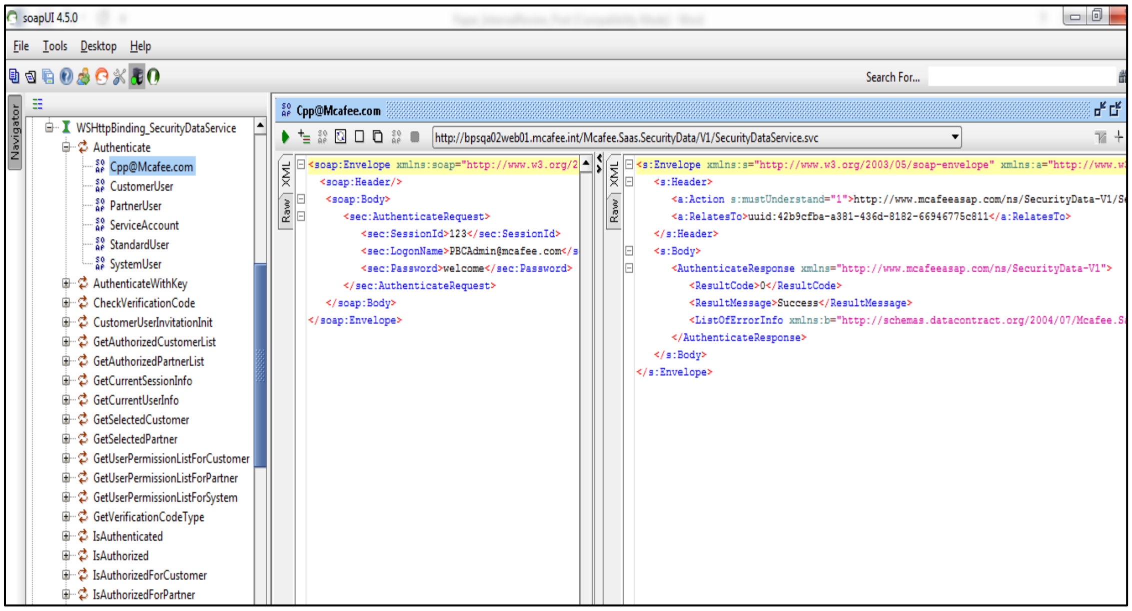
- **Interoperability**: A single platform used to exchange information using various network protocols and platforms. Typically, an Asp.Net web service uses HTTP to communicate between client and server. Similarly, in .Net Remoting the client and the server must use .Net applications to share or exchange information. WCF services are interoperable, which uses a variety of network protocols such as HTTP, TCP and MSMQ etc.
- **Security and Reliability:** WCF provides better security and reliability as compared to web services or ASMX services. Security is a key element in any Service Oriented Architecture (SOA), and it is provided in the form of auditing, authentication, authorization, confidentiality and integrity of messages shared between the client and the service.
- **Support for Plain XML, Ajax and REST:** WCF libraries now support other formats for sharing or exchanging messages (data) across the network. We can now configure WCF to share plain XML messages between clients and services, formats that are no more controlled by SOAP anymore. We can now build WCF services using **REST**, also known by the name Representational State Transfer. It is simply an architecture to design distributed applications on a network, where clients can make requests for services.

3.3.1.2 Disadvantages

- The Service Host instance is not shut down gracefully – thus any pending requests are aborted when the WCF Service Host is closed.
- Cannot customize Service Host initialization – and it is common to programmatically initialize fixed behaviors and provide exception handling.
- The WCF Test Client does not provide a way to save and reload inputs to service operations – thus must retype those inputs each time you run the test client.
- The user interface for supplying test values is automatically generated and cumbersome to use.

3.3.2 SOAP UI

SOAP stands for **Simple Object Access Protocol**. A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this. SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.



http://bpsqa02web01.mcafee.int/Mcafee.Saas.SecurityData/V1/SecurityDataService.svc

```
<soap:Envelope xmlns:soap="http://www.w3.org/2
  <soap:Header/>
  <soap:Body>
     <sec:AuthenticateRequest>
        <sec:SessionId>123</sec:SessionId>
        <sec:LogonName>PBCAdmin@mcafee.com</s
        <sec:Password>welcome</sec:Password>
     </sec:AuthenticateRequest>
  </soap:Body>
</soap:Envelope>
```

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://www.w
   <s:Header>
      <a:Action s:mustUnderstand="1">http://www.mcafeeasap.com/ns/SecurityData-V1/S
      <a:RelatesTo>uuid:42b9cfba-a381-436d-8182-66946775c811</a:RelatesTo>
   </s:Header>
   <s:Body>
      <AuthenticateResponse xmlns="http://www.mcafeeasap.com/ns/SecurityData-V1">
         <ResultCode>0</ResultCode>
         <ResultMessage>Success</ResultMessage>
         <ListOfErrorInfo xmlns:b="http://schemas.datacontract.org/2004/07/Mcafee.S
      </AuthenticateResponse>
   </s:Body>
</s:Envelope>
```

Fig 3.3.2: SOAP UI

3.3.2.1 Advantages

- SOAP is a Communication protocol designed to communicate via Internet.
- Extend HTTP for XML messaging.
- Provides data transport for Web services.
- Can exchange complete documents or call a remote procedure.
- Can be used for broadcasting a message.
- Platform- and language-independent.
- XML way of defining what information is sent and how.
- Enables client applications to easily connect to remote services and invoke remote methods.

3.3.2.2 Disadvantages

- The SOAP specification contains no mention of security facilities.
- SOAP specification does not specify a default encoding for the message body. There is an encoding defined in the spec, but it is not required that you use this encoding to be compliant: Any custom encoding that you choose can be specified in the encoding Style attribute of the message or of individual elements in the message.
- Because SOAP deals with objects serialized to plain text and not with stringified remote object references (interoperable object references, IORs, as defined in CORBA), distributed garbage collection has no meaning.
- SOAP clients do not hold any state references to remote objects

3.4 API testing using Automation

Automation approach for API Testing: When we need to test a feature multiple times because of frequent modification in the dependent subsystems, then automation is the best of testing. **Regression testing** is made easy with automation. Automation can be done with some tools or if someone is very good in programming can do API automation without any tool. This is explained in below sections

3.4.1 Soap UI
3.4.2 Microsoft Web Test
3.4.3 Using Programming Language

3.4.1 SOAP UI

Automation to execute test suites developed using Soap UI, without using the user interface of SoapUI, 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 in common that they are good at triggering anything from a command line prompt. This could be a bat script in Windows, a shell script in UNIX, or it could be Maven project in a Java build environment.

Command line: Command line tools are good if you want to be able to execute something from a script that you have created yourself. SoapUI has support for executing from a command line. Execute it like this:

```
Testrunner project_file
```

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

Automate the execution after creating your SoapUI functional tests with test suites and connecting SoapUI tests to your build tool, test suite will get executed, as soon as a change has occurred.

3.4.1.1 Advantages

- We 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.
- Delivery of high quality APIs is assured, compared to when testing the services with manual effort.

3.4.1.2 Disadvantages

- The SOAP specification contains no mention of security facilities.
- SOAP specification does not specify a default encoding for the message body. There is an encoding defined in the spec, but it is not required that you use this encoding to be compliant: Any custom encoding that you choose can be specified in the encoding Style attribute of the message or of individual elements in the message.
- Because SOAP deals with objects serialized to plain text and not with stringified remote object references (interoperable object references, IORs, as defined in CORBA), distributed garbage collection has no meaning.
- SOAP clients do not hold any state references to remote objects

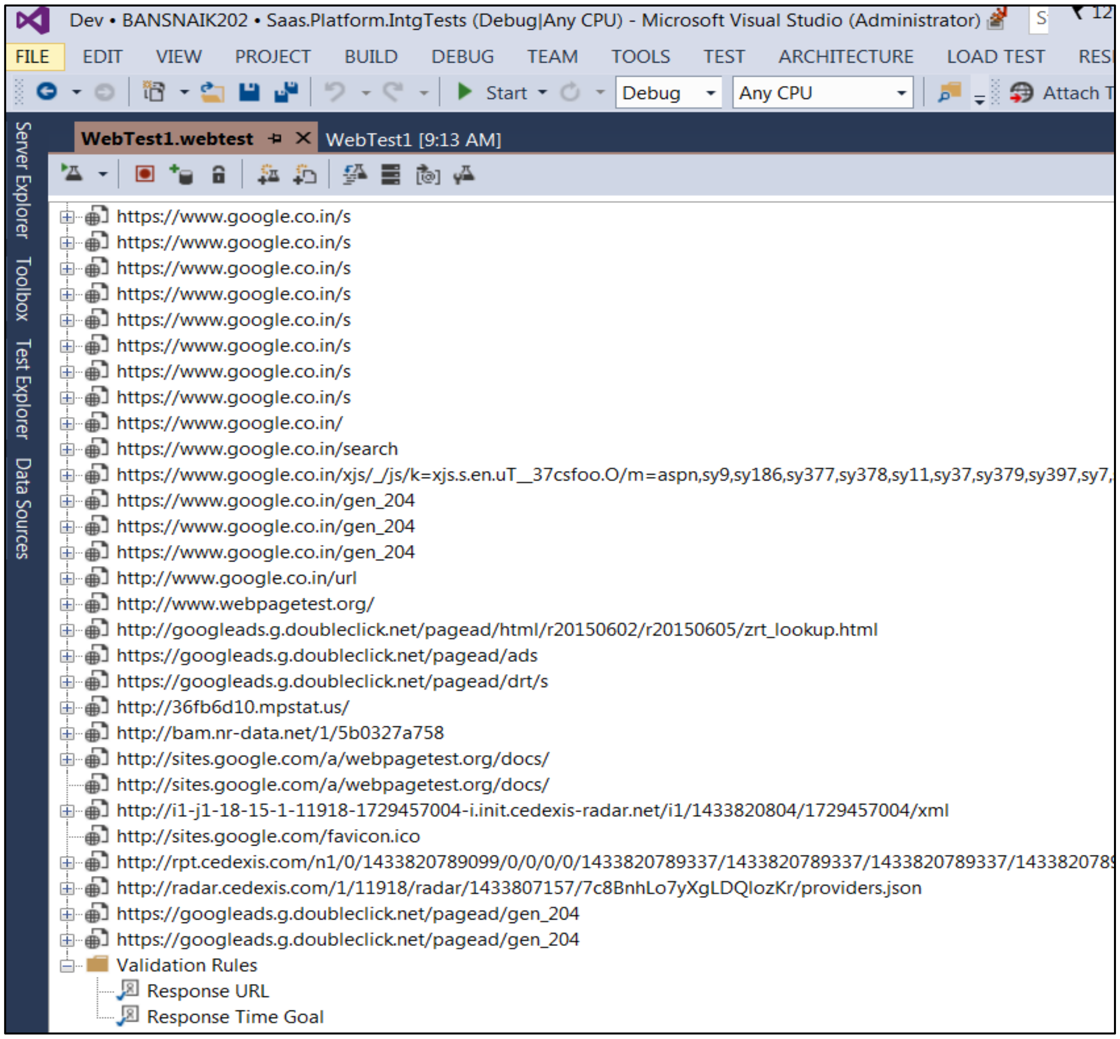
3.4.2 Microsoft Web Test

Web tests allow simulating a user performing a set of operations – typically a defined use case – on ASP.NET Web application, and validating the responses to see if the application is working as expected.

Web test is a better when we have relatively simpler web services where our objective is to test simpler / straightforward functionality and primary focus is to test their **performance** down the line. We are working on legacy app that is based on ASMX kind of services which have a better support in Web Test.

Steps to write Web Test: Web test can be created from Microsoft visual studio.

- Start Visual Studio as Administrator.
- Go to the Test menu and select New Test.
- From the list of project templates, select the Web Performance Test template
- When the project is created a new IE window will open and using a plug-in we can record the session.
- Browse a number of pages of the publishing portal and then press the Stop button. While you are browsing, notice how the requests will be listed in the Web Test Recorder plug-in.
- Press the Stop button and the requests will be added to your Web Performance Test.
- Press the Run Test Button in Visual Studio to run the recorded test.
- Once test run completes double click on the test name to see details.
- By using validation rules, we can determine what conditions are necessary for a test to pass. For example, we can consider that all requests that take longer than 5000 Milliseconds have failed.

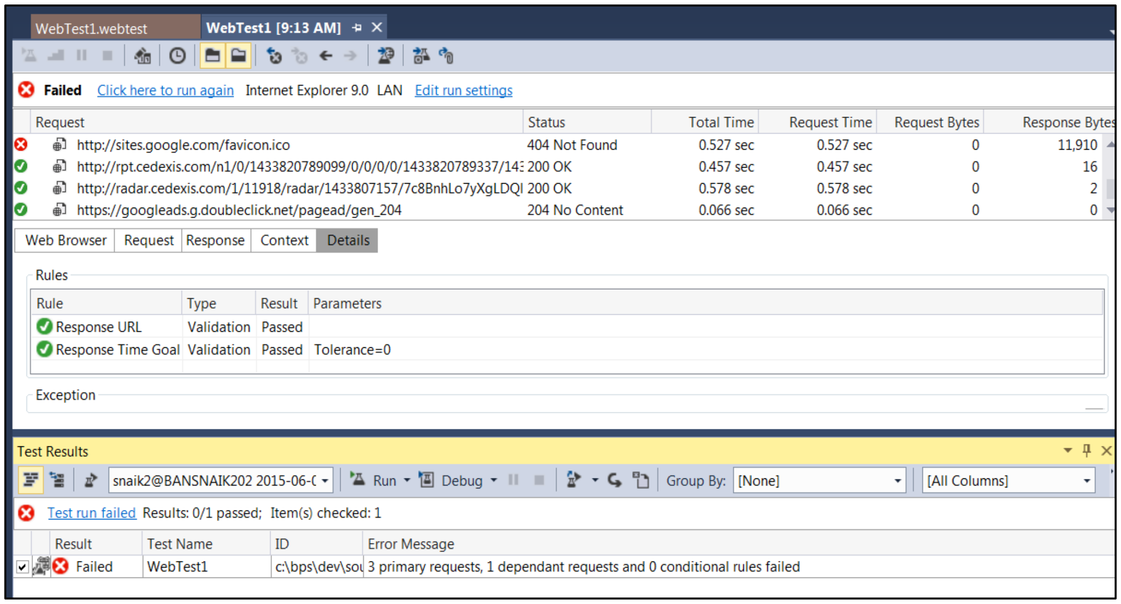

Dev • BANSNAIK202 • Saas.Platform.IntgTests (Debug|Any CPU) - Microsoft Visual Studio (Administrator)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE LOAD TEST
Start Debug Any CPU Attach
Server Explorer Toolbox Test Explorer Data Sources
WebTest1.webtest
WebTest1 [9:13 AM]

```
https://www.google.co.in/s
https://www.google.co.in/s
https://www.google.co.in/s
https://www.google.co.in/s
https://www.google.co.in/s
https://www.google.co.in/s
https://www.google.co.in/s
https://www.google.co.in/s
https://www.google.co.in/
https://www.google.co.in/search
https://www.google.co.in/xjs/_/js/k=xjs.s.en.uT__37csfoo.O/m=aspn,sy9,sy186,sy377,sy378,sy11,sy37,sy379,sy397,sy7,
https://www.google.co.in/gen_204
https://www.google.co.in/gen_204
https://www.google.co.in/gen_204
http://www.google.co.in/url
http://www.webpagetest.org/
http://googleads.g.doubleclick.net/pagead/html/r20150602/r20150605/zrt_lookup.html
https://googleads.g.doubleclick.net/pagead/ads
https://googleads.g.doubleclick.net/pagead/drt/s
http://36fb6d10.mpstat.us/
http://bam.nr-data.net/1/5b0327a758
http://sites.google.com/a/webpagetest.org/docs/
http://sites.google.com/a/webpagetest.org/docs/
http://i1-j1-18-15-1-11918-1729457004-i.init.cedexis-radar.net/i1/1433820804/1729457004/xml
http://sites.google.com/favicon.ico
http://rpt.cedexis.com/n1/0/1433820789099/0/0/0/0/1433820789337/1433820789337/1433820789337/143382078
http://radar.cedexis.com/1/11918/radar/1433807157/7c8BnhLo7yXgLDQlozKr/providers.json
https://googleads.g.doubleclick.net/pagead/gen_204
https://googleads.g.doubleclick.net/pagead/gen_204
Validation Rules
    Response URL
    Response Time Goal
```

Fig 3.4.2a: Web Test

If we don’t want add web test by record and playback, we can add service requests manually and write test cases.

Performance testing: We can use test results obtained by running webtest for performance testing. This can be done by analyzing metrics on the server and on the client, explore the HTTP traffic requests and performance counter data. Double click the test name in test result pane to see the details.


WebTest1.webtest | WebTest1 [9:13 AM]

Failed Click here to run again Internet Explorer 9.0 LAN Edit run settings

Request	Status	Total Time	Request Time	Request Bytes	Response Bytes
http://sites.google.com/favicon.ico	404 Not Found	0.527 sec	0.527 sec	0	11,910
http://rpt.cedexis.com/n1/0/1433820789099/0/0/0/0/1433820789337/143	200 OK	0.457 sec	0.457 sec	0	16
http://radar.cedexis.com/1/11918/radar/1433807157/7c8BnhLo7yXgLDQI	200 OK	0.578 sec	0.578 sec	0	2
https://googleads.g.doubleclick.net/pagead/gen_204	204 No Content	0.066 sec	0.066 sec	0	0

Web Browser | Request | Response | Context | Details

Rules

Rule	Type	Result	Parameters
Response URL	Validation	Passed	
Response Time Goal	Validation	Passed	Tolerance=0

Exception

Test Results

snaik2@BANSNAIK202 2015-06-(Run Debug Group By: [None] [All Columns]

Test run failed Results: 0/1 passed; Item(s) checked: 1

Result	Test Name	ID	Error Message
Failed	WebTest1	c:\bps\dev\sou	3 primary requests, 1 dependant requests and 0 conditional rules failed

Fig 3.4.2b: Test Result pane for webtest

3.4.2.1 Advantages

- Creating web tests without writing code using the Web Performance Test Recorder is good for beginners.
- We can edit recorded tests to tailor to specific needs.
- Simplify JavaScript interaction by automatically promoting dynamic parameters to editable web test parameters
- Load tests can be created by aggregation of existing web tests

3.4.2.2 Disadvantages

- Not much validation logic can be put into the tests: Validation can be put only on validation rules available in webtest.
- Maintenance is difficult: A small change in contract, would impact in changing all affected test cases
- Data Cleanup was not done properly
- High Volume tests were not at all executed in the test suite

3.4.3 Using Programming Language

Limitations of Microsoft webtest and dependency of third party tool like, SoapUI led us to write an automation framework on **Microsoft .Net** platform by using programming language **C#**.

Reasons for selecting C#:

- It is a simple, object-oriented language.
- Anyone with basic knowledge of C++ and C# language will be able to drive the framework.

Framework: When we talk about test framework, there is lot more things comes in picture. Before starting to develop framework we had to think about below aspects

- Suitable Programming Language
- Suitable Tool [If Applicable]
- Programming knowledge of team members
- Time required to develop a framework

Choosing a programing language is the key thing to start developing test framework. Because all the team members should be educated in that language or proper training should be given to educate them. If there is any tool available for automation it can be adopted. With all these boundaries C# can be used as programming language to start with framework. A framework should be an integrated system which integrates different libraries, test data sources, object details, reusable components and different utility modules. Once framework is developed integrate the framework with testing environment where we can run all the API tests.

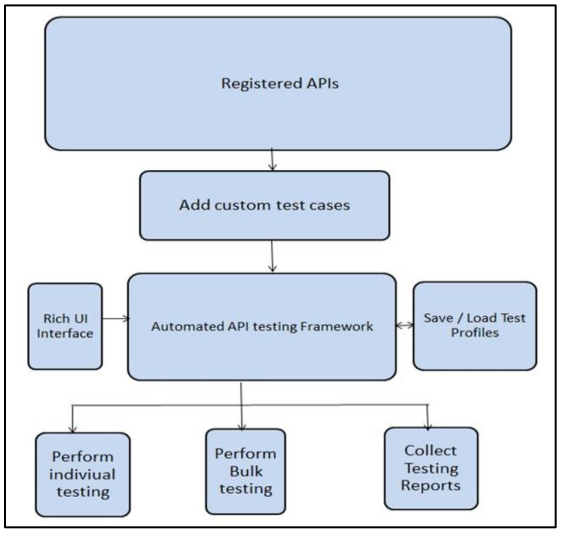


Fig 3.4.3a: C# Automation Framework

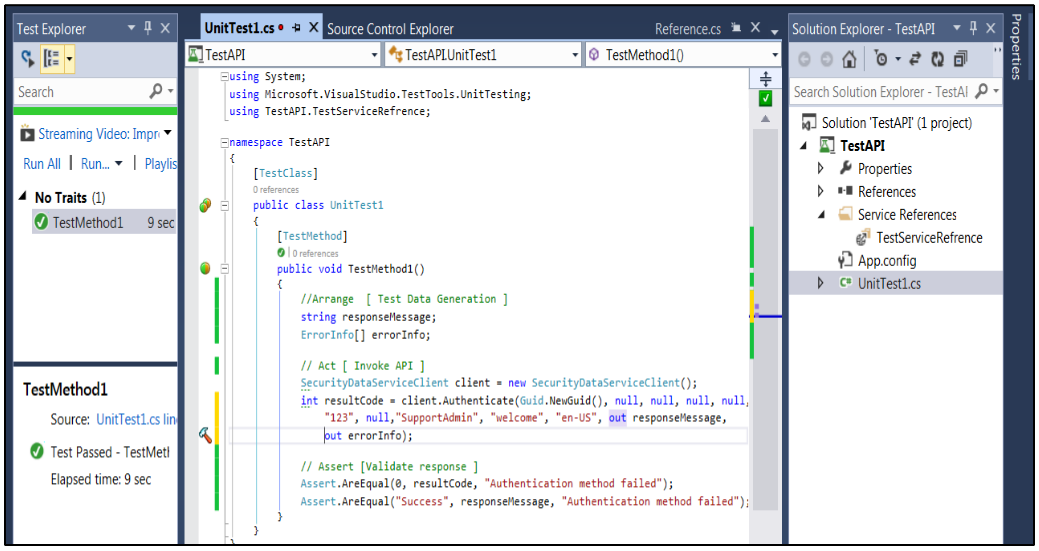
API testing is often compared with **unit testing** because the purpose served by both API testing and unit testing is same in most of the scenarios. Unit testing is covers some block of application code and the test cases can only verify its behavior with respect to boundary, valid input data and incorrect input values. Whereas API testing works for the same kind of testing but API testing can be taken to test end to end flow where we have to testing APIs which internally call one or more APIs to get the response. Unit testing has limited in scope whereas API testing has broader in scope as it can run for end to end testing.

API testing can be done by,

1. Create a new project of unit test in Microsoft visual studio
2. Adding Service Reference:
    - If service endpoint URL is available, then adding service reference is easy. Example of service endpoint: http://localhost/Mcafee.Saas.SecurityData/V1/SecurityDataService.svc?wsdl.
    - Once service is added to project, prepare required test data and consume that service client by providing test data as input.
3. Writing Test Method:

    In UnitTest.cs add test methods which validates required service.
    Create new test method and add service client which exposes all APIs available in the test method.

4. Invoke API and validate Response: Using Service Client invoke required API by providing the input data. Validate response of the API call with suitable result message and error codes as show in below diagram Fig 3.3.3b.



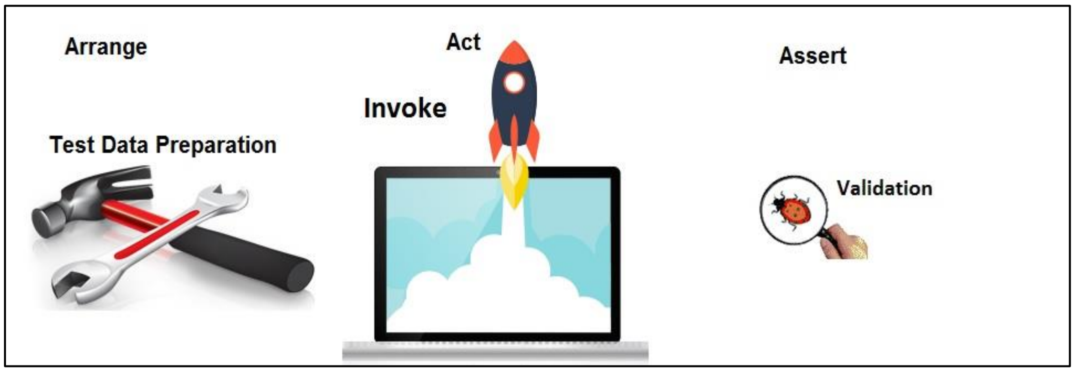
```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TestAPI.TestServiceRefrence;

namespace TestAPI
{
    [TestClass]
    0 references
    public class UnitTest1
    {
        [TestMethod]
        0 references
        public void TestMethod1()
        {
            //Arrange  [ Test Data Generation ]
            string responseMessage;
            ErrorInfo[] errorInfo;

            // Act [ Invoke API ]
            SecurityDataServiceClient client = new SecurityDataServiceClient();
            int resultCode = client.Authenticate(Guid.NewGuid(), null, null, null, null,
                "123", null,"SupportAdmin", "welcome", "en-US", out responseMessage,
                out errorInfo);

            // Assert [Validate response ]
            Assert.AreEqual(0, resultCode, "Authentication method failed");
            Assert.AreEqual("Success", responseMessage, "Authentication method failed");
        }
    }
```

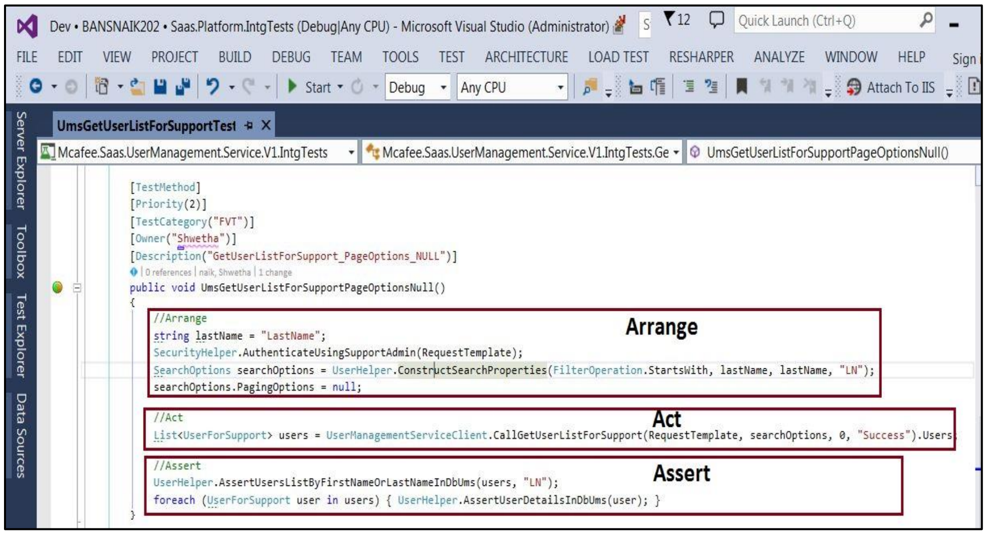
Fig 3.4.3b: Sample C# test case

Test Case: Any test case written for API testing will have below 3A's (as shown in Fig 3.4.3b below) as mandatory steps:

- **Arrange**: Arrange all requirements. Create test data required for API request.
- **Act**: Act on the API. With the test data created in step 1 invoke any API or invoke required APIs and get the response.
- **Assert**: Assert on the response obtained from the step 2 Validate the response from the API request with the local or remote data source.

In Simple words




Dev • BANSNAIK202 • Saas.Platform.IntgTests (Debug|Any CPU) - Microsoft Visual Studio (Administrator)
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE LOAD TEST RESHARPER ANALYZE WINDOW HELP
Start
Debug
Any CPU
Attach To IIS
Server Explorer
Toolbox
Test Explorer
Data Sources
UmsGetUserListForSupportTest
Mcafee.Saas.UserManagement.Service.V1.IntgTests
Mcafee.Saas.UserManagement.Service.V1.IntgTests.Ge
UmsGetUserListForSupportPageOptionsNull()
Arrange
Act
Assert

```
[TestMethod]
[Priority(2)]
[TestCategory("FVT")]
[Owner("Shwetha")]
[Description("GetUserListForSupport_PageOptions_NULL")]
0 references | naik, Shwetha | 1 change
public void UmsGetUserListForSupportPageOptionsNull()
{
    //Arrange
    string lastName = "LastName";
    SecurityHelper.AuthenticateUsingSupportAdmin(RequestTemplate);
    SearchOptions searchOptions = UserHelper.ConstructSearchProperties(FilterOperation.StartsWith, lastName, lastName, "LN");
    searchOptions.PagingOptions = null;

    //Act
    List<UserForSupport> users = UserManagementServiceClient.CallGetUserListForSupport(RequestTemplate, searchOptions, 0, "Success").Users

    //Assert
    UserHelper.AssertUsersListByFirstNameOrLastNameInDbUms(users, "LN");
    foreach (UserForSupport user in users) { UserHelper.AssertUserDetailsInDbUms(user); }
}
```

Fig 3.4.3c: **3A's mandatory steps**

Writing appropriate helpers, utility libraries, reusable methods one can design a good automation framework which can be used by different systems as platform of writing test cases of their own.

When we are dealing with the APIs more than one in one test case we have to make sure first API is not blocking the invocation of the second or rest APIs. When we call multiple APIs in one call or in multiple call it is good practice to have meaningful error messages at all steps so that tester can easily analyze error message or result message and can log **defect** according to that.

C# framework for End to end testing: API testing framework can be used for end to end testing before going to UI. To do this end to end testing user should be aware of sequence of APIs to be called and amount of time wait required between each API. For end to end 'Arrange' part should be kept as simple as much so that it should not lead to overburden for any person who joins later.

Test Result: Reporting test execution results is very important part of testing, whenever test execution cycle is complete, tester should make a complete test results report which includes the Test Pass/Fail status of the test cycle. If manual testing is done then the test pass/fail result should be captured in an excel sheet and if automation testing is done using automation tool then the HTML or XML reports should be provided to stakeholders as test deliverable.

By writing some scripts for running all the automated test cases at some cycle and generate proper test result. Test result of the run should include,

- Test cases pass / fail count
- Complete assert message for failed test cases
- Test Run time – This should include time for service call for each and every test case
- Metrics – metrics for test case pass fail statistics compared from last few runs
- % of test coverage and code coverage
- Major areas where failures are more.

3.4.3.1 Advantages

- C# framework is tool independent. There are n numbers of third party tools available for API testing automation but having a framework with C# will act independently without relying on any tools.
- Anyone with good basic knowledge of C# language will be able to manage framework.
- Programmers who are familiar with C++ concepts can easily learn C#.
- The Assembly concept of C# solves the versioning control problem of application
- Ease-to-development, the rich class library makes many functions easy to be implemented.
- Working with C# will give access to Microsoft .NET Framework libraries, which are extensive.
- As C# is .NET language it supports interoperability that means code written in C# can easily transformed to other languages of .NET [E.g., Visual Basic, C++ etc.]

3.4.3.2 Disadvantages

- C# mainly depends on .NET framework. Any library which is not found in .net will be difficult to implement
- C# programs are largely for Microsoft windows environment but C# doesn't come up with open source technologies and operating systems like Linux.

4. Result

Moving from manual testing to automation framework for API testing, increased lot of scope in testing. The journey from manual to automation and stabilization of code gave good results in managing the web services.

4.1 Opportunity for every QA to learn and enhance programming knowledge
4.2 With the stable framework an application can be more stabilized and can deliver good product to customers.
4.3 API testing also gives the deeper insight into the product which in turn helped QA to increase the product knowledge.
4.4 Analyzing the failed test cases helped in detecting defects at the earlier stage. Early detection of defects reduced rate of defect detection at the GUI level.
Percentage of early defect detection: 20%

4.5 Manual validation effort reduced because of the 100% automation of smoke test cases and 80 % automation of functional test cases.

Build Validation Tests: (Fig 4.1a)

- No of Test Cases: 390
- Frequency of test run: Every night with latest build
- Time: All 390 test cases take 3hours to complete and generate test result report. This saved 75% of manual effort and 12hours of resource time.

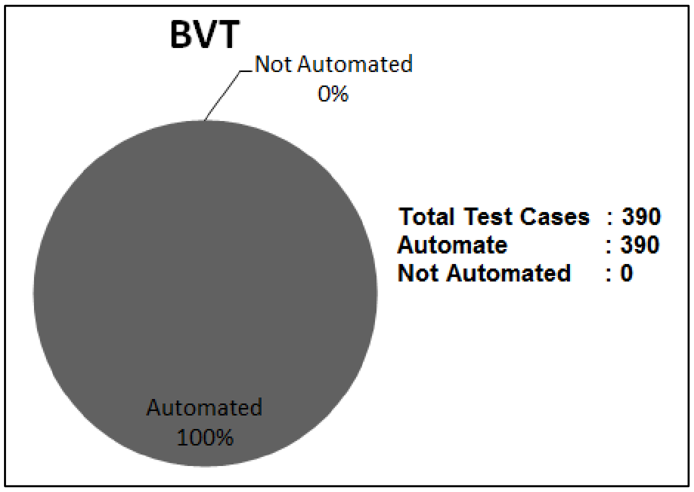


Fig 4.1a: 100% BVT Automation

Functional Validation Tests: (Fig 4.1b)

- No of Test Cases: 1550
- Frequency of test run: Weekly twice with latest build
- Time: All 850 test cases take 5 hours to complete and generate test result report. This saved 75% of manual effort.

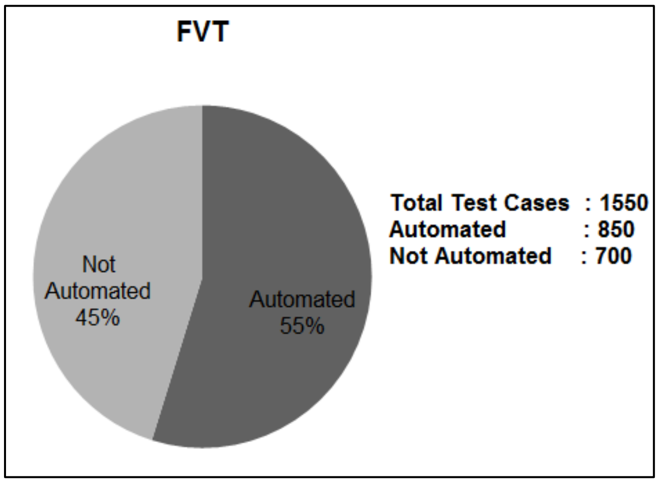


Fig 4.1b: 55% FVT Automation

Below graph depicts the overall automated test coverage across BVT, FVT and End to End Tests carried out till date. End to end testing for all BVT scenarios has started and is covered 10% as of now. Scope to cover more test cases in future.

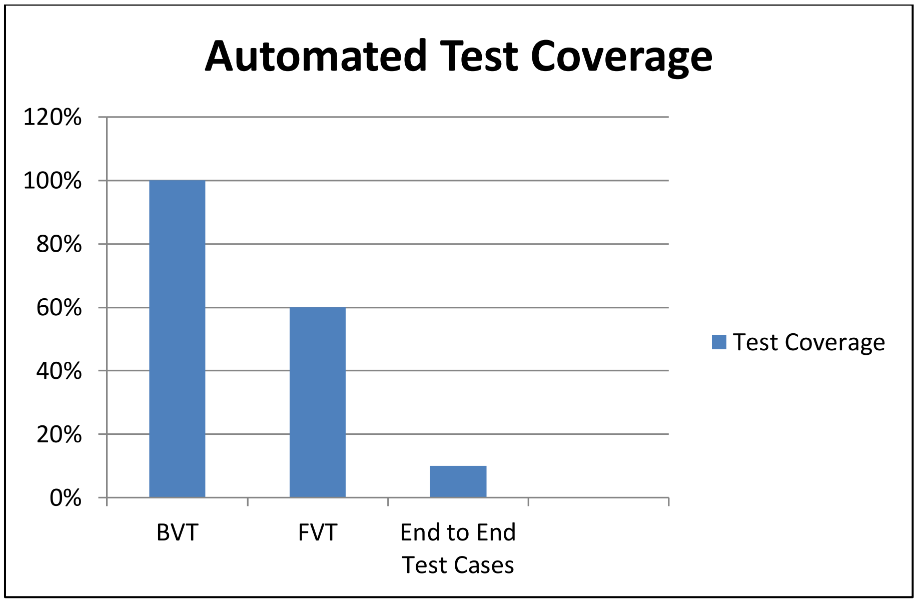


Fig 4.1c: Test Coverage

5. Summary

API consists of set of classes / functions / procedures which represent the business logic layer. If API is not tested properly, it may cause problems not only the API application but also in the calling application.

5.1 Advantages

- More effort into API testing leads to much healthier final product, ensuring the data access through the API only.
- API testing simplifies security and compliance testing and thereby certification, since there is only one interface.
- Ensuring that all the required business rules are being enforced at the API tier allows time for much more complete user-experience tests once the UI is released, and not having to concentrate on testing every single business rule and path through the application near the end of the project.
- Ensuring that the API offers complete functionality allows for easy future expansion of the application as new business needs arise.
- As API testing is structured way to testing, it makes automation more feasible.

5.2 Disadvantages

- No encryption - All requests and responses are visible to anyone between the requesting server and the API server.
- Increasing Tooling needs: API testing can be done only using different tools available. Cannot be tested without the tool like in GUI.
- Straightforward reason for failing API is not known, until and unless debugged into the code, unlike in GUI where it gives user-friendly errors.

Though automation is 100% possible, still there remains a question whether automating 100% test cases is really profitable? Because of few reasons like priority of test cases, frequency of those test cases run and effort calculated in automating them.

5.3 Future

With all the merits and de-merits kept in mind, we have strategy planned to automate End to End tests and Integration tests.

- **End to End Testing**: Plan to extend all BVT scenarios to cover end to end testing. This is one step ahead from normal API testing where we just test one functionality. End To End testing make hold of the scenario which user performs from the UI. Calling different APIs in single method is little exasperating because failing of some API will mark whole test case to fail but with good coding style and capturing exact result at all points we can develop a robust test suite.
- **Integration Testing**: Integration of different systems in testing is little difficult. We have a roadmap of covering different systems from the BVT perspective. Capturing behavior of each system for one single functionality and automation of this will lead us to have non-defective system.

References

Book:

James D. McCaffrey. 2012. .NET Test Automation Recipes: A Problem-Solution Approach. California. The Expert's Voice

Web Sites:

Marc van't Veer. 2013. "Testing the API behind a mobile app", https://www.polteq.com

Sulagna. 2009." Introduction-to-API-Testing", www.scribd.com on Jan 07, 2009," http://www.scribd.com/doc/9808382/Introduction-to-API-Testing#scribd

"Software Testing - API testing", http://www.tutorialspoint.com/software_testing_dictionary/api_testing.htm

Michael-churchman, 2014. "API Testing: Why It Matters, and How to Do It". https://blog.udemy.com on 2014. https://blog.udemy.com/api-testing/,

Joe Colantonio, 2013, "UFT – What is an API test type?" www.joecolantonio.com on February 19, 2013 http://www.joecolantonio.com/2013/02/19/uft-what-is-an-api-test-type/

http://sqa.stackexchange.com/questions/6745/how-do-you-test-a-backend-api

http://sqa.fyicenter.com/FAQ/Testing-Techniques/What_is_API_Testing_.html

http://www.encodedna.com/wcf/tutorial/advantages-of-wcf.htm

http://www.soapui.org/

http://www.codeproject.com/

http://en.wikipedia.org/

http://www.guru99.com/api-testing.html