

API TEST STRATEGY

Smart Check Further Lending

Contents

[**Summary**](#_Toc417567133) **3**

[**Introduction**](#_Toc417567135) **3**

[**Scope**](#_Toc417567134) **3**

[**In Scope**](#_Toc417567135) **3**

[**Out of Scope**](#_Toc417567136) **4**

[**Challenges**](#_Toc417567137) **4**

[**Resources**](#_Toc417567137) **5**

[**Roles and Responsibilities**](#_Toc417567138) **5**

[**Environments**](#_Toc417567137) **5**

[**Test Approach**](#_Toc417567137) **5**

[**Test Tools**](#_Toc417567145)

[**Entry Criteria**](#_Toc417567137) **5**

[**Exit Criteria**](#_Toc417567137) **5**

[**Denpendencies and Constraints**](#_Toc417567137) **5**

# **Summary**

The purpose of this document is to provide a concise high level Test Strategy to test the new API’s that is been redesigned.

**Introduction:**

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 centre stage and hence giving this layer the attention it truly deserves by shifting the onus from UI testing to Web Services based validation. Web services extend the World Wide Web infrastructure to facilitate a software to connect to other software applications. Web services combine the best aspects of component-based development and the Web. They are a cornerstone of the Microsoft .NET programming model.

API stands for Application Programming Interface, which specifies how one component should interact with the other. 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
* Analyse the output response

# **Scope**

## **In Scope**

The scope of this document is to define the strategy for the API testing only including:

* Be independent of environments used to API testing.
* Underneath services (e.g. Pan Credit, Experian, etc...) will be mocked and the test are written against the new web services that are been redesigned.

## 

## **Out of Scope**

The items out of scope are:

* Volume Testing
* Stress testing
* Load Testing
* 3rd Party services

# **Challenges:**

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.

* Main challenges in API testing are Parameter Combination, Parameter Selection, and Call Sequencing.
* Validating and Verifying the output in different system is not very easy for testers.
* Parameters selection should be known to the testers.
* Exception handling function needs to be tested.
* Coding knowledge is necessary for testers.

**Resources:**

Development, Execution and Maintenance of the API test suite will be picked up by the Manual and Automation Testers.

|  |  |
| --- | --- |
| Roles | Responsibility |
| Manual Tester | Creating, Executing and Maintaining the manual test cases. |
| Automation Tester | Designing, Executing and Maintaining the automated API tests. |

**Environments:**

The API testing will be carried out using the following integrated environment:

* World Pay
* Pan Credit
* Call Credit
* Offer Service
* Power Curve

**Test Approach:**

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 behaviour of your application / APIs.
* Since API tests bypass the user interface, they tend to be quicker and much more reliable than GUI tests.
* API automated testing often takes a fraction of the time as UI automated tests.

An efficient automation test strategy calls for automating tests at three different levels: unit, service and user interface which is explained by Test Pyramid.

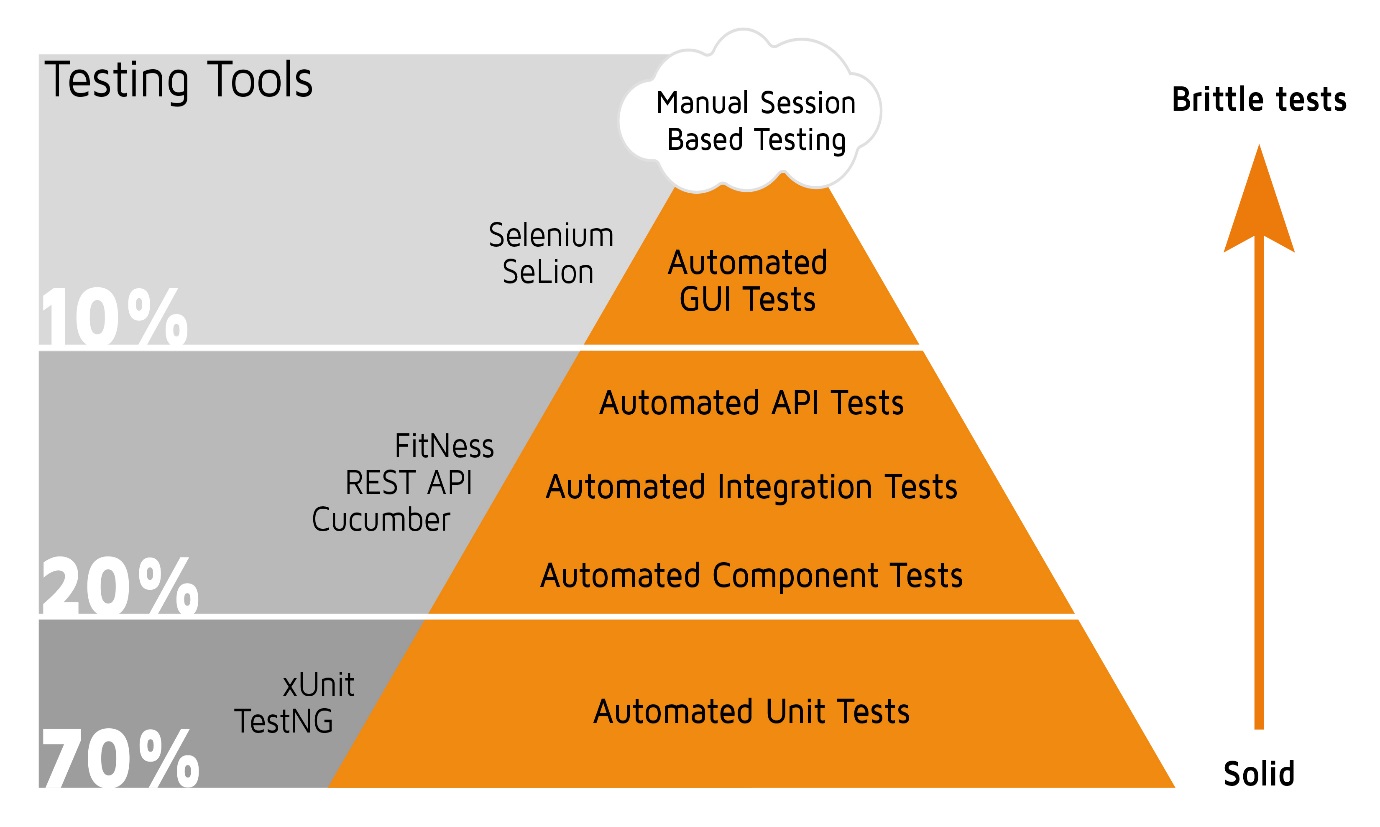
**Test Tools:**

There are number of open source tools available in the market one of which is **RestSharp**.

What is RestSharp? It is a comprehensive, open-source HTTP client library that works with all kinds of .Net technologies. It can be used to build robust applications by making it easy to interface with public APIs and quickly access data without the complexity of dealing with raw HTTP requests.

Some of these features include:

* Custom serialization and deserialization via ISerializer and IDeserializer.
* Both synchronous and asynchronous requests.
* Automatic XML and JSON parsing, including fuzzy element name matching (“product\_id” in XML/JSON will match C# property named ‘ProductId’).
* Multipart file/form uploads.
* OAuth, Basic, NTLM and Parameter-based Authentication Support for features such as GET, PUT, HEAD, POST, DELETE and OPTIONS.



### **Entry Criteria:**

Entry criteria are the required conditions and standards for work product quality that must be present or met prior to the start of a test phase.

Entrance criteria shall include following:

* Review of completed API test framework the prior test phase.
* Correct versioning of components moved into the appropriate test environment.
* Testing environment is configured and ready.

### **Exit Criteria:**

Exit criteria are the required conditions and standards for work product quality that block the promotion of incomplete or defective work products to the next test phase.

Exit criteria shall include the following:

* Successful execution of the test scripts(s) for the current test phase.
* No open critical, major, or average severity defects unless the issue is determined to be low impact and low risk.
* Component stability in the appropriate test environment.

**Dependencies and Constraints:**