Test Case Specification

For

JOHN KARU

May 10, 2019

**Table of content**

1. **Introduction …………………….**
2. **Test cases: MyConnect………….**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description** |
| 1 | 10/05/19 | John Karu | Initial Document |
| 2 | 10/05/19 | John Karu | Updated test case numbers to eliminate typographical errors |

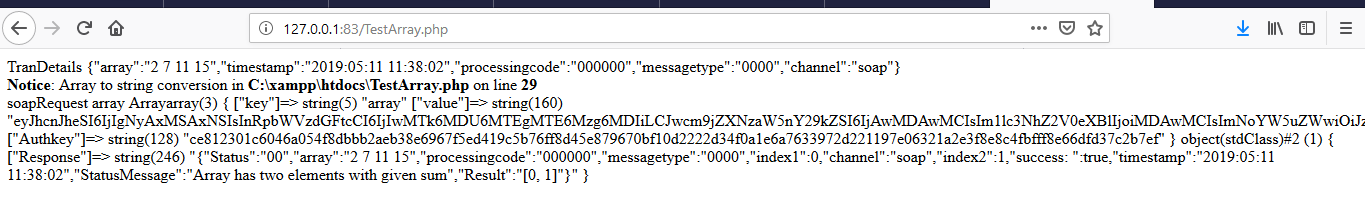
# Introduction

In the System Documentation I tackled all the tasks in question 1 , 2 , & 3 (i) using a single web service end point in the project : SafHackArrayChallenge which is my application under test called “MyConnect” just to harmonize (simply acting like a service bus )and hasten my work and avoid code duplications especially in accessing utilities like Datasource for Managing my DB connection among other Computational Logics as you will see below.so after getting the concept I hope the valuer will accept the approach due to time factor.

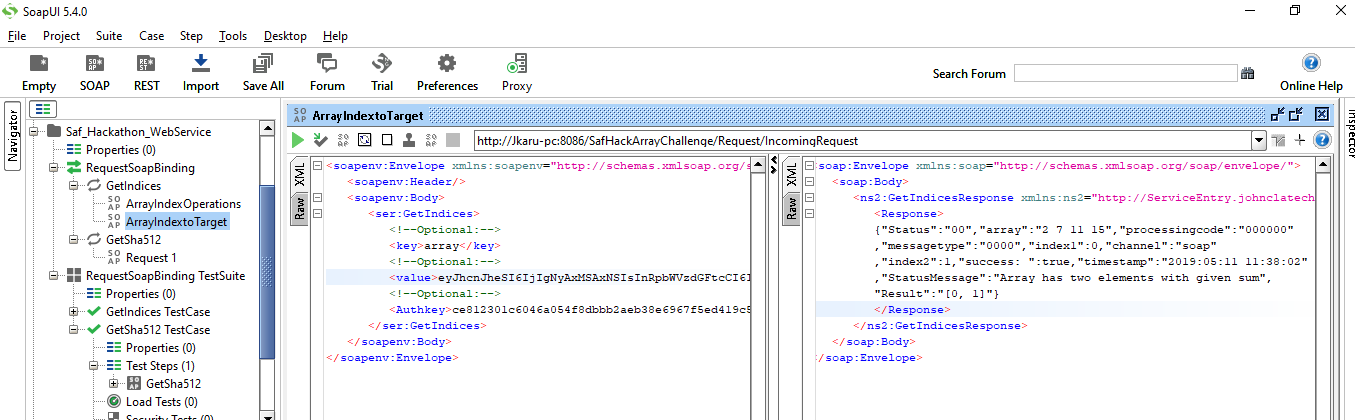
NB: The provided php files namely: TestArray.php, TestArrayOperators.php & TestAddCartoMngr.php. should act as the client request source hitting the web service end point. I have tailored these php files to generate requests in a uniform format of different requests that the application under test understands.so values input is hard coded at this point so that they are encoded before being send to the MyConnect Channel.

This document provides the test cases to be carried out for the **MyConnect** application. Each task to be tested is represented by an individual test case. Each case details the input and expected outputs.

Run the php files in a browser to generate the encoded and encrypted values and authkey respectively copy these values for separate requests you’re testing and paste to the SoapUI xml request parameters.



Above image shows a screen shot of requests and responses as displayed on the browser.



The result of the above shows same results displayed on the browser being displayed with the SoapUI.

# Test Cases:

# **APPLICATION: SAFHACKARRAYCHALLENGE: array indices for element summing to match a TARGET (Q1)**

# Authentication and Security

|  |  |
| --- | --- |
| Test ID | 1.1 |
| Title | Feeding correct Array request parameters to the Service |
| Feature | Testing request inputs security authentication and encoding of client requests to allow for a handshake. |
| Objective | Confirm that proper encoded values and correctly system generated salted authkey yields access to the web service as expected. |
| Setup | Soap UI simulator (hitting ws endpoint: http://127.0.0.1:8086/SafHackArrayChallenge/Request/IncomingRequest?wsdl)(port: 8086 is my application servers port where the service is deployed);  the php API file hosted in the htdocs file of your Apache server(pointing to the same wsdl ) is able to send the request values and generate and prepare the request to MyConnect application respectively. |
| Test Data | Request PARAM’s:  key = “array”  value = “base64 encoded json request” e.g. use: -> “eyJhcnJheSI6IjIxIDEzIiwidGltZXN0YW1wIjoiMjAxOTowNToxMSAxMTowNDozNCIsInByb2Nlc3Npbmdjb2RlIjoiMDAwMjAwIiwibWVzc2FnZXR5cGUiOiIwMDAwIiwib3BlcmF0aW9uIjoic3VidHJhY3QiLCJjaGFubmVsIjoic29hcCJ9”  Authkey = “sha512 encrypted base64 encoded combination of specific PARAM’s” e.g. use : “9f316907e94d50ec9cb470fbb73618bf630094b3e528f5d9997f77f66553e1ddf9b49e59bd47fee1f0b665047c69b2ab65ea3eb40460b8f83e25f704bbe62c3a” |
| Test Actions | 1. for SoapUI at the url endpoint section pull :http://<host>:8086/SafHackArrayChallenge/Request/IncomingRequest?wsdl it pulls the XML request where you put the respective PARAM’s. Click the play button  2. for the php file hosted in your apache, named e.g. Test1.php  Run on the browser: <http://localhost/Test1.php> and press Enter  In this case run : <http://localhost/TestArray.php> |
| Expected Results | The service will authenticate and will console a **status: 00**. if a successful handshake has happened. |

|  |  |
| --- | --- |
| Test ID | 1.2 |
| Title | Feeding correct Array request parameters to the Service |
| Feature | Testing request inputs security authentication and encoding of client requests to allow for a handshake. |
| Objective | Confirm that proper encoded values and incorrect system generated salted authkey denies access to the web service as expected. |
| Setup | Perform actions as the preceding test case. |
| Test Data | Correct user value, incorrect authkey (delete some characters of the authkey)  Request PARAM’s:  key = “array”  value = “base64 encoded json request” e.g. (eyJhcnJheSI6IjIxIDEzIiwidGltZXN0YW1wIjoiMjAxOTowNToxMSAxMTowNDozNCIsInByb2Nlc3Npbmdjb2RlIjoiMDAwMjAwIiwibWVzc2FnZXR5cGUiOiIwMDAwIiwib3BlcmF0aW9uIjoic3VidHJhY3QiLCJjaGFubmVsIjoic29hcCJ9)  Authkey = “sha512 encrypted base64 encoded combination of specific PARAM’s” e.g. (4d50ec9cb470fbb73618bf630094b3e528f5d9997f77f66553e1ddf9b49e59bd47fee1f0b665047c69b2ab65ea3eb40460b8f83e25f704bbe62c3a)  Removed first 10 characters. |
| Test Actions | 1 Perform actions as the preceding test case.  2. Enter invalid authkey values. |
| Expected Results | System returns a fail message.  The service will not authenticate and will console a **status: 01** as the handshake never happened.  If you alter any of the values of the PARAM’s being passed an error message will be returned.  Errors return with **status : 02** |

**Purpose achievements**

|  |  |
| --- | --- |
| Test ID | 1.3 |
| Title | Delivering the right array in form of “x y z q” Indices as implemented in the application under test |
| Feature | Providing different sets of arrays to confirm if the program will be able to pick the right positions of the indices containing elements summing up to an anonymous target. |
| Objective | Confirm the application under test can handle various array sets and deliver the expected results. |
| Setup | This also applies to the above test cases. |
| Test Data | Correct request parameters but set varying arrays from the API that performs the encoding and other data preparations.  Key = “array”  value = “9f316907e94d50ec9cb470fbb73618bf630094b3e528f5d9997f77f66553e1ddf9b49e59bd47fee1f0b665047c69b2ab65ea3eb40460b8f83e25f704bbe62c3a”  authkey = “eyJhcnJheSI6IjIxIDEzIiwidGltZXN0YW1wIjoiMjAxOTowNToxMSAxMTowNDozNCIsInByb2Nlc3Npbmdjb2RlIjoiMDAwMjAwIiwibWVzc2FnZXR5cGUiOiIwMDAwIiwib3BlcmF0aW9uIjoic3VidHJhY3QiLCJjaGFubmVsIjoic29hcCJ9”  Hit the play button in the SoupUI.  In this case run : <http://localhost/TestArray.php> |
| Test Actions | 1 Perform actions as the preceding test case.  2. Enter valid array values. |
| Expected Results | The service should get the proper indices and console a **status: 00**. a successful process has happened. |

|  |  |
| --- | --- |
| Test ID | 1.4 |
| Title | Delivering a wrong array in form of “x, y, z, q” Indices as not implemented in the application under test |
| Feature | different sets of arrays to confirm if the program will be able to pick the right arrays and return positions of the indices containing elements summing up to an anonymous target. |
| Objective | Confirm that positions of matching elements summation to a target is not returned if the array is incorrect. |
| Setup | SoupUI tool or php API has MyConnect application ready to run. |
| Test Data | Correct request parameters but set varying arrays from the API that performs the encoding and other data preparations.  Key = “array”  value = “eyJhcnJheSI6IjIsIDcsIDExLCAxNSIsInRpbWVzdGFtcCI6IjIwMTk6MDU6MTEgMTA6NTY6MDMiLCJwcm9jZXNzaW5nY29kZSI6IjAwMDAwMCIsIm1lc3NhZ2V0eXBlIjoiMDAwMCIsImNoYW5uZWwiOiJzb2FwIn0=”  authkey = “710c299787b84ccdb0fed7da9b375114dd73b52a6d6efd9d225acc4258bfb88a3b3f406e0c8839efe897237435954eb481b5f6fe68e9e2e75a4c83aed14eed06”  Hit the play button in the SoupUI.  In this case run : <http://localhost/TestArray.php> |
| Test Actions | 1 Perform actions as the preceding test case.  2. Enter an invalid string array on the php side values. |
| Expected Results | The system should be unable to consume and manipulate the provided array therefore returns a fail status code and a status message  System returns a fail message also.  The service will return a **status: 01**.  If you alter any of the values of the PARAM’s being passed an error message will be returned.  Errors return with **status : 02** |

1. **Application: WSConsumeCalculator: array elements exposed to different Operations to return the results and save to database (Q3(i)**

**NB: SafHackArrayChallenge has the similar implementations like in the test below, only that its saves the results to the database whereas WSConsumeCalculator is for testing and display results to the console. So During db. test we will test from class SafHackArrayChallenge and the PHP file will be: TestArrayOperators.php.**

The application under test consumes a soap endpoint. : <http://www.dneonline.com/calculator.asmx?WSDL>

|  |  |
| --- | --- |
| Test ID | 2.5 |
| Title | Correct Operation |
| Feature | Enumerating the different operations exposed by the soap end point. add, divide, subtract, multiply.  Input the operations one at a time. |
| Objective | Confirm that the operations fed to the application under test are present in the soap endpoint. |
| Setup | Must have internet connection to your computer.  In your IDE i.e. Eclipse, InteliJ, NetBeans, etc. open project  WSConsumeCalculator and just run:  In these case am using NetBeans 8.2 IDE  It uses Scanner Library on the CONSOLE to input the operations |
| Test Data | Enter the Operation:  multiply |
| Test Actions | 1. Hit run application (main class thread will be started/ executed)  2. Console prompts you to enter operation  3. Enter one of the Operations handled: (add, divide, subtract, multiply) |
| Expected Results | System should console that the operation accepted after enumeration of the operations finds a match.  No error has occurred. |

|  |  |
| --- | --- |
| Test ID | 3.1 |
| Title | Incorrect operation |
| Feature | Enumerating the different operations exposed by the soap end point. add, divide, subtract, multiply.  Input a Wrong operations i.e. say modulus or some random characters. |
| Objective | Confirm that incorrect operation does not yields access to the web service as expected. |
| Setup | This also applies to the above test case 2.5. |
| Test Data | Enter the Operation:  modulus |
| Test Actions | This also applies to the above test case 2.5. |
| Expected Results | System should console that the operation rejected after enumeration of the operations doesn’t find a match.  an error has occurred on the enumeration. |

|  |  |
| --- | --- |
| Test ID | 3.2 |
| Title | Correct results |
| Feature | Enumerating the different operations exposed by the soap end point. add, divide, subtract, multiply.  Input an operations i.e. say add or any of (add, divide, subtract, multiply). |
| Objective | Confirm operation entered access to the web service and consoles a result. |
| Setup | Access to internet:  This also applies to the above test case 2.5. |
| Test Data | This also applies to the above test case 2.5. |
| Test Actions | This also applies to the above test case 2.5. |
| Expected Results | A result for the operation entered will be consoled. |

Below test requires an access to the database therefore I implemented these task as mentioned in the NB above.

|  |  |
| --- | --- |
| Test ID | 3.3 |
| Title | Saves in the database. |
| Feature | Saving the result of the operation in a database and confirming it’s is there. |
| Objective | Confirm that after putting the array to the system and specifying the operation the application under test is able to save this data to the database. |
| Setup | Access to the internet.  Ensure you have Microsoft Sql server manager to view your db.  Soap UI simulator (hitting ws endpoint: http://127.0.0.1:8086/SafHackArrayChallenge/Request/IncomingRequest?wsdl)(port: 8086 is my application servers port where the service is deployed);  the php API file hosted in the htdocs file of your Apache server (pointing to the same wsdl) is able to generate, prepare and send the request values to MyConnect application.  In this case php file to edit is : [TestArrayOperators.php](http://localhost/TestArrayOperators.php%20) |
| Test Data | Correct request parameters but set varying arrays from the API that performs the encoding and other data preparations.  Key = “array”  value = “9f316907e94d50ec9cb470fbb73618bf630094b3e528f5d9997f77f66553e1ddf9b49e59bd47fee1f0b665047c69b2ab65ea3eb40460b8f83e25f704bbe62c3a”  authkey = “eyJhcnJheSI6IjIxIDEzIiwidGltZXN0YW1wIjoiMjAxOTowNToxMSAxMTowNDozNCIsInByb2Nlc3Npbmdjb2RlIjoiMDAwMjAwIiwibWVzc2FnZXR5cGUiOiIwMDAwIiwib3BlcmF0aW9uIjoic3VidHJhY3QiLCJjaGFubmVsIjoic29hcCJ9”  Hit the play button in the SoupUI.  In this case run : [http://localhost/TestArrayOperators.php](http://localhost/TestArrayOperators.php%20) in your browser. |
| Test Actions | 1. for SoapUI at the url endpoint section pull :http://<host>:8086/SafHackArrayChallenge/Request/IncomingRequest?wsdl it pulls the XML request where you put the respective PARAM’s. Click the play button  2. for the php file hosted in your apache, named e.g. Test1.php  Run on the browser: <http://localhost/Test1.php> and press Enter  In this case run : [http://localhost/TestArrayOperators.php](http://localhost/TestArrayOperators.php%20) |
| Expected Results | The results of every operation gets saved into a database table  My table is: [HackathonDB]. [dbo]. [Arrays\_Operations] check the response on your browser or the SoapUI results pane to confirm.  Go to your Mssql manager studio [HackathonDB].[dbo].[Arrays\_Operations] run:  SELECT \* FROM [HackathonDB].[dbo].[Arrays\_Operations]  Database affected on the Operations of the arrays values |

1. **Application: SafHackArrayChallenge: Show room Car Manager API (Q2)**

All of the problems above follow the same process.

I have provided the following php files for testing in your browser and SoapUI:

* TestAddVehicleMakeToMngr.php – adds a vehicle make in the system.
* TestAddVehicleTypeToMngr.php – adds a vehicle model / type in the system.
* TestAddVehicleColorToMngr.php - adds a vehicle color in the system.
* TestRegVehicleToMngr.php – Registers a vehicle with its full details to the system.
* TestSearchAllVehicleMngr.php – Searches and Get all vehicles registered in the system.
* TestSearchVehicleAttrMngr.php - Searches and Get all vehicles using specific attributes (make, color, availability).
* TestUpdateVehicleDetsMngr.php – Updates Vehicle details using specific attributes. (Color and Availability).
* TestDeleteCarFromMngr.php – This deletes a vehicle from show room but retains the details of the vehicle using a flag deleted as “1”. Hence not actually removing the record.