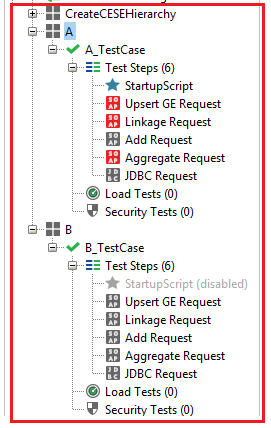
**API Automation using SoapUI**

**Pre-requisites:**

1. SoapUI Tool 5.2.1 or plus
2. Web Services List
3. Data Sheet

**Steps**

1. Download and install SoapUI 5.2.1 tool
2. Create a SOAP project with the valid WSDL URL. Look at this screenshot



1. Inside the Startup Script groovy step, written the code like this.

import jxl.\*

import java.io.File

import java.util.Date

import java.lang.String

import java.util.Calendar

import java.io.InputStream

import java.util.LinkedHashMap

import java.io.FileInputStream

import java.io.FileOutputStream

import java.text.SimpleDateFormat

import java.util.concurrent.TimeUnit

import org.apache.poi.ss.usermodel.\*

import org.apache.poi.hssf.util.HSSFColor

import org.apache.poi.hssf.usermodel.HSSFRow

import org.apache.poi.hssf.usermodel.HSSFFont

import org.apache.poi.hssf.usermodel.HSSFCell

import org.apache.poi.hssf.usermodel.HSSFSheet

import org.apache.poi.ss.usermodel.IndexedColors

import org.apache.poi.hssf.usermodel.HSSFWorkbook

import org.apache.poi.hssf.usermodel.HSSFCellStyle

import org.apache.poi.ss.usermodel.FillPatternType

def filePath = "D:\\working\\POC\\RunManager.xls"

String startTime = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.getInstance().getTime());

log.info ("Execution Started On :" + startTime);

Workbook book = Workbook.getWorkbook(new File(filePath))

Sheet sheet = book.getSheet(0) // Opens "BaseSheet"

int totalRows = sheet.getRows()

int totalCols = sheet.getColumns()

// Read "BaseSheet" rows which has "Y" flag enabled

log.info "Reading BaseSheet....."

for(int baseRow = 1; baseRow < totalRows; baseRow++)

{

if (sheet.getCell(1,baseRow).getContents().equalsIgnoreCase("Y"))

{

def colsList = []

// Fetch all the column data of the current row from the BaseSheet

for (int baseCol=0; baseCol < totalCols; baseCol++)

{

colsList.add(sheet.getCell(baseCol,baseRow).getContents())

}

// Store the base column values to the properties

testRunner.testCase.testSuite.project.setPropertyValue("SUITE\_NAME",colsList[0])

testRunner.testCase.testSuite.project.setPropertyValue("SUITE\_RESULT",colsList[2])

testRunner.testCase.testSuite.project.setPropertyValue("TIME\_TAKEN",colsList[3])

String sheetName = testRunner.testCase.testSuite.project.getPropertyValue("SUITE\_NAME")

if (sheetName.equals("A"))

{

String suiteStartTime = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.getInstance().getTime());

log.info "A testsuite executing..." + "\n"

// Navigate to the respective TestSuite -> TestCase sheet

Sheet tcSheet = book.getSheet(sheetName)

int rows = tcSheet.getRows()

int cols = tcSheet.getColumns()

// This loop iterates fetch the rows which testcases are marked as "Y"

for (int row = 1; row < rows; row++)

{

if (tcSheet.getCell(1,row).getContents().equalsIgnoreCase("Y"))

{

def columnList = []

def resultList = []

for (int col = 0; col < cols; col++)

{

columnList.add(tcSheet.getCell(col,row).getContents())

}

context.testCase.setPropertyValue("TC\_ID", columnList[0])

context.testCase.setPropertyValue("FLAG", columnList[1])

context.testCase.setPropertyValue("RESULT", columnList[2])

context.testCase.setPropertyValue("STEP\_NAME\_1", columnList[3])

context.testCase.setPropertyValue("COMPANY\_NAME", columnList[4])

context.testCase.setPropertyValue("ADDRESS\_LINE\_1", columnList[5])

context.testCase.setPropertyValue("CITY\_NAME", columnList[6])

context.testCase.setPropertyValue("STATE\_CODE", columnList[7])

context.testCase.setPropertyValue("POSTAL\_CODE", columnList[8])

context.testCase.setPropertyValue("COUNTRY\_NAME", columnList[9])

context.testCase.setPropertyValue("COUNTRY\_CODE", columnList[10])

context.testCase.setPropertyValue("AREA\_CODE", columnList[11])

context.testCase.setPropertyValue("TEL\_NUMBER", columnList[12])

context.testCase.setPropertyValue("AGG\_CODE", columnList[13])

context.testCase.setPropertyValue("STEP\_NAME\_2", columnList[14])

context.testCase.setPropertyValue("GO\_ID", columnList[15])

context.testCase.setPropertyValue("STEP\_NAME\_3", columnList[16])

context.testCase.setPropertyValue("GEO\_SUBDIVISION", columnList[17])

context.testCase.setPropertyValue("STEP\_NAME\_4", columnList[18])

context.testCase.setPropertyValue("STEP\_NAME\_5", columnList[19])

// Pass this value to XML and Start run the test step - PutRequest

def request\_1 = context.testCase.getPropertyValue("STEP\_NAME\_1")

def upsertGeRequest = context.testCase.getTestStepByName(request\_1)

upsertGeRequest.run(testRunner,context)

// Add execution status in the list

resultList.add(context.testCase.getPropertyValue("RESPONSE\_STATUS"))

// Pass this value to XML and Start run the test step - GetRequest

def request\_2 = context.testCase.getPropertyValue("STEP\_NAME\_2")

def linkageRequest = context.testCase.getTestStepByName(request\_2)

linkageRequest.run(testRunner,context)

// Add execution status in the list

resultList.add(context.testCase.getPropertyValue("RESPONSE\_STATUS"))

// Pass this value to XML and Start run the test step - JdbcRequest

def request\_3 = context.testCase.getPropertyValue("STEP\_NAME\_3")

def addRequest = context.testCase.getTestStepByName(request\_3)

addRequest.run(testRunner,context)

// Add execution status in the list

resultList.add(context.testCase.getPropertyValue("RESPONSE\_STATUS"))

// Pass this value to XML and Start run the test step - JdbcRequest

def request\_4 = context.testCase.getPropertyValue("STEP\_NAME\_4")

def aggRequest = context.testCase.getTestStepByName(request\_4)

aggRequest.run(testRunner,context)

resultList.add(context.testCase.getPropertyValue("RESPONSE\_STATUS"))

// Pass this value to XML and Start run the test step - JdbcRequest

def request\_5 = context.testCase.getPropertyValue("STEP\_NAME\_5")

def jdbcRequest = context.testCase.getTestStepByName(request\_5)

jdbcRequest.run(testRunner,context)

resultList.add(context.testCase.getPropertyValue("RESPONSE\_STATUS"))

///////////////////////////////////////////////////////////////////////////////////

String tSuiteName = sheetName

String tcId = context.testCase.getPropertyValue("TC\_ID")

// Get consolidated result of all Requests and store it to the LinkedHashMap

LinkedHashMap<String,String> suitesInfo = new LinkedHashMap<String,String>();

suitesInfo.put("SUITE\_NAME",tSuiteName)

suitesInfo.put("TC\_ID",tcId)

suitesInfo.put(request\_1,context.testCase.getPropertyValue(request\_1 + "\_RESPONSE\_STATUS"))

suitesInfo.put(request\_2,context.testCase.getPropertyValue(request\_2 + "\_RESPONSE\_STATUS"))

suitesInfo.put(request\_3,context.testCase.getPropertyValue(request\_3 + "\_RESPONSE\_STATUS"))

suitesInfo.put(request\_4,context.testCase.getPropertyValue(request\_4 + "\_RESPONSE\_STATUS"))

suitesInfo.put(request\_5,context.testCase.getPropertyValue(request\_5 + "\_RESPONSE\_STATUS"))

// Pass the tcID, tSuiteName and Map to the method writeTestSuiteStatusInExcel()

writeExcel(tSuiteName,tcId,suitesInfo)

///////////////////////////////////////////////////////////////////////////////////

// Update the result in the excel sheet against the TestCase ID

FileInputStream fis = new FileInputStream(new File(filePath))

HSSFWorkbook wBook = new HSSFWorkbook(fis)

HSSFSheet hSheet = wBook.getSheet(sheetName)

org.apache.poi.ss.usermodel.Cell resultCell = null;

// Check if 'resultList' list all PASS or not

def resCount = 0

for (int x=0; x < resultList.size(); x++)

{

if (resultList[x].equalsIgnoreCase("PASS"))

{

resCount++;

}

}

// If 'resultList.size' and 'resCount' are equal,

// update testcase status as PASS

// Else update it as FAIL

if (resultList.size() == resCount)

{

// Set the cell values Request and Response

resultCell = hSheet.getRow(row).getCell(2)

resultCell.setCellValue("PASS")

// Highlight the status cell - Background

HSSFCellStyle style = wBook.createCellStyle()

style.setFillForegroundColor(IndexedColors.GREEN.getIndex())

style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND)

// Foreground

HSSFFont font = wBook.createFont()

font.setColor(HSSFColor.WHITE.index)

style.setFont(font)

// Apply border for the cells

style.setBorderBottom(HSSFCellStyle.BORDER\_THIN)

style.setBorderTop(HSSFCellStyle.BORDER\_THIN)

style.setBorderRight(HSSFCellStyle.BORDER\_THIN)

style.setBorderLeft(HSSFCellStyle.BORDER\_THIN)

resultCell.setCellStyle(style)

// Assign the corresponding TestSuite status as 'FAIL'

testRunner.testCase.testSuite.project.setPropertyValue("SUITE\_RESULT", "PASS")

}

else

{

// Set the cell values Request and Response

resultCell = hSheet.getRow(row).getCell(2)

resultCell.setCellValue("FAIL")

// Highlight the status cell - Background

HSSFCellStyle style = wBook.createCellStyle()

style.setFillForegroundColor(IndexedColors.RED.getIndex())

style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND)

// Foreground

HSSFFont font = wBook.createFont()

font.setColor(HSSFColor.WHITE.index)

style.setFont(font)

// Apply border for the cells

style.setBorderBottom(HSSFCellStyle.BORDER\_THIN)

style.setBorderTop(HSSFCellStyle.BORDER\_THIN)

style.setBorderRight(HSSFCellStyle.BORDER\_THIN)

style.setBorderLeft(HSSFCellStyle.BORDER\_THIN)

resultCell.setCellStyle(style)

// Assign the corresponding TestSuite status as 'FAIL'

testRunner.testCase.testSuite.project.setPropertyValue("SUITE\_RESULT", "FAIL")

}

// Update the excel file with the Testcase Status (Pass/Fail)

FileOutputStream fout = new FileOutputStream(new File(filePath))

wBook.write(fout)

fout.close()

fis.close()

}

}

String strSuiteResult = testRunner.testCase.testSuite.project.getPropertyValue("SUITE\_RESULT")

// Open the TestSuite status in the BaseSheet

FileInputStream fis1 = new FileInputStream(new File(filePath))

HSSFWorkbook pWbook = new HSSFWorkbook(fis1)

HSSFSheet pSheet = pWbook.getSheetAt(0) // BaseSheet

org.apache.poi.ss.usermodel.Cell suiteResultCell = null;

org.apache.poi.ss.usermodel.Cell suiteTimeCell = null;

// Set the status of TestSuite in the appropriate cell

suiteResultCell = pSheet.getRow(baseRow).getCell(2)

suiteResultCell.setCellValue(strSuiteResult)

if (strSuiteResult.equalsIgnoreCase("PASS"))

{

// Highlight the status cell - Background

HSSFCellStyle style = pWbook.createCellStyle();

style.setFillForegroundColor(IndexedColors.GREEN.getIndex())

style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND)

// Foreground

HSSFFont font = pWbook.createFont();

font.setColor(HSSFColor.WHITE.index);

style.setFont(font);

// Apply border for the cells

style.setBorderBottom(HSSFCellStyle.BORDER\_THIN)

style.setBorderTop(HSSFCellStyle.BORDER\_THIN)

style.setBorderRight(HSSFCellStyle.BORDER\_THIN)

style.setBorderLeft(HSSFCellStyle.BORDER\_THIN)

suiteResultCell.setCellStyle(style);

}

else

{

// Highlight the status cell - Background

HSSFCellStyle style = pWbook.createCellStyle();

style.setFillForegroundColor(IndexedColors.RED.getIndex())

style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND)

// Foreground

HSSFFont font = pWbook.createFont();

font.setColor(HSSFColor.WHITE.index);

style.setFont(font);

// Apply border for the cells

style.setBorderBottom(HSSFCellStyle.BORDER\_THIN)

style.setBorderTop(HSSFCellStyle.BORDER\_THIN)

style.setBorderRight(HSSFCellStyle.BORDER\_THIN)

style.setBorderLeft(HSSFCellStyle.BORDER\_THIN)

suiteResultCell.setCellStyle(style);

}

// Find total time taken for the test suite

String endTime = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.getInstance().getTime())

// Invoke TimeDifference method

String strSuiteTime = getDiffTime(suiteStartTime,endTime)

// Set the Total Time Taken of TestSuite in the appropriate cell

suiteTimeCell = pSheet.getRow(baseRow).getCell(3)

suiteTimeCell.setCellValue(strSuiteTime)

// Update the excel file

FileOutputStream fout1 = new FileOutputStream(new File(filePath))

pWbook.write(fout1)

fout1.close()

fis1.close()

}

else if (sheetName.equals("B"))

{

String suiteStartTime = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.getInstance().getTime());

log.info "B testsuite executing..." + "\n"

// Navigate to the respective TestSuite -> TestCase sheet

Sheet tcSheet = book.getSheet(sheetName)

int rows = tcSheet.getRows()

int cols = tcSheet.getColumns()

// This loop iterates till the "A" suite sheet rows

for (int row = 1; row < rows; row++)

{

if (tcSheet.getCell(1,row).getContents().equalsIgnoreCase("Y"))

{

def columnList = []

def resultList = []

for (int col = 0; col < cols; col++)

{

columnList.add(tcSheet.getCell(col,row).getContents())

}

context.testCase.setPropertyValue("TC\_ID", columnList[0])

context.testCase.setPropertyValue("FLAG", columnList[1])

context.testCase.setPropertyValue("RESULT", columnList[2])

context.testCase.setPropertyValue("STEP\_NAME\_1", columnList[3])

context.testCase.setPropertyValue("COMPANY\_NAME", columnList[4])

context.testCase.setPropertyValue("ADDRESS\_LINE\_1", columnList[5])

context.testCase.setPropertyValue("CITY\_NAME", columnList[6])

context.testCase.setPropertyValue("STATE\_CODE", columnList[7])

context.testCase.setPropertyValue("POSTAL\_CODE", columnList[8])

context.testCase.setPropertyValue("COUNTRY\_NAME", columnList[9])

context.testCase.setPropertyValue("COUNTRY\_CODE", columnList[10])

context.testCase.setPropertyValue("AREA\_CODE", columnList[11])

context.testCase.setPropertyValue("TEL\_NUMBER", columnList[12])

context.testCase.setPropertyValue("AGG\_CODE", columnList[13])

context.testCase.setPropertyValue("STEP\_NAME\_2", columnList[14])

context.testCase.setPropertyValue("GO\_ID", columnList[15])

context.testCase.setPropertyValue("STEP\_NAME\_3", columnList[16])

context.testCase.setPropertyValue("GEO\_SUBDIVISION", columnList[17])

context.testCase.setPropertyValue("STEP\_NAME\_4", columnList[18])

context.testCase.setPropertyValue("STEP\_NAME\_5", columnList[19])

def project = testRunner.testCase.testSuite.project

// Pass this value to XML and Start run the test step - PutRequest

def request\_1 = context.testCase.getPropertyValue("STEP\_NAME\_1")

testRunner.runTestStep(project.testSuites[sheetName].testCases["B\_TestCase"].testSteps[request\_1])

resultList.add(context.testCase.getPropertyValue("RESPONSE\_STATUS"))

// Pass this value to XML and Start run the test step - GetRequest

def request\_2 = context.testCase.getPropertyValue("STEP\_NAME\_2")

testRunner.runTestStep(project.testSuites[sheetName].testCases["B\_TestCase"].testSteps[request\_2])

resultList.add(context.testCase.getPropertyValue("RESPONSE\_STATUS"))

// Pass this value to XML and Start run the test step - JdbcRequest

def request\_3 = context.testCase.getPropertyValue("STEP\_NAME\_3")

testRunner.runTestStep(project.testSuites[sheetName].testCases["B\_TestCase"].testSteps[request\_3])

resultList.add(context.testCase.getPropertyValue("RESPONSE\_STATUS"))

// Pass this value to XML and Start run the test step - JdbcRequest

def request\_4 = context.testCase.getPropertyValue("STEP\_NAME\_4")

testRunner.runTestStep(project.testSuites[sheetName].testCases["B\_TestCase"].testSteps[request\_4])

resultList.add(context.testCase.getPropertyValue("RESPONSE\_STATUS"))

// Pass this value to XML and Start run the test step - JdbcRequest

def request\_5 = context.testCase.getPropertyValue("STEP\_NAME\_5")

testRunner.runTestStep(project.testSuites[sheetName].testCases["B\_TestCase"].testSteps[request\_5])

resultList.add(context.testCase.getPropertyValue("RESPONSE\_STATUS"))

///////////////////////////////////////////////////////////////////////////////////

String tSuiteName = sheetName

String tcId = context.testCase.getPropertyValue("TC\_ID")

// Get consolidated result of all Requests and store it to the LinkedHashMap

LinkedHashMap<String,String> suitesInfo = new LinkedHashMap<String,String>();

suitesInfo.put("SUITE\_NAME",tSuiteName)

suitesInfo.put("TC\_ID",tcId)

suitesInfo.put(request\_1,context.testCase.getPropertyValue(request\_1 + "\_RESPONSE\_STATUS"))

suitesInfo.put(request\_2,context.testCase.getPropertyValue(request\_2 + "\_RESPONSE\_STATUS"))

suitesInfo.put(request\_3,context.testCase.getPropertyValue(request\_3 + "\_RESPONSE\_STATUS"))

suitesInfo.put(request\_4,context.testCase.getPropertyValue(request\_4 + "\_RESPONSE\_STATUS"))

suitesInfo.put(request\_5,context.testCase.getPropertyValue(request\_5 + "\_RESPONSE\_STATUS"))

// Pass the tcID, tSuiteName and Map to the method writeTestSuiteStatusInExcel()

writeExcel(tSuiteName,tcId,suitesInfo)

///////////////////////////////////////////////////////////////////////////////////

// Update the result in the excel sheet against the 'TestCase'

FileInputStream fis = new FileInputStream(new File(filePath))

HSSFWorkbook wBook = new HSSFWorkbook(fis)

HSSFSheet hSheet = wBook.getSheet(sheetName)

org.apache.poi.ss.usermodel.Cell resultCell = null;

// 'resultList' contains each service's execution status e.g Pass/Fail

def resCount = 0

for (int x=0; x < resultList.size(); x++)

{

if (resultList[x].equalsIgnoreCase("PASS"))

{

resCount++;

}

}

// If 'resultList.size' and 'resCount' are equal,

// update testcase status as PASS

// Else update it as FAIL

if (resultList.size() == resCount)

{

// Set the cell values Request and Response

resultCell = hSheet.getRow(row).getCell(2)

resultCell.setCellValue("PASS")

// Highlight the status cell - Background

HSSFCellStyle style = wBook.createCellStyle()

style.setFillForegroundColor(IndexedColors.GREEN.getIndex())

style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND)

// Foreground

HSSFFont font = wBook.createFont()

font.setColor(HSSFColor.WHITE.index)

style.setFont(font)

// Apply border for the cells

style.setBorderBottom(HSSFCellStyle.BORDER\_THIN)

style.setBorderTop(HSSFCellStyle.BORDER\_THIN)

style.setBorderRight(HSSFCellStyle.BORDER\_THIN)

style.setBorderLeft(HSSFCellStyle.BORDER\_THIN)

resultCell.setCellStyle(style)

// Assign the corresponding TestSuite status as 'FAIL'

testRunner.testCase.testSuite.project.setPropertyValue("SUITE\_RESULT", "PASS")

}

else

{

// Set the cell values Request and Response

resultCell = hSheet.getRow(row).getCell(2)

resultCell.setCellValue("FAIL")

// Highlight the status cell - Background

HSSFCellStyle style = wBook.createCellStyle()

style.setFillForegroundColor(IndexedColors.RED.getIndex())

style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND)

// Foreground

HSSFFont font = wBook.createFont()

font.setColor(HSSFColor.WHITE.index)

style.setFont(font)

// Apply border for the cells

style.setBorderBottom(HSSFCellStyle.BORDER\_THIN)

style.setBorderTop(HSSFCellStyle.BORDER\_THIN)

style.setBorderRight(HSSFCellStyle.BORDER\_THIN)

style.setBorderLeft(HSSFCellStyle.BORDER\_THIN)

resultCell.setCellStyle(style)

// Assign the corresponding TestSuite status as 'FAIL'

testRunner.testCase.testSuite.project.setPropertyValue("SUITE\_RESULT", "FAIL")

}

// Update the excel file with the Testcase Status (Pass/Fail)

FileOutputStream fout = new FileOutputStream(new File(filePath))

wBook.write(fout)

fout.close()

fis.close()

//wBook.close()

}

}

// Update the TestSuite status in the "BaseSheet"

String strSuiteResult = testRunner.testCase.testSuite.project.getPropertyValue("SUITE\_RESULT")

// Open the TestSuite status in the BaseSheet

FileInputStream fis1 = new FileInputStream(new File(filePath))

HSSFWorkbook pWbook = new HSSFWorkbook(fis1)

HSSFSheet pSheet = pWbook.getSheetAt(0) // BaseSheet

org.apache.poi.ss.usermodel.Cell suiteResultCell = null;

org.apache.poi.ss.usermodel.Cell suiteTimeCell = null;

// Set the status of TestSuite in the appropriate cell

suiteResultCell = pSheet.getRow(baseRow).getCell(2)

suiteResultCell.setCellValue(strSuiteResult)

if (strSuiteResult.equalsIgnoreCase("PASS"))

{

// Highlight the status cell - Background

HSSFCellStyle style = pWbook.createCellStyle()

style.setFillForegroundColor(IndexedColors.GREEN.getIndex())

style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND)

// Foreground

HSSFFont font = pWbook.createFont()

font.setColor(HSSFColor.WHITE.index)

style.setFont(font)

// Apply border for the cells

style.setBorderBottom(HSSFCellStyle.BORDER\_THIN)

style.setBorderTop(HSSFCellStyle.BORDER\_THIN)

style.setBorderRight(HSSFCellStyle.BORDER\_THIN)

style.setBorderLeft(HSSFCellStyle.BORDER\_THIN)

suiteResultCell.setCellStyle(style)

}

else

{

// Highlight the status cell - Background

HSSFCellStyle style = pWbook.createCellStyle();

style.setFillForegroundColor(IndexedColors.RED.getIndex())

style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND)

// Foreground

HSSFFont font = pWbook.createFont()

font.setColor(HSSFColor.WHITE.index)

style.setFont(font)

// Apply border for the cells

style.setBorderBottom(HSSFCellStyle.BORDER\_THIN)

style.setBorderTop(HSSFCellStyle.BORDER\_THIN)

style.setBorderRight(HSSFCellStyle.BORDER\_THIN)

style.setBorderLeft(HSSFCellStyle.BORDER\_THIN)

suiteResultCell.setCellStyle(style)

}

// Find total time taken for the test suite

String endTime = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.getInstance().getTime())

// Invoke TimeDifference method

String strSuiteTime = getDiffTime(suiteStartTime,endTime)

// Set the Total Time Taken of TestSuite in the appropriate cell

suiteTimeCell = pSheet.getRow(baseRow).getCell(3)

suiteTimeCell.setCellValue(strSuiteTime)

// Update the excel file with Request and Response value

FileOutputStream fout1 = new FileOutputStream(new File(filePath))

pWbook.write(fout1)

fout1.close()

fis1.close()

}

else if (sheetName.equals("C"))

{

String suiteStartTime = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.getInstance().getTime());

log.info "C testsuite executing..." + "\n"

// Navigate to the respective TestSuite -> TestCase sheet

Sheet tcSheet = book.getSheet(sheetName)

int rows = tcSheet.getRows()

int cols = tcSheet.getColumns()

// This loop iterates till the "A" suite sheet rows

for (int row = 1; row < rows; row++)

{

if (tcSheet.getCell(1,row).getContents().equalsIgnoreCase("Y"))

{

def columnList = []

def resultList = []

for (int col = 0; col < cols; col++)

{

columnList.add(tcSheet.getCell(col,row).getContents())

}

context.testCase.setPropertyValue("TC\_ID", columnList[0])

context.testCase.setPropertyValue("FLAG", columnList[1])

context.testCase.setPropertyValue("RESULT", columnList[2])

context.testCase.setPropertyValue("STEP\_NAME\_1", columnList[3])

context.testCase.setPropertyValue("USERNAME", columnList[4])

context.testCase.setPropertyValue("PASSWORD", columnList[5])

context.testCase.setPropertyValue("ENCRYPTED", columnList[6])

context.testCase.setPropertyValue("ORS\_ID", columnList[7])

context.testCase.setPropertyValue("SYSTEM\_NAME", columnList[8])

context.testCase.setPropertyValue("STRING\_VALUE1", columnList[9])

context.testCase.setPropertyValue("STRING\_NAME1", columnList[10])

context.testCase.setPropertyValue("STRING\_VALUE2", columnList[11])

context.testCase.setPropertyValue("STRING\_NAME2", columnList[12])

context.testCase.setPropertyValue("STRING\_VALUE3", columnList[13])

context.testCase.setPropertyValue("STRING\_NAME3", columnList[14])

context.testCase.setPropertyValue("STRING\_VALUE4", columnList[15])

context.testCase.setPropertyValue("STRING\_NAME4", columnList[16])

context.testCase.setPropertyValue("SIPERIAN\_OBJECT\_UID", columnList[17])

context.testCase.setPropertyValue("GENERATE\_SOURCE\_KEY", columnList[18])

context.testCase.setPropertyValue("STEP\_NAME\_2", columnList[19])

context.testCase.setPropertyValue("ISASYNCHRONOUS", columnList[20])

context.testCase.setPropertyValue("STEP\_NAME\_3", columnList[21])

// Get the project name and execute the TestSuite - TestCases of TestSteps one by one

def project = testRunner.testCase.testSuite.project

// Pass this value to XML and Start run the test step - PutRequest

def request\_1 = context.testCase.getPropertyValue("STEP\_NAME\_1")

testRunner.runTestStep(project.testSuites['C'].testCases['TestCase3'].testSteps[request\_1])

resultList.add(context.testCase.getPropertyValue("RESPONSE\_STATUS"))

// Pass this value to XML and Start run the test step - GetRequest

def request\_2 = context.testCase.getPropertyValue("STEP\_NAME\_2")

testRunner.runTestStep(project.testSuites['C'].testCases['TestCase3'].testSteps[request\_2])

resultList.add(context.testCase.getPropertyValue("RESPONSE\_STATUS"))

def request\_3 = context.testCase.getPropertyValue("STEP\_NAME\_3")

testRunner.runTestStep(project.testSuites['C'].testCases['TestCase3'].testSteps[request\_3])

//resultList.add(context.testCase.getPropertyValue("RESPONSE\_STATUS"))

// Update the result in the excel sheet against the 'TestCase'

FileInputStream fis = new FileInputStream(new File(filePath))

HSSFWorkbook wBook = new HSSFWorkbook(fis)

HSSFSheet hSheet = wBook.getSheet(sheetName)

org.apache.poi.ss.usermodel.Cell resultCell = null;

// 'resultList' contains each service's execution status e.g Pass/Fail

def resCount = 0

for (int x=0; x < resultList.size(); x++)

{

if (resultList[x].equalsIgnoreCase("PASS"))

{

resCount++;

}

}

// If 'resultList.size' and 'resCount' are equal,

// update testcase status as PASS

// Else update it as FAIL

if (resultList.size() == resCount)

{

// Set the cell values Request and Response

resultCell = hSheet.getRow(row).getCell(2)

resultCell.setCellValue("PASS")

// Highlight the status cell - Background

HSSFCellStyle style = wBook.createCellStyle()

style.setFillForegroundColor(IndexedColors.GREEN.getIndex())

style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND)

// Foreground

HSSFFont font = wBook.createFont()

font.setColor(HSSFColor.WHITE.index)

style.setFont(font)

// Apply border for the cells

style.setBorderBottom(HSSFCellStyle.BORDER\_THIN)

style.setBorderTop(HSSFCellStyle.BORDER\_THIN)

style.setBorderRight(HSSFCellStyle.BORDER\_THIN)

style.setBorderLeft(HSSFCellStyle.BORDER\_THIN)

resultCell.setCellStyle(style)

// Assign the corresponding TestSuite status as 'FAIL'

testRunner.testCase.testSuite.project.setPropertyValue("SUITE\_RESULT", "PASS")

}

else

{

// Set the cell values Request and Response

resultCell = hSheet.getRow(row).getCell(2)

resultCell.setCellValue("FAIL")

// Highlight the status cell - Background

HSSFCellStyle style = wBook.createCellStyle()

style.setFillForegroundColor(IndexedColors.GREEN.getIndex())

style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND)

// Foreground

HSSFFont font = wBook.createFont()

font.setColor(HSSFColor.WHITE.index)

style.setFont(font)

// Apply border for the cells

style.setBorderBottom(HSSFCellStyle.BORDER\_THIN)

style.setBorderTop(HSSFCellStyle.BORDER\_THIN)

style.setBorderRight(HSSFCellStyle.BORDER\_THIN)

style.setBorderLeft(HSSFCellStyle.BORDER\_THIN)

resultCell.setCellStyle(style)

// Assign the corresponding TestSuite status as 'FAIL'

testRunner.testCase.testSuite.project.setPropertyValue("SUITE\_RESULT", "FAIL")

}

// Update the excel file with the Testcase Status (Pass/Fail)

FileOutputStream fout = new FileOutputStream(new File(filePath))

wBook.write(fout)

fout.close()

fis.close()

//wBook.close()

}

}

// Update the TestSuite status in the "BaseSheet"

String strSuiteResult = testRunner.testCase.testSuite.project.getPropertyValue("SUITE\_RESULT")

// Open the TestSuite status in the BaseSheet

FileInputStream fis1 = new FileInputStream(new File(filePath))

HSSFWorkbook pWbook = new HSSFWorkbook(fis1)

HSSFSheet pSheet = pWbook.getSheetAt(0) // BaseSheet

org.apache.poi.ss.usermodel.Cell suiteResultCell = null

org.apache.poi.ss.usermodel.Cell suiteTimeCell = null

// Set the status of TestSuite in the appropriate cell

suiteResultCell = pSheet.getRow(baseRow).getCell(2)

suiteResultCell.setCellValue(strSuiteResult)

if (strSuiteResult.equalsIgnoreCase("PASS"))

{

// Highlight the status cell - Background

HSSFCellStyle style = pWbook.createCellStyle()

style.setFillForegroundColor(IndexedColors.GREEN.getIndex())

style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND)

// Foreground

HSSFFont font = pWbook.createFont()

font.setColor(HSSFColor.WHITE.index)

style.setFont(font)

// Apply border for the cells

style.setBorderBottom(HSSFCellStyle.BORDER\_THIN)

style.setBorderTop(HSSFCellStyle.BORDER\_THIN)

style.setBorderRight(HSSFCellStyle.BORDER\_THIN)

style.setBorderLeft(HSSFCellStyle.BORDER\_THIN)

suiteResultCell.setCellStyle(style)

}

else

{

// Highlight the status cell - Background

HSSFCellStyle style = pWbook.createCellStyle()

style.setFillForegroundColor(IndexedColors.RED.getIndex())

style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND)

// Foreground

HSSFFont font = pWbook.createFont()

font.setColor(HSSFColor.WHITE.index)

style.setFont(font)

// Apply border for the cells

style.setBorderBottom(HSSFCellStyle.BORDER\_THIN)

style.setBorderTop(HSSFCellStyle.BORDER\_THIN)

style.setBorderRight(HSSFCellStyle.BORDER\_THIN)

style.setBorderLeft(HSSFCellStyle.BORDER\_THIN)

suiteResultCell.setCellStyle(style)

}

// Find total time taken for the test suite

String endTime = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.getInstance().getTime())

// Invoke TimeDifference method

String strSuiteTime = getDiffTime(suiteStartTime,endTime)

// Set the Total Time Taken of TestSuite in the appropriate cell

suiteTimeCell = pSheet.getRow(baseRow).getCell(3)

suiteTimeCell.setCellValue(strSuiteTime)

// Update the excel file with Request and Response value

FileOutputStream fout1 = new FileOutputStream(new File(filePath))

pWbook.write(fout1)

fout1.close()

fis1.close()

}

}

}

book.close()

log.info ("Execution Finished On :" + new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.getInstance().getTime()) );

// Method that returns the difference between two date times

public String getDiffTime(String startTime, String endTime)

{

// Calculate Total Time taken for the execution

SimpleDateFormat format = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss")

Date d1 = null;

Date d2 = null;

d1 = format.parse(startTime)

d2 = format.parse(endTime)

long diff = d2.getTime() - d1.getTime()

long diffInSecs = TimeUnit.MILLISECONDS.toSeconds(diff)

return diffInSecs

}

// This method writes the testsuite, test cases

public void writeExcel(String tSuite, String tc, LinkedHashMap <String, String> testSteps)

{

String xlFile = "D:\\working\\POC\\RunManager.xls"

try

{

FileInputStream inputStream = new FileInputStream(new File(xlFile))

HSSFWorkbook myBook = new HSSFWorkbook(inputStream)

HSSFSheet mySheet = myBook.getSheet("TestSuiteSummary")

int maxRows = mySheet.getPhysicalNumberOfRows()

int col = 0

// Dealing with BLANK sheet / rows

if (maxRows == 0)

{

readAndUpdateBlankExcelCells(tSuite, tc, testSteps)

}

else

{

HSSFRow row = mySheet.createRow(maxRows)

int cl = 0

for (Map.Entry<String, String> entry : testSteps)

{

HSSFCell cell = row.createCell(cl).setCellValue(entry.getKey())

cl++

}

HSSFRow dataRow = mySheet.createRow(maxRows+1)

int c = 0

for (Map.Entry<String, String> entry : testSteps)

{

HSSFCell cell = dataRow.createCell(c).setCellValue(entry.getValue())

c++

}

inputStream.close()

FileOutputStream fileOut = new FileOutputStream(xlFile)

myBook.write(fileOut)

fileOut.flush()

fileOut.close()

}

}

catch(Exception ex)

{

System.out.println("Inside catch")

ex.printStackTrace()

}

}

// Dealing with Blank Cells

public void readAndUpdateBlankExcelCells(String tSuite, String tc, LinkedHashMap <String, String> steps)

{

String xlPath = "D:\\working\\POC\\RunManager.xls"

try

{

FileInputStream inputStream = new FileInputStream(new File(xlPath))

HSSFWorkbook myBook = new HSSFWorkbook(inputStream)

HSSFSheet sheet = myBook.getSheet("TestSuiteSummary")

for (int rw = 0; rw < 1; rw++)

{

HSSFRow row = sheet.createRow(rw)

int cl = 0

for (Map.Entry<String, String> entry : steps)

{

HSSFCell cell = row.createCell(cl).setCellValue(entry.getKey())

cl++

}

HSSFRow dataRow = sheet.createRow(sheet.getLastRowNum()+1)

int c = 0

for (Map.Entry<String, String> entry : steps)

{

HSSFCell cell = dataRow.createCell(c).setCellValue(entry.getValue()) c++

}

}

inputStream.close()

FileOutputStream fileOut = new FileOutputStream(xlPath)

myBook.write(fileOut)

fileOut.flush()

fileOut.close()

}

catch(Exception ex)

{

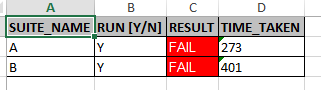
System.out.println("Inside catch")

ex.printStackTrace()

}

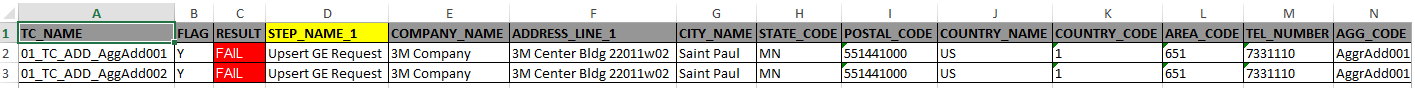
}

1. Datasheet is categorized into three parts
   1. BaseSheet
   2. TestSuites
   3. TestSuiteSummary
2. **BaseSheet** looks like the following:



In the **BaseSheet**, test suite execution will be controlled by the Flag [Y/N]. After executed of the each test suite results and the timing will be captured in the following columns (RESULT and TIME\_TAKEN)

1. **TestSuite** sheets look like,

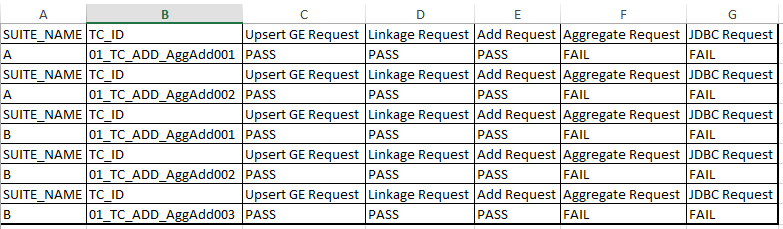


In this sheet, Testdata will be given for the each test case. Each test case can be controlled by the Flag value [Y/N]. Based on the execution of the test case, RESULT column will be filled with status and highlighted with an appropriate background color. **Example:** Green for PASS, Red for FAIL

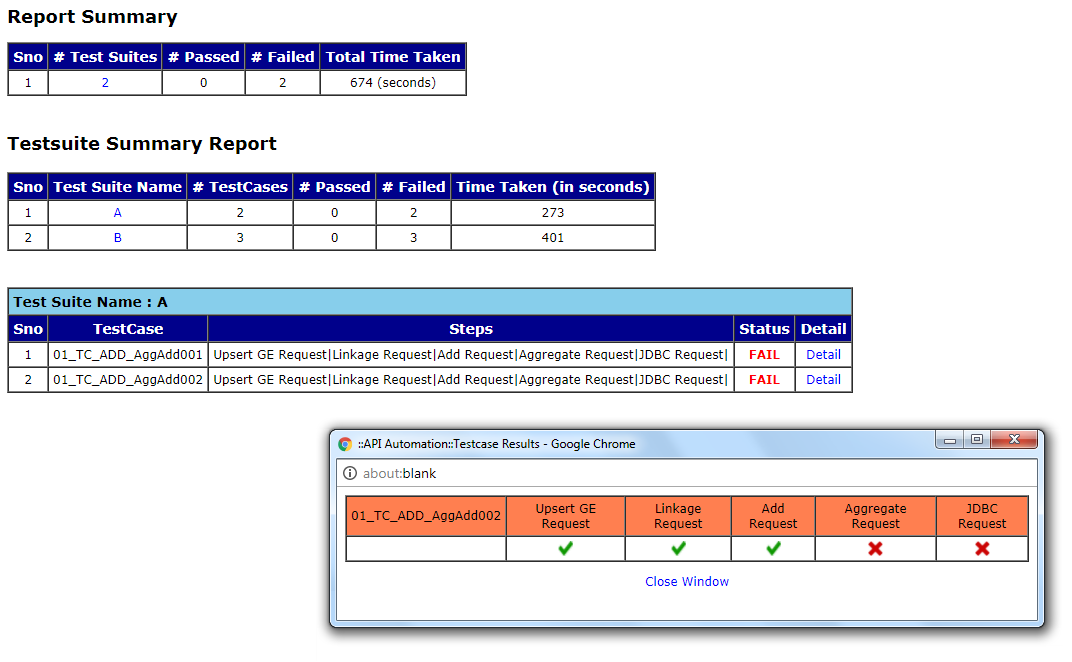
**Note: Testsuite** sheets should be created based on the **BaseSheet -> Suite Name** list

* **TC\_NAME** should be unique in all the Test Suite sheets, same test case name should not be present anywhere in the test suite sheets.
* **STEP\_NAME\_X** should contain the word like “Request” in each test case

1. **TestSuiteSummary** sheet contains the following data:



This sheet contains detailed summary of the test suites will be recorded against each test cases by Groovy scripting. From this excel sheet, an HTML report will be generated by **GenerateReport** java class. And the final HTML report will be look like this.



As mentioned earlier, the above HTML reports will be generated dynamically by **GenerateReport** java class and it will be invoked by the Groovy script. For that, we need to convert the class as an executable JAR file and kept it in to **\SoapUI-5.2.1\bin\ext** location. And then, call the java class from the groovy script as mentioned below.

*import utilities.GenerateReport*

*GenerateReport.buildTestSuiteSummaryReport()*