Contents

1. Introduction

1.1 Problem That I am Solving:

As part of this project I developed a web based tool which will generate a .ROBOT script for testing various Negative triggers. The script generated by this tool is readily executable without any additional changes or efforts.

1.2 Background Information:

In my current job as part of functional test team, my primary responsibilities includes testing of various new L2 and L3 routing functionality.

Typical tasks involved our day to day job are:

a) Designing test network topology.
b) Write FTP (Functional Test Plan).
c) Build the testbed.
d) Automate all the TCs (test cases) in FTP.
e) Perform Testing using scripts developed.

Each FTP contains good amount (approximately 30%) of Negative TCs.
In our company we use ROBOT frame work for our automation.

Typical Steps involved in a Negative TC are:

a) Load Initial Configuration on all the devices in the testbed.
b) Perform Checks in Steady state – Check that all primary protocols are UP and control and Forwarding Plane entries exists as expected.
c) Perform Negative Trigger on DUT (Device Under Test)
d) Check again all primary protocols are UP and control and Forwarding Plane entries exists as expected (after negative trigger)
e) Perform core functionality check, to ensure DUT is recovered after Negative trigger.

Typically testers will write KW (Keywords – Equivalent to Functions/Subroutines in other languages) for functional TCs (which covers steps# a, b, d and e mentioned above). For Negative Triggers, the only additional step is #c. This tool reuses the KWs already developed by the tester for any kind of Checks and only adds script snippet related to Negative triggers and any common checks.

Here are the typical Negative Triggers we use:

a) Flap Protocols – Protocols like OSPF, ISIS, BGP, MPLS, etc.
b) Restart Daemons – Daemons like RPD, L2ALD, Chassisd, etc.
c) Links Flap – Core links, access links, etc.
d) High Availability Tests – GR, NSR, ISSU, etc.

Advantages of this tool:

1. Saves time. Though the time saved per user is just few hours, it will be huge if we consider the total time saved for the whole team.
2. Include Common Checks like CPU Hogs, memory leaks, errors in log files, etc – Typically testers focus on the checks related to their functionality and tend to ignore common check (due to lck of knowledge or lack of time) and this tool covers such checks by adding KW related to common checks automatically.
3. This approach can be extended for other kind of tasks.

2. Requirements

Here is the list of all the Python modules that need to be installed to run my program:

```
import re
from flask import Flask, render_template, request, url_for, redirect, send_file
from werkzeug.datastructures import CombinedMultiDict, MultiDict
```

3. Description of my Python program

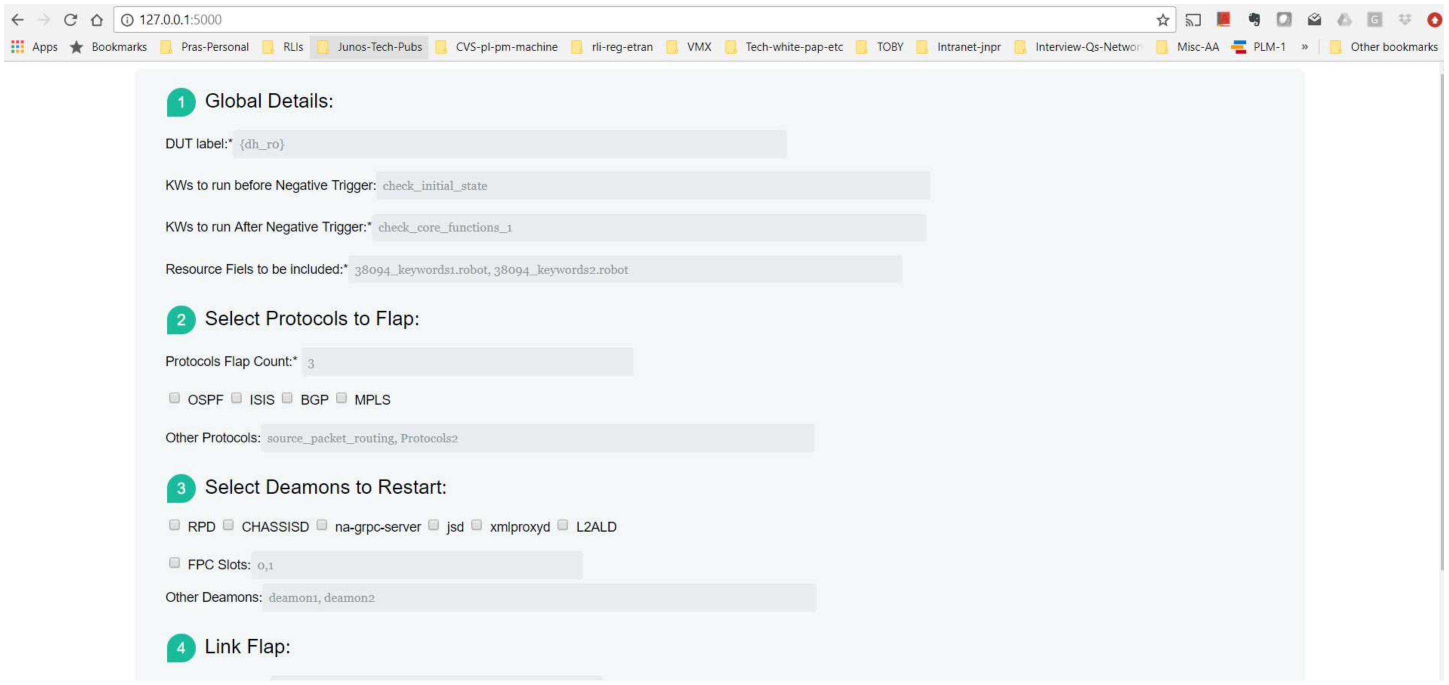
The program starts a webserver (using FLASK) and when users access the root HTML page, it will show a HTML form. Users will enter required fields and clicks on the SUBMIT button. Next the program takes to a different page where the user has option to download the automatically generated .ROBOT file or to go back to the main page to re-submit the form.

Steps involved in my python program execution:

1) Start Web server and users can access the main HTML form from http://127.0.0.1:5000/.
2) Users will enter the appropriate fields in the form and click on Submit.
3) Python program receives the data from HTML page and saves it into a python dictionary.
4) Python program reads each key in the dictionary and generates code for corresponding TC and writes it into a .ROBOT file.
5) After reading the whole Dictionary and generating and writing all the corresponding TCs, the file is closed and a 2nd HTML page is displayed to the user.
6) User has option to download the automatically generated .ROBOT file or to go back to the main page to re-submit the form.

4. Screenshots of the program output

Main HTML Form:



1 Global Details:

DUT label:* {dh_ro}

KWs to run before Negative Trigger: check_initial_state

KWs to run After Negative Trigger:* check_core_functions_1

Resource Fiels to be included:* 38094_keywords1.robot, 38094_keywords2.robot

2 Select Protocols to Flap:

Protocols Flap Count:* 3

☐ OSPF ☐ ISIS ☐ BGP ☐ MPLS

Other Protocols: source_packet_routing, Protocols2

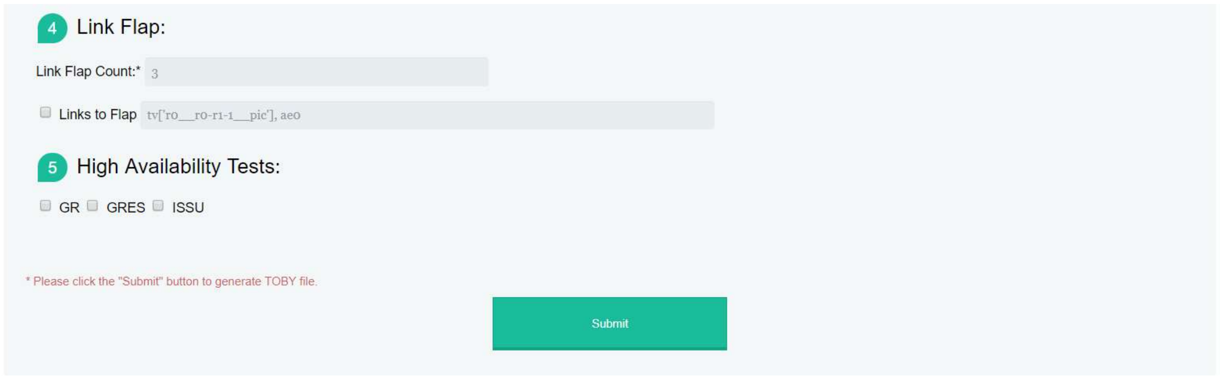
3 Select Deamons to Restart:

☐ RPD ☐ CHASSISD ☐ na-grpc-server ☐ jsd ☐ xmlproxyd ☐ L2ALD

☐ FPC Slots: 0,1

Other Deamons: deamon1, deamon2

4 Link Flap:



4 Link Flap:

Link Flap Count:* 3

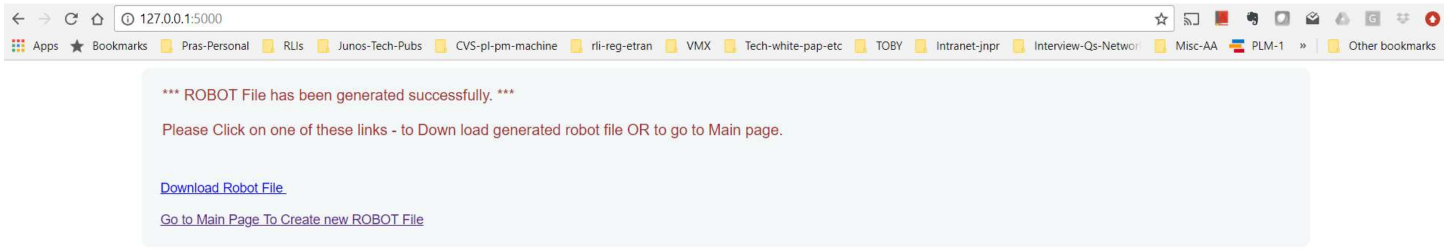
☐ Links to Flap tv['r0__r0-r1-1__pic'], aeo

5 High Availability Tests:

☐ GR ☐ GRES ☐ ISSU

* Please click the "Submit" button to generate TOBY file.

Results Download Page:



127.0.0.1:5000

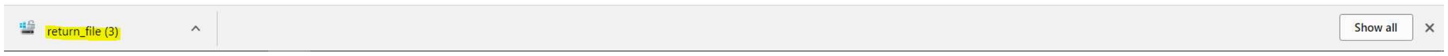
*** ROBOT File has been generated successfully. ***

Please Click on one of these links - to Down load generated robot file OR to go to Main page.

Download Robot File

Go to Main Page To Create new ROBOT File

Click on the Download Robot File option:

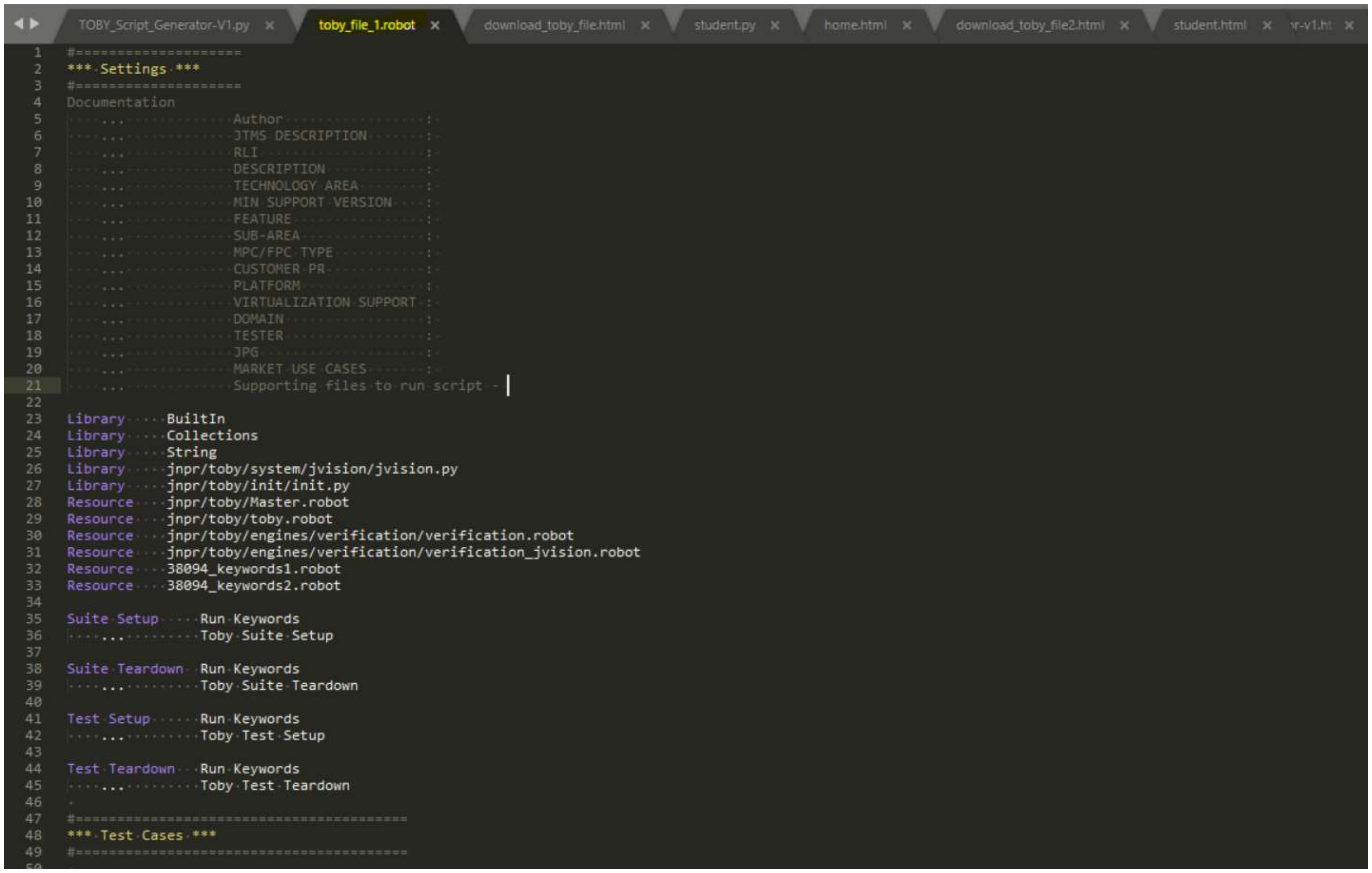


Click on Go to Main Page:

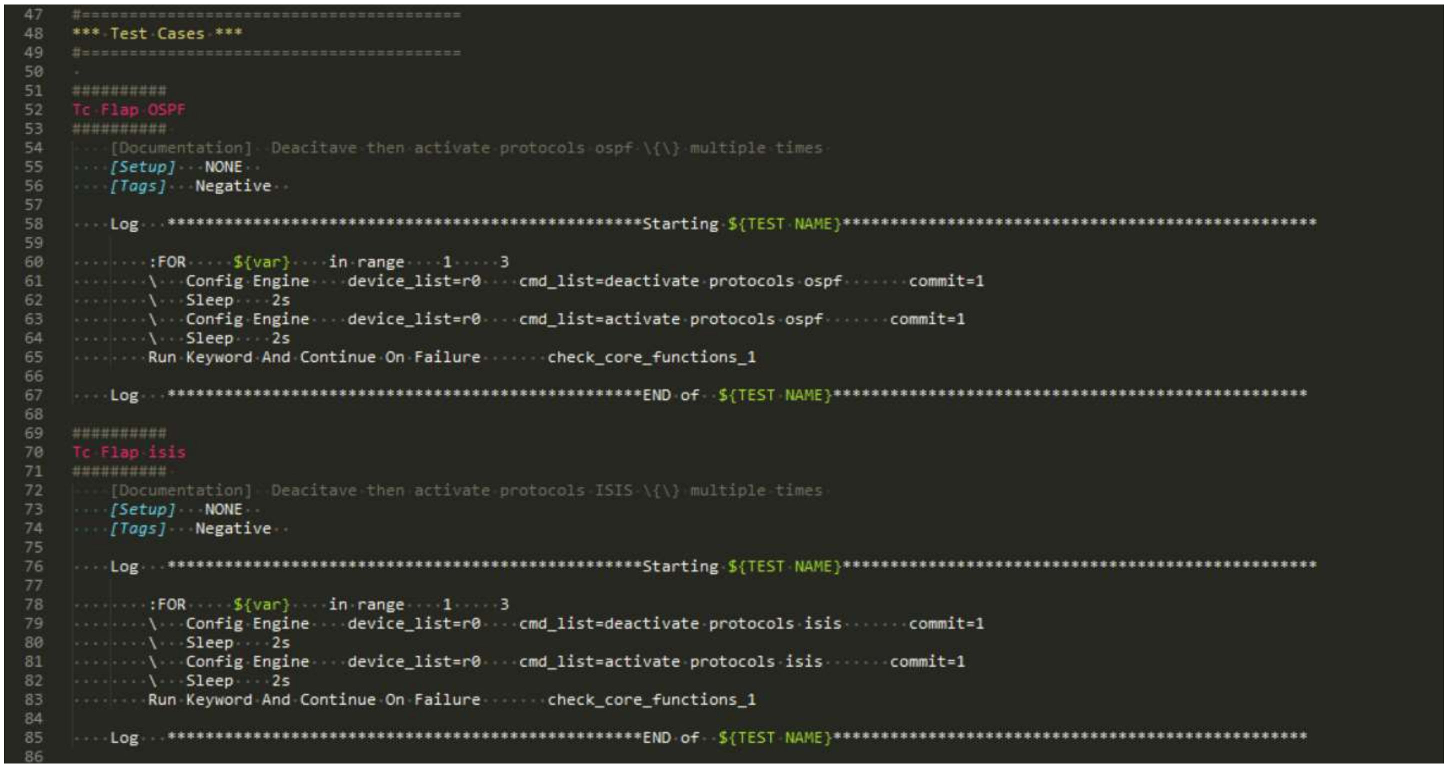
Control goes to the the Main HTML Page, shown above.

5. Conclusion

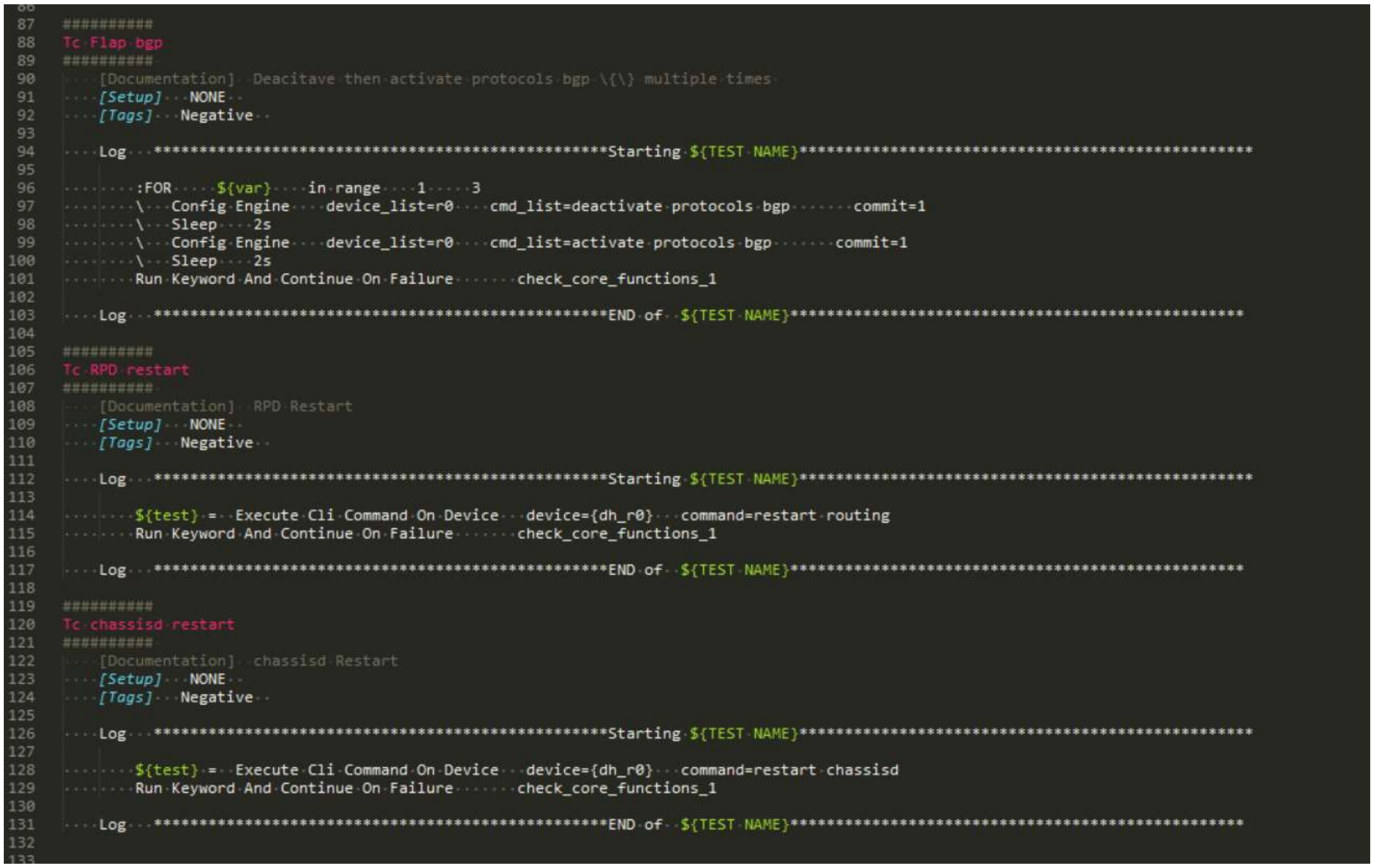
Using my program a .ROBOT file shown below is generated:



```
#====================
*** Settings ***
#====================
Documentation
    ...            Author                  :
    ...            JTMS DESCRIPTION        :
    ...            RLI                     :
    ...            DESCRIPTION             :
    ...            TECHNOLOGY AREA         :
    ...            MIN SUPPORT VERSION     :
    ...            FEATURE                 :
    ...            SUB-AREA                :
    ...            MPC/FPC TYPE            :
    ...            CUSTOMER PR             :
    ...            PLATFORM                :
    ...            VIRTUALIZATION SUPPORT  :
    ...            DOMAIN                  :
    ...            TESTER                  :
    ...            JPG                     :
    ...            MARKET USE CASES        :
    ...            Supporting files to run script  -

Library    BuiltIn
Library    Collections
Library    String
Library    jnpr/toby/system/jvision/jvision.py
Library    jnpr/toby/init/init.py
Resource   jnpr/toby/Master.robot
Resource   jnpr/toby/toby.robot
Resource   jnpr/toby/engines/verification/verification.robot
Resource   jnpr/toby/engines/verification/verification_jvision.robot
Resource   38094_keywords1.robot
Resource   38094_keywords2.robot

Suite Setup      Run Keywords
    ...          Toby Suite Setup

Suite Teardown   Run Keywords
    ...          Toby Suite Teardown

Test Setup       Run Keywords
    ...          Toby Test Setup

Test Teardown    Run Keywords
    ...          Toby Test Teardown

#========================================
*** Test Cases ***
#========================================
```



```
#========================================
*** Test Cases ***
#========================================

##########
Tc Flap OSPF
##########
    [Documentation]  Deacitave then activate protocols ospf \{\} multiple times
    [Setup]   NONE
    [Tags]   Negative

    Log   ******************************************************Starting ${TEST NAME}******************************************************

        :FOR     ${var}    in range    1     3
        \   Config Engine    device_list=r0    cmd_list=deactivate protocols ospf       commit=1
        \   Sleep    2s
        \   Config Engine    device_list=r0    cmd_list=activate protocols ospf       commit=1
        \   Sleep    2s
        Run Keyword And Continue On Failure       check_core_functions_1

    Log   ******************************************************END of  ${TEST NAME}******************************************************

##########
Tc Flap isis
##########
    [Documentation]  Deacitave then activate protocols ISIS \{\} multiple times
    [Setup]   NONE
    [Tags]   Negative

    Log   ******************************************************Starting ${TEST NAME}******************************************************

        :FOR     ${var}    in range    1     3
        \   Config Engine    device_list=r0    cmd_list=deactivate protocols isis       commit=1
        \   Sleep    2s
        \   Config Engine    device_list=r0    cmd_list=activate protocols isis       commit=1
        \   Sleep    2s
        Run Keyword And Continue On Failure       check_core_functions_1

    Log   ******************************************************END of  ${TEST NAME}******************************************************
```



```
##########
Tc Flap bgp
##########
    [Documentation]  Deacitave then activate protocols bgp \{\} multiple times
    [Setup]   NONE
    [Tags]   Negative

    Log   **************************************************Starting ${TEST NAME}**************************************************

        :FOR     ${var}    in range    1     3
        \   Config Engine    device_list=r0    cmd_list=deactivate protocols bgp       commit=1
        \   Sleep    2s
        \   Config Engine    device_list=r0    cmd_list=activate protocols bgp       commit=1
        \   Sleep    2s
        Run Keyword And Continue On Failure       check_core_functions_1

    Log   **************************************************END of  ${TEST NAME}**************************************************

##########
Tc RPD restart
##########
    [Documentation]  RPD Restart
    [Setup]   NONE
    [Tags]   Negative

    Log   **************************************************Starting ${TEST NAME}**************************************************

        ${test} =  Execute Cli Command On Device   device={dh_r0}   command=restart routing
        Run Keyword And Continue On Failure       check_core_functions_1

    Log   **************************************************END of  ${TEST NAME}**************************************************

##########
Tc chassisd restart
##########
    [Documentation]  chassisd Restart
    [Setup]   NONE
    [Tags]   Negative

    Log   **************************************************Starting ${TEST NAME}**************************************************

        ${test} =  Execute Cli Command On Device   device={dh_r0}   command=restart chassisd
        Run Keyword And Continue On Failure       check_core_functions_1

    Log   **************************************************END of  ${TEST NAME}**************************************************
```

6. Python program

```
import os, re
from flask import Flask, render_template, request, url_for, redirect, json, send_file
from werkzeug.datastructures import CombinedMultiDict, MultiDict

app = Flask(__name__)

@app.route('/',methods=['GET', 'POST'])
def form1():
    if request.method == "POST":
        create_toby_file(request.values)
        #return render_template('download_toby_file.html')
        return render_template('download_toby_file2.html')
    else:
        return render_template('toby_script_generator-v1.html')

@app.route('/return_file')
```

```
def return_file():
    return send_file('toby_file_1.robot')

@app.route('/results',methods=['GET', 'POST'])
def form1results():
    if request.method == "POST":
        return render_template('form1result.html')
    else:
        return render_template('form1result.html')

def create_toby_file(data_from_form):
    print ("inside create_toby_file")
    form_dict = data_from_form.to_dict()
    print(form_dict)

    patterns = [',']
    dut_handle = form_dict.get('dut_handle', 'dh_none')
    #
    kw0 = form_dict.get('kw0')
    if (kw0):
        if re.search(',', kw0):
            kw0 = kw0.split(',')
            kw0 = [x.strip() for x in kw0]
        else:
            kw0 = kw0.strip()

    #
    kw1 = form_dict.get('kw1')
    if (kw1):
        if re.search(',', kw1):
            kw1 = kw1.split(',')
            kw1 = [x.strip() for x in kw1]
        else:
            kw1 = kw1.strip()

    #

    fr   =   open("C:\\Users\\pgudipati\\Me-Pras-Cloud\\Technical-Docs\\Python-UCSC\\Project\\toby_file_1.robot",
'w')

    fr.write('''#====================\n''')
    fr.write('''*** Settings ***\n''')
    fr.write('''#====================\n''')
    fr.write('''Documentation\n''')
    fr.write('''    ...             Author                 : \n''')
    fr.write('''    ...             JTMS DESCRIPTION       : \n''')
    fr.write('''    ...             RLI                    : \n''')
    fr.write('''    ...             DESCRIPTION            : \n''')
    fr.write('''    ...             TECHNOLOGY AREA        : \n''')
    fr.write('''    ...             MIN SUPPORT VERSION    : \n''')
    fr.write('''    ...             FEATURE                : \n''')
    fr.write('''    ...             SUB-AREA               : \n''')
    fr.write('''    ...             MPC/FPC TYPE           : \n''')
    fr.write('''    ...              CUSTOMER PR            : \n''')
    fr.write('''    ...              PLATFORM               : \n''')
    fr.write('''    ...              VIRTUALIZATION SUPPORT : \n''')
    fr.write('''    ...              DOMAIN                 : \n''')
    fr.write('''    ...              TESTER                 : \n''')
    fr.write('''    ...              JPG                    : \n''')
    fr.write('''    ...              MARKET USE CASES       : \n''')
    fr.write('''    ...              Supporting files to run script - \n''')
    fr.write('''\n''')

    fr.write('''Library     BuiltIn\n''')
    fr.write('''Library     Collections\n''')
    fr.write('''Library     String\n''')
    fr.write('''Library     jnpr/toby/system/jvision/jvision.py\n''')
    fr.write('''Library     jnpr/toby/init/init.py\n''')

    fr.write('''Resource    jnpr/toby/Master.robot\n''')
    fr.write('''Resource    jnpr/toby/toby.robot\n''')
    fr.write('''Resource    jnpr/toby/engines/verification/verification.robot\n''')
    fr.write('''Resource    jnpr/toby/engines/verification/verification_jvision.robot\n''')

    #
    resource_files = form_dict.get('resource_files')
    if resource_files:
        if re.search(',', resource_files):
            resource_files = resource_files.split(',')
            resource_files = [x.strip() for x in resource_files]
        else:
            resource_files = resource_files.strip()
    #
    if isinstance(resource_files, list):
        for i in resource_files:
            fr.write('''Resource    '''+i+'''\n''')
    else:
        fr.write('''Resource    '''+resource_files+'''\n''')

    fr.write('''\n''')
    fr.write('''Suite Setup     Run Keywords\n''')
    fr.write('''    ...         Toby Suite Setup\n''')
    fr.write('''\n''')
    fr.write('''Suite Teardown  Run Keywords\n''')
    fr.write('''    ...         Toby Suite Teardown\n''')
    fr.write('''\n''')
    fr.write('''Test Setup      Run Keywords\n''')
    fr.write('''    ...         Toby Test Setup\n''')
    fr.write('''\n''')
    fr.write('''Test Teardown   Run Keywords\n''')
    fr.write('''    ...         Toby Test Teardown\n''')

    fr.write(''' \n''')
    fr.write('''#========================================\n''')
    fr.write('''*** Test Cases ***\n''')
    fr.write('''#========================================\n''')
```

```
    fr.write(''' \n''')

# TCs for Protocols Flapping:
    # OSPF
    proto_flap_count = form_dict.get('prot_flap_count')
    ospf_flag = form_dict.get('ospf', 'No')
    if (ospf_flag == "yes"):
        fr.write('''##########\n''')
        fr.write('''Tc Flap OSPF\n''')
        fr.write('''########## \n''')
        fr.write('''    [Documentation]  Deacitave then activate protocols ospf \{\} multiple times \n''')
        fr.write('''    [Setup]   NONE  \n''')
        fr.write('''    [Tags]   Negative  \n\n''')
        fr.write('''             Log          **************************************************Starting   ${TEST
NAME}**************************************************\n\n''')
        fr.write('''        :FOR     ${var}    in range    1     ''' + str(proto_flap_count) + '''\n''')
        fr.write('''            \\    Config  Engine      device_list=r0      cmd_list=deactivate  protocols  ospf
commit=1\n''')
        fr.write('''        \\   Sleep    2s\n''')
        fr.write('''             \\    Config  Engine      device_list=r0      cmd_list=activate  protocols  ospf
commit=1\n''')
        fr.write('''        \\   Sleep    2s\n''')
        fr.write('''        Run Keyword And Continue On Failure       ''' + kw1 + '''\n\n''')
        fr.write('''            Log         **************************************************END   of      ${TEST
NAME}**************************************************\n\n''')

    # isis
    isis_flag = form_dict.get('isis', 'No')
    if (isis_flag == "yes"):
        fr.write('''##########\n''')
        fr.write('''Tc Flap isis\n''')
        fr.write('''########## \n''')
        fr.write('''    [Documentation]  Deacitave then activate protocols ISIS \{\} multiple times \n''')
        fr.write('''    [Setup]   NONE  \n''')
        fr.write('''    [Tags]   Negative  \n\n''')
        fr.write('''             Log          **************************************************Starting   ${TEST
NAME}**************************************************\n\n''')
        fr.write('''        :FOR     ${var}    in range    1     ''' + str(proto_flap_count) + '''\n''')
        fr.write('''            \\    Config  Engine      device_list=r0      cmd_list=deactivate  protocols  isis
commit=1\n''')
        fr.write('''        \\   Sleep    2s\n''')
        fr.write('''             \\    Config  Engine      device_list=r0      cmd_list=activate  protocols  isis
commit=1\n''')
        fr.write('''        \\   Sleep    2s\n''')
        fr.write('''        Run Keyword And Continue On Failure       ''' + kw1 + '''\n\n''')
        fr.write('''            Log         **************************************************END   of      ${TEST
NAME}**************************************************\n\n''')

    # bgp
    bgp_flag = form_dict.get('bgp', 'No')
    if (bgp_flag == "yes"):
        fr.write('''##########\n''')
        fr.write('''Tc Flap bgp\n''')
        fr.write('''########## \n''')
        fr.write('''    [Documentation]  Deacitave then activate protocols bgp \{\} multiple times \n''')
        fr.write('''    [Setup]   NONE  \n''')
        fr.write('''    [Tags]   Negative  \n\n''')
        fr.write('''             Log          **************************************************Starting   ${TEST
NAME}**************************************************\n\n''')
        fr.write('''        :FOR     ${var}    in range    1     ''' + str(proto_flap_count) + '''\n''')
        fr.write('''            \\     Config  Engine      device_list=r0      cmd_list=deactivate  protocols  bgp
commit=1\n''')
        fr.write('''        \\   Sleep    2s\n''')
        fr.write('''             \\     Config  Engine      device_list=r0      cmd_list=activate  protocols  bgp
commit=1\n''')
        fr.write('''        \\   Sleep    2s\n''')
        fr.write('''        Run Keyword And Continue On Failure       ''' + kw1 + '''\n\n''')
        fr.write('''            Log         **************************************************END   of      ${TEST
NAME}**************************************************\n\n''')

    # mpls
    mpls_flag = form_dict.get('mpls', 'No')
    if (mpls_flag == "yes"):
        fr.write('''##########\n''')
        fr.write('''Tc Flap mpls\n''')
        fr.write('''########## \n''')
        fr.write('''    [Documentation]  Deacitave then activate protocols mpls \{\} multiple times \n''')
        fr.write('''    [Setup]   NONE  \n''')
        fr.write('''    [Tags]   Negative  \n\n''')
        fr.write('''             Log          **************************************************Starting   ${TEST
NAME}**************************************************\n\n''')
        fr.write('''        :FOR     ${var}    in range    1     ''' + str(proto_flap_count) + '''\n''')
        fr.write('''            \\     Config  Engine      device_list=r0      cmd_list=deactivate  protocols  mpls
commit=1\n''')
        fr.write('''        \\   Sleep    2s\n''')
        fr.write('''             \\     Config  Engine      device_list=r0      cmd_list=activate  protocols  mpls
commit=1\n''')
        fr.write('''        \\   Sleep    2s\n''')
        fr.write('''        Run Keyword And Continue On Failure       ''' + kw1 + '''\n\n''')
        fr.write('''            Log         **************************************************END   of      ${TEST
NAME}**************************************************\n\n''')

# TCs for Deamons Restart:
    # RPD restart
    rpd_flag = form_dict.get('rpd', 'No')
    if (rpd_flag == "yes"):
        fr.write('''##########\n''')
        fr.write('''Tc RPD restart\n''')
        fr.write('''########## \n''')
        fr.write('''    [Documentation]  RPD Restart\n''')
        fr.write('''    [Setup]   NONE  \n''')
        fr.write('''    [Tags]   Negative  \n\n''')
        fr.write('''             Log          **************************************************Starting   ${TEST
NAME}**************************************************\n\n''')
        fr.write('''      ${test} =  Execute Cli Command On Device   device=''' + dut_handle + '''   command=restart
routing\n''')
```

```
        fr.write('''        Run Keyword And Continue On Failure       ''' + kw1 + '''\n\n''')
        fr.write('''            Log         **************************************************END   of      ${TEST
NAME}**************************************************\n\n''')

    #chassisd
    chassisd_flag = form_dict.get('chassisd', 'No')
    if (chassisd_flag == "yes"):
        fr.write('''##########\n''')
        fr.write('''Tc chassisd restart\n''')
        fr.write('''########## \n''')
        fr.write('''    [Documentation]  chassisd Restart\n''')
        fr.write('''    [Setup]   NONE  \n''')
        fr.write('''    [Tags]   Negative  \n\n''')
        fr.write('''             Log          **************************************************Starting   ${TEST
NAME}**************************************************\n\n''')
        fr.write('''      ${test} =  Execute Cli Command On Device   device=''' + dut_handle + '''   command=restart
chassisd\n''')
        fr.write('''        Run Keyword And Continue On Failure       ''' + kw1 + '''\n\n''')
        fr.write('''            Log         **************************************************END   of      ${TEST
NAME}**************************************************\n\n''')

# TCs for Links Flap:

    # End of File Writing =======================================
    fr.close()
    return True

if __name__ == "__main__":
    #host = os.getenv('IP', '127.0.0.1')
    #port = int(os.getenv('PORT', 5000))
    #print (host, port)
    app.run()
```