

Contents

1. Introduction	2
1.1 Problem That I am Solving	2
1.2 Background Information:	2
2. Requirements.....	3
3. Description of my Python program.....	3
4. Screenshots of the program output.....	3
Main HTML Form:	4
Results Download Page:.....	4
Click on the Download Robot File option:	4
Click on Go to Main Page:	5
5. Conclusion	5
6. Python program	7

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 lack 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:

127.0.0.1:5000

Apps★ BookmarksPras-PersonalRLIsJunos-Tech-PubsCVS-pl-pm-machinerli-reg-etranVMXTech-white-pap-etcTOBYIntranet-jnprInterview-Qs-NetworkMisc-AAPLM-1»Other bookmarks

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:

Link Flap Count:* 3

☐ Links to Flap tv[r0__r0-r1-1__pic], ae0

5 High Availability Tests:

☐ GR ☐ GRES ☐ ISSU

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

Submit

Results Download Page:

A screenshot of a web browser window. The address bar shows the URL "127.0.0.1:5000". The browser's bookmark bar is visible with various bookmarks like "Apps", "Bookmarks", "Pras-Personal", "RLIs", "Junos-Tech-Pubs", "CVS-pl-pm-machine", "rli-reg-etran", "VMX", "Tech-white-pap-etc", "TOBY", "Intranet-jnpr", "Interview-Qs-Network", "Misc-AA", "PLM-1", and "Other bookmarks". The main content area of the browser displays a message in red text: "*** ROBOT File has been generated successfully. ***". Below this message, it says "Please Click on one of these links - to Down load generated robot file OR to go to Main page." in red text. There are two blue underlined links: "Download Robot File" and "Go to Main Page To Create new ROBOT File".

Click on the Download Robot File option:

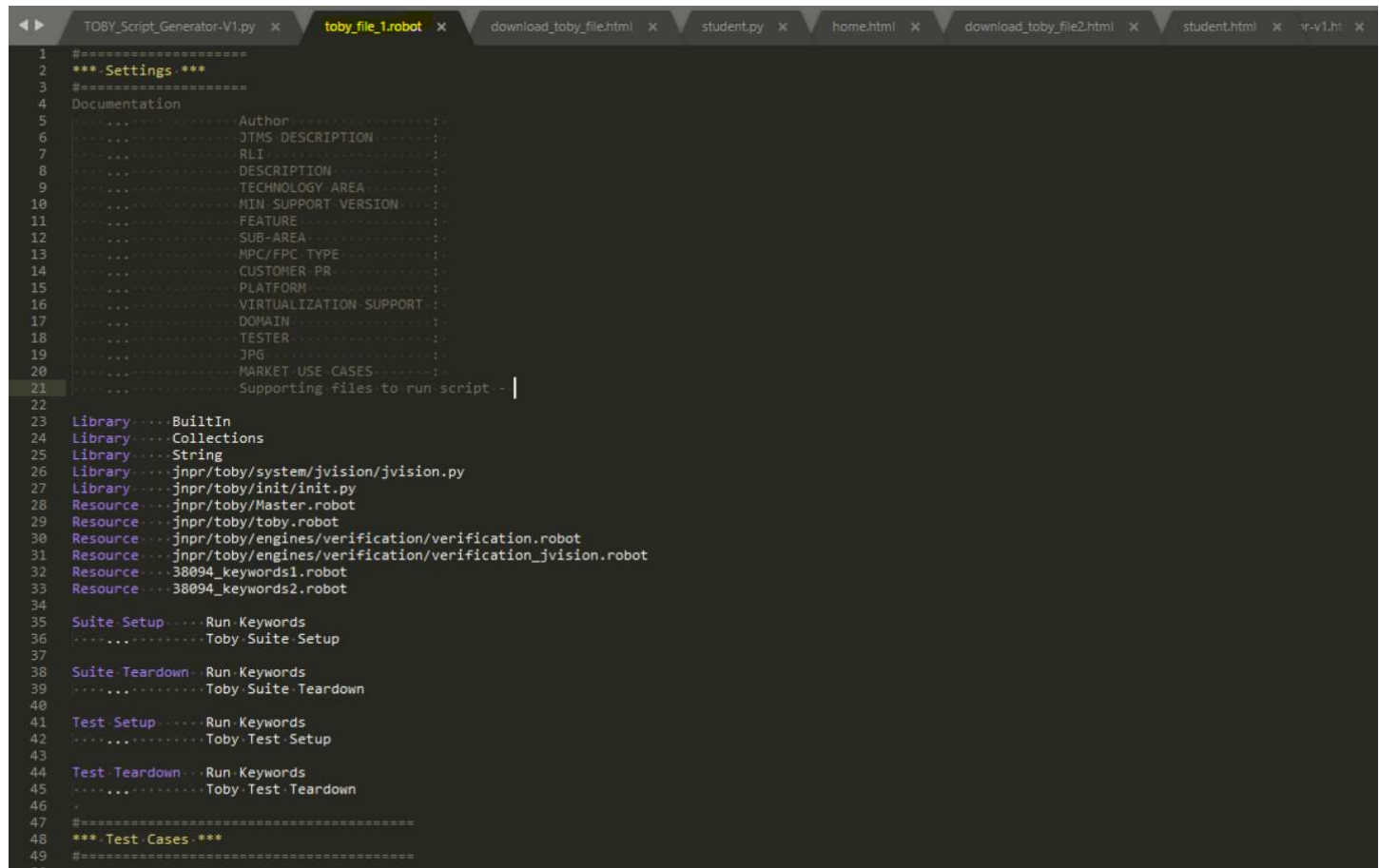
return_file (3) Show all

[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:



```
1 #####
2 *** Settings ***
3 #####
4 Documentation
5 .....Author .....
6 .....TMS DESCRIPTION .....
7 .....URL .....
8 .....DESCRIPTION .....
9 .....TECHNOLOGY AREA .....
10 .....MIN SUPPORT VERSION .....
11 .....FEATURE .....
12 .....SUB-AREA .....
13 .....MPC/FPC TYPE .....
14 .....CUSTOMER PR .....
15 .....PLATFORM .....
16 .....VIRTUALIZATION SUPPORT .....
17 .....DOMAIN .....
18 .....TESTER .....
19 .....JPG .....
20 .....MARKET USE CASES .....
21 .....Supporting files to run script - |
22
23 Library....BuiltIn
24 Library....Collections
25 Library....String
26 Library....jnpr/toby/system/jvision/jvision.py
27 Library....jnpr/toby/init/init.py
28 Resource....jnpr/toby/Master.robot
29 Resource....jnpr/toby/toby.robot
30 Resource....jnpr/toby/engines/verification/verification.robot
31 Resource....jnpr/toby/engines/verification/verification_jvision.robot
32 Resource....38094_keywords1.robot
33 Resource....38094_keywords2.robot
34
35 Suite Setup....Run Keywords
36 .....Toby Suite Setup
37
38 Suite Teardown....Run Keywords
39 .....Toby Suite Teardown
40
41 Test Setup....Run Keywords
42 .....Toby Test Setup
43
44 Test Teardown....Run Keywords
45 .....Toby Test Teardown
46
47 #####
48 *** Test Cases ***
49 #####
50
```

```

47 #####
48 *** Test Cases ***
49 #####
50 .
51 #####
52 Tc_Flap OSPF
53 #####
54 ....[Documentation].. Deactivate then activate protocols ospf \{\} multiple times
55 ....[Setup]... NONE..
56 ....[Tags]... Negative..
57
58 ....Log...*****Starting ${TEST_NAME}*****
59
60 .....:FOR....${var}....in range...1....3
61 .....\\...Config Engine... device_list=r0...cmd_list=deactivate protocols ospf.....commit=1
62 .....\\...Sleep... 2s
63 .....\\...Config Engine... device_list=r0...cmd_list=activate protocols ospf.....commit=1
64 .....\\...Sleep... 2s
65 .....Run Keyword And Continue On Failure.....check_core_functions_1
66
67 ....Log...*****END of ${TEST_NAME}*****
68
69 #####
70 Tc_Flap isis
71 #####
72 ....[Documentation].. Deactivate then activate protocols ISIS \{\} multiple times
73 ....[Setup]... NONE..
74 ....[Tags]... Negative..
75
76 ....Log...*****Starting ${TEST_NAME}*****
77
78 .....:FOR....${var}....in range...1....3
79 .....\\...Config Engine... device_list=r0...cmd_list=deactivate protocols isis.....commit=1
80 .....\\...Sleep... 2s
81 .....\\...Config Engine... device_list=r0...cmd_list=activate protocols isis.....commit=1
82 .....\\...Sleep... 2s
83 .....Run Keyword And Continue On Failure.....check_core_functions_1
84
85 ....Log...*****END of ${TEST_NAME}*****
86

```

```

87 #####
88 Tc_Flap bgp
89 #####
90 ....[Documentation].. Deactivate then activate protocols bgp \{\} multiple times
91 ....[Setup]... NONE..
92 ....[Tags]... Negative..
93
94 ....Log...*****Starting ${TEST_NAME}*****
95
96 .....:FOR....${var}....in range...1....3
97 .....\\...Config Engine... device_list=r0...cmd_list=deactivate protocols bgp.....commit=1
98 .....\\...Sleep... 2s
99 .....\\...Config Engine... device_list=r0...cmd_list=activate protocols bgp.....commit=1
100 .....\\...Sleep... 2s
101 .....Run Keyword And Continue On Failure.....check_core_functions_1
102
103 ....Log...*****END of ${TEST_NAME}*****
104
105 #####
106 Tc_RPD restart
107 #####
108 ....[Documentation].. RPD Restart
109 ....[Setup]... NONE..
110 ....[Tags]... Negative..
111
112 ....Log...*****Starting ${TEST_NAME}*****
113
114 .....${test}=..Execute Cli Command On Device... device={dh_r0}...command=restart routing
115 .....Run Keyword And Continue On Failure.....check_core_functions_1
116
117 ....Log...*****END of ${TEST_NAME}*****
118
119 #####
120 Tc_chassisd restart
121 #####
122 ....[Documentation].. chassisd Restart
123 ....[Setup]... NONE..
124 ....[Tags]... Negative..
125
126 ....Log...*****Starting ${TEST_NAME}*****
127
128 .....${test}=..Execute Cli Command On Device... device={dh_r0}...command=restart chassisd
129 .....Run Keyword And Continue On Failure.....check_core_functions_1
130
131 ....Log...*****END of ${TEST_NAME}*****
132
133

```

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] Deactivate 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] Deactivate 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] Deactivate 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] Deactivate 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()

```