Automating Device Certifications with Robot Framework

Pratik Lotia
Robot Framework Introduction

- Open source generic test automation framework for acceptance testing
- Keyword driven approach supported with several libraries in Python & Java
- Ideal implementation with high level tests pre-written and network engineers using keywords to develop framework
- Not specifically made for network based testing
- Data driven test cases
- OS and application independent
Robot Framework Major Components

Python Code
- Functions
  - Generic
  - Single task
  - Arguments
  - Return Value

Framework
- Define dependencies & pointers
- Test Cases
  - Keywords
  - Variables

Data Set
- Pointers
- Configuration Files

SSH/NETCONF
Server/VM
Router/Firewall/Switch/Load Balancer
Framework Format

- Extension based
  - HTML
  - TSV – spreadsheet, programmatic
  - Plain text
  - reST (HTML compiled)
Framework Structure

- Modular model
- Structure combines
  - Settings
  - Pre-test setup
  - Test criteria
  - Post-test cleanup
- Each Test has a true/false outcome
- Each Test has 1 or more functions
- Top-down approach for Test Case
  - One fail, all fail model
Framework Sample

```plaintext
user1234@hostname1234:~/Robot-Fw-Testing/NANOG$ cat nanog.robot
*** Settings ***
Documentation This is a Test structure for NANOG74

Library OperatingSystem
Library ${(CURDIR)}/../lib/my_python_code.py
Variables ${(CURDIR)}/../variables/${(TEST_HOST)}.yaml

Suite Setup Open connection
Suite Teardown Close connection

*** Variables ***
${(TEST_HOST)} NANOG-Router

*** TEST Cases ***
Test Case: Fetch interface status
   [Documentation] This should be first step for configuration
   ${output} = some_function1 ${(some_var1)}
   Log to Console ${output}

Test: Load xyz configuration - IPv4
   [Documentation] Loading configs
   ${(output1)} = some_function2 ${(some_var2)}
   ${(output2)} = some_function3 ${(some_var3)}
   some_function4 ${(output1)} ${(output2)}

*** Keywords ***
Open connection
   ${some_result_1} = some_function_4 ${(some_var_4)} ${(some_var_5)} ${(some_var_6)}
   Set Suite Variable ${some_result_1} ${some_result_1}
Close connection
   some_function_x ${(some_var_x)}
```
Robot Command Options

- `robot /path/to/file.robot`
- Options to:
  - Set documentation
  - Set suite, report name
  - Set tags, variables
  - Rerun failed tests
  - Run/exclude certain tests
  - Set logging level, output level
  - Set timestamp
  - Error handling

```
-D --doc
-M --metadata
-G --settag
-t --test name
-i --include tag
-R --rerunfailed
-v --variable
-o --output
-T --timestampoutputs
-L --loglevel
-X --exitonfailure
--dryrun
--quiet
```
## Style Conventions

<table>
<thead>
<tr>
<th>Field</th>
<th>Convention</th>
<th>Example/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Name inside ${}</td>
<td>${mgt_ip_addr}</td>
</tr>
<tr>
<td>Function Name</td>
<td>Keyword(s) with or w/ space</td>
<td>Check Interface Config check_interface_config</td>
</tr>
<tr>
<td>Passing Variables</td>
<td>Leave &gt;4 whitespaces between function &amp; variable</td>
<td>Show Interface  ${mgt_ip_addr}  ${user}  ${passwd}</td>
</tr>
<tr>
<td>Notes</td>
<td>Under [Documentation] in test cases</td>
<td>Test case: check interface config [Documentation] Load and verify IP</td>
</tr>
<tr>
<td>Variables</td>
<td>Define separately yaml file</td>
<td>Helps to keep framework generic and data driven</td>
</tr>
<tr>
<td>Framework</td>
<td>Tabular model</td>
<td>Equal spacing</td>
</tr>
</tbody>
</table>
Creating Test Case

- Whitespaces ignored*
- Keyword (What?)
- Library + Python code
- Arguments:
  - Mandatory
  - Default
- Return Value
- Single Test

```
*** Test Cases ***
Sample

[Documentation] To show functions and arguments
Copy File ${SOME_DIR}/notes.txt   ${ANOTHER_DIR}/merge.txt
Create File ${TEMPDIR}/file1.txt
Create File ${TEMPDIR}/file2.txt   Hello World   ISO-8859-1
${POSITION} = FIND IP ${HTMLCONTENT}
```
Example 1 – Operational Status of Device – Framework

```
user12345@hostname1234:/Robot-Fw-Testing/NANOG/test_cases$ cat mx_1.robot
*** Settings ***
Documentation     This is the Certification test for Juniper MX

Library             OperatingSystem
Library             ${CURDIR}/../lib/nanog_mx_1.py
Variables           ${CURDIR}/../variables/${TEST_HOST}.yaml

*** Variables ***
${TEST_HOST}         MX-MASTER

*** TEST Cases ***
Test Case: Enter Config Mode
[Documentation]     This should be first step for configuration
${connection_en}    =     mx_connect    ${IPADDR}    ${USERNAME}    ${PASSWD}
${output}            =     mx_verify_facts    ${connection_en}
Log to Console      ${output}
```
Example 1 – Variables

```
user12345@hostname1234:/Robot-Fw-Testing/NANOG/variables$ cat MX-MASTER.yaml
IPADDR: 1.2.3.4
IPADDR6: 6600::1
IP6LLADDR: fe80::1
USERNAME: root
PASSWD: @wbty*6
```

*Fake credentials on this and subsequent slides*
Example 1 – Python Code

```python
import sys
import os
import logging
import re
import subprocess
import itertools
from time import sleep
from jnpr.junos import Device
from jnpr.junos import exception
from jnpr.junos.utils.config import Config
from jnpr.junos.utils.start_shell import StartShell

logging.basicConfig(filename='error.log', level=logging.DEBUG)
logger = logging.getLogger("Py_EZ")

def mx_verify_facts(connect):
    return connect.facts

def mx_connect(ip, username, password):
    connect = Device(host=ip, user=username, password=password)
    connect.open()
    return connect
```
Example 1 - Results

```
Srx 1 :: This is the Certification test for Juniper

Test Case: Enter Config Mode :: This should be first step for conf... ...

Srx 1 :: This is the Certification test for Juniper MX  | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed

Output: /home/plotia/Robot-Fw-Testing/NANOG/test_cases/output.xml
Log: /home/plotia/Robot-Fw-Testing/NANOG/test_cases/log.html
```
Results – Executive Summary

*Failed test may be a result of misconfiguration and not a failure of the device*
Results – Detailed Logs

*Failed test may be a result of misconfiguration and not a failure of the device
Suite Setup and Teardown

*** Settings ***
Documentation   This is the Certification test for Juniper MX

Library          OperatingSystem
Library          ${CURDIR}/../lib/load_config.py
Variables        ${CURDIR}/../variables/${TEST_HOST}.yaml

Suite Setup      Open connection to JunOS
Suite Teardown   Close connection

*** Variables ***
${TEST_HOST}      MX-MASTER

*** Keywords ***
Open connection to JunOS
   ${connection_en} = mx_connect ${IPADDR} ${USERNAME} ${PASSWD}
Set Suite Variable
   ${connection_en} = ${connection_en}
Close connection
   disconnect ${connection_en}
Troubleshooting Errors

- Default errors are minimal
- Tedious to look at html for errors
- Logging module in Python
- Similar to print statements
- Prints while running tests

```python
logging.basicConfig(filename='error.log', level=logging.DEBUG)
logger = logging.getLogger("Py_EZ")

def func(var1, var2):
    some_logic_1
    logging.critical(out)
    logging.critical(err)
    some_logic_2
```
Libraries

• Standard Libraries
  ▪ Built-in
    ➢ Run with conditions
    ➢ Evaluation
    ➢ Matching expected behavior
  ▪ Process oriented
    ➢ Control process execution
    ➢ Fetch process attributes
    ➢ Switch process

<table>
<thead>
<tr>
<th>Should Be True</th>
<th>$src &lt; 10</th>
<th>Return code greater than 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Keyword If</td>
<td>$status == 'PASS'</td>
<td>Log</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Passed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Start Process</th>
<th>program alias=example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Process</td>
<td>python -c print 'hello' alias=hello</td>
</tr>
</tbody>
</table>
Libraries

• Standard Libraries
  ▪ DateTime
    ➢ Date and Time conversions
    ➢ Adding time/date
    ➢ Subtracting time/date
  ▪ OS level functions
    ➢ Directory changes/verification
    ➢ File changes/verification (copy, size)
    ➢ Environment variables
    ➢ Merge/List
Libraries

• Standard Libraries
  - String functions
    - Length control/verification
    - Behavior matching
    - Byte conversion
  - Collections
    - Control Lists/Dictionaries
    - Behavior matching
  - Dynamic input, Telnet
  - Screenshots
Libraries

- Extended Libraries
  - Selenium, Selenium with Angular JS
  - Suds (SOAP), MQTT, Faker
  - SSH, Nc client, Django, FTP
  - Database, HTTP, Archive

```
Extract Tar File  [file, dest=None]

Start Django and open Browser
Start Django
Open Browser ${SERVER} ${BROWSER}

Create Session httpbin http://httpbin.org
\(\{data\}\) Create Dictionary name-bulkan surname-evcimen
\(\{headers\}\) Create Dictionary Content-Type=application/x-www-form-urlencoded

Execute SQL CREATE TABLE DemoTable (ID INT NOT NULL, Name VARCHAR(255))
Execute SQL ALTER TABLE DemoTable ADD PRIMARY KEY (Id);
```

```python
${words} = FakerLibrary.Words
Log words: ${words}
```
Tagging

- Classifying test cases & providing metadata
- Report shows statistics based on tags
- Include/Exclude execution of specific tags
- Tags for critical, non-critical, trivial
- Types
  - Force tags
  - Default tags
  - Customized tags

```plaintext
*** Settings ***
Force Tags  nanog-74
Default Tags  user1  security

*** Variables ***
${HOST}  1.2.3.4

*** Test Cases ***
No own tags
[Documentation]  This test has tags nanog-74, user1 and security
No Operation

With own tags
[Documentation]  This test has tags nanog-74, canada and network
[Tags]  canada  network
No Operation

Own tags with variables
[Documentation]  This test has tags nanog-74 and host-1.2.3.4
[Tags]  host-$(HOST)
No Operation
```
Editor

• RIDE – Standalone editor
• Plugins for various editors
  – Eclipse
  – Sublime
  – Vim
  – Emacs
  – Gedit
  – Notepad++

*https://www.youtube.com/watch?feature=player_embedded&v=6F_xGKdoN1E
Example 2 - Framework

Test: Load SNMP Poll v2c - IPv6

[Documentation] SNMP Server Configuration

${SNMP_VERSION}  Set Variable  1
${output}   snmp_walk  ${SNMP_IPV6}  ${WRONG_COMMUNITY}  ${SNMP_VERSION}
${TIMEOUTRESPONSE}  Set Variable  Timeout
Should Contain  ${output}  ${TIMEOUTRESPONSE}
${SNMP_VERSION}  Set Variable  2c
${output}   snmp_walk  ${SNMP_IPV6}  ${WRONG_COMMUNITY}  ${SNMP_VERSION}
${TIMEOUTRESPONSE}  Set Variable  Timeout: No Response
Should Contain  ${output}  ${TIMEOUTRESPONSE}
File Should Exist  ${SNMP_POLL_V2C_IPV6}
${status} =  load_setfile_and_execute  ${connection_en}  ${SNMP_POLL_V2C_IPV6}
Should be Equal  ${status}  ${NONE}

Test: Verify SNMP Poll v2c - IPv6

[Documentation] snmpwalk should receive valid output

${SNMP_VERSION}  Set Variable  2c
${output}   snmp_walk  ${SNMP_IPV6}  ${RIGHT_COMMUNITY}  ${SNMP_VERSION}
${TIMEOUTRESPONSE}  Set Variable  Timeout: No Response
Should Not Contain  ${output}  ${TIMEOUTRESPONSE}
Should Contain  ${output}  Juniper${SPACE}Networks
##Rollback v4 syslog config##
${status} =  rollback  ${connection_en}  1
Should be Equal  ${status}  ${TRUE}
Example 2 - Variables

```
IPADDR: 1.2.3.4
IPADDR6: 6600::1
IP6LLADDR: fe80::1
USERNAME: root
PASSWD: @wbty* &
SNMP_IPV4: 1.2.3.4
SNMP_IPV6: 6600::1
WRONG_COMMUNITY: wrong
RIGHT_COMMUNITY: # (dh12v4
SNMP_POLL_V2C_IPV4: ../resources/snmp_poll_v2c_ipv4.txt
SNMP_POLL_V2C_IPV6: ../resources/snmp_poll_v2c_ipv6.txt
```
def snmp_walk(ipaddr, community, version):
    #logging.critical('first')
    if '-u' not in ipaddr:
        call = subprocess.Popen(['snmpwalk', '-v', str(version), '-c', str(community), str(ipaddr)],
                                stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    else:
        command = 'snmpwalk '+ipaddr
        call = subprocess.Popen(command.split(), stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    output, error = call.communicate()
    if error:
        return(error)
    else:
        return(output)

def load_setfile_and_execute(connect, filename):
    conf = Config(connect)
    with open(filename, 'r') as fh:
        for i in fh:
            conf.load(i, format='set')
    conf.commit()

def rollback(connect, num):
    conf = Config(connect)
    conf.rollback(rb_id=int(num))
    conf.commit()
    return True
Loops

- ‘For’ Loop
- Repetitive tasks
- Keyword/Variable

```plaintext
*** Test Cases ***
Example 1
:FOR ${attendee} IN nanog73 ${NANOG74}
 \ Log ${attendee}
 \ Log ${company}
Log Outside loop
```
Additional Tools

• Rebot
  ▪ Process XML output
  ▪ Generate html reports
  ▪ Combine or Merge reports

• Libdoc
  ▪ Generate Documentation

• Tidy
  ▪ Cleanup / Change format

• DbBot
  ▪ Reports to SQLite
  ▪ Unify storage of reports

```
rebot output.xml
rebot output1.xml output2.xml
rebot --merge --name Sample --critical regression original.xml merged.xml
rebot --rerunfailed output1.xml --output rerun.xml tests
rebot --merge original.xml rerun.xml
```

```
python -m robot.libdoc test/resource.html doc/doc_resource_doc.html
```

```
python -m robot.tidy [options] infile [outfile]
```

```
python -m ddbot.run atest/testdata/one_suite/output.xml
```
Additional Tools

• Robot Corder
  ▪ Record GUI actions
  ▪ HTML framework generation
• Pabot
  ▪ Parallel execution
  ▪ Time
• Fixml
  ▪ Fixing incomplete xml results
• Mabot
  ▪ Manual tests with compatible outputs
API

• Running code via code!
• API functions to run tests along with options
• Includes all basic tools such as rebot, libdoc, tidy
• Retrieve results
• Customize reports HTML/XML format
• Use standard libraries with API
• Specify variables and resources
Example 3 - Framework

Test: Load AAA-IPV4 Configuration

[Documentation] If file present, load into running-config
File Should Exist ${AAA_IPv4_CONF}
${status} = load_file_and_execute ${connection_en} ${AAA_IPv4_CONF}
Should be Equal ${status} ${TRUE}

Test: Verify AAA-IPV4 (test1 account)

[Documentation] Ensure `test1` account does not have configuration privilege
${connection_test} = router_connect ${DRIVER} ${IPADDR} ${TESTUSER} ${TESTPASS}
${enable_status} = check_enable ${connection_test}
Should be Equal ${enable_status} ${FALSE}
${output} = router_show ${connection_test} ${VERSION}
${status} = must_have_keywords_in ${VERSION_WORDS} ${output}
Should be Equal ${status} ${TRUE}
Example 3 - Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPADD:</td>
<td>192.168.0.2</td>
</tr>
<tr>
<td>IPADD:6</td>
<td>6600::1</td>
</tr>
<tr>
<td>USERNAME:</td>
<td>admin</td>
</tr>
<tr>
<td>PASSWD:</td>
<td>(dsajng#8)</td>
</tr>
<tr>
<td>DRIVER:</td>
<td>router_xx</td>
</tr>
<tr>
<td>TESTUSER:</td>
<td>test1</td>
</tr>
<tr>
<td>TESTPASS:</td>
<td>test1</td>
</tr>
<tr>
<td>AAA_IPV4_CONF</td>
<td>../resources/aaa_ipv4_conf.txt</td>
</tr>
<tr>
<td>VERSION:</td>
<td>version</td>
</tr>
<tr>
<td>VERSION_WORDS:</td>
<td>../resources/version_words.txt</td>
</tr>
</tbody>
</table>
Example 3 – Python Code

```python
def load_file_and_execute(connect, file_name):
    status = connect.send_config_set(open(file_name).readlines())
    if "ERROR" in status:
        return False
    return True

def router_connect(device_type, ip, username, password):
    connect = ConnectHandler(device_type=device_type, ip=ip,
                             username=username, password=password, secret='')
    connect.enable()
    return connect

def check_enable(connect):
    connect.enable()
    return (connect.check_enable_mode())

def must_have_keywords_in(file_name, output):
    keywords = open(file_name).read().split("","
    flag = 1
    for key in keywords:
        if key not in output:
            flag = 0
            break
    if flag == 0:
        return False
    return True

def router_show(connect, call):
    show_result = connect.send_command("show "+call)
    return show_result
```
Dos and Don’ts

• Dos
  ▪ Documentation (options)
  ▪ Short & easy naming
  ▪ What, not how

Test: Load Remote Syslog - IPv4
[Documentation] Config syslog client

- Tabular uniformity

Test: Verify Remote Syslog - IPv4
[Documentation] Verifying on this host acting as Syslog server
$\{status\} = if\_nonzero\_file\_exists /var/log $\{IPADDR\}.log
Should be Equal $\{status\} $\{TRUE\} $\{status\} = rollback $\{connection\_en\} 1
Should be Equal $\{status\} $\{TRUE\}

▪ Generic and simple framework
Dos and Don’ts

• Dos
  ▪ Logic in code
  ▪ Data driven
  ▪ Checks
  ▪ Syntax (Given, When, Then)

• Don’ts
  ▪ Dependencies
  ▪ Granular test
  ▪ Hardcoded variables
  ▪ Sleeping in place of polling

<table>
<thead>
<tr>
<th>Test: Copy running config</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Documentation] Copy to local directory</td>
</tr>
<tr>
<td>Copy Running Config</td>
</tr>
<tr>
<td>Sleep</td>
</tr>
<tr>
<td>Wait Till Creation</td>
</tr>
</tbody>
</table>
Test: Verify zero packet loss & then Load SYN Flood config IPv4

[Documentation] If file present, load into running-config

```plaintext
${output}
   hping_flood  count=500  ip=192.168.0.2  port=22
${loss}
   hping_packet_loss  ${output}

Should Be Equal as Numbers
   ${loss}  0
```

File Should Exist
   ${SYN_FLOOD_4_CONF}

```plaintext
${status} =
   load_file_and_execute  ${connection_en}  ${SYN_FLOOD_4_CONF}

Should be Equal
   ${status}  ${TRUE}
```

Test: Verify SYN Flood IPv4

[Documentation] Check packet loss

```plaintext
${output}
   hping_flood  count=500  ip=192.168.0.2  port=22
${loss}
   hping_packet_loss  ${output}

${status} =
   should_be_greater_than
   ${loss}  50

Should be Equal
   ${status}  ${TRUE}
```
Example 4 - Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPADDR</td>
<td>192.168.0.2</td>
</tr>
<tr>
<td>IPADDR6</td>
<td>6600:1</td>
</tr>
<tr>
<td>USERNAME</td>
<td>admin</td>
</tr>
<tr>
<td>PASSWD</td>
<td>(^73fhjdl)</td>
</tr>
<tr>
<td>DRIVER</td>
<td>cisco_asa</td>
</tr>
<tr>
<td>SYN_FLOOD_4_CONF</td>
<td>../resources/syn_flood_4_conf.txt</td>
</tr>
</tbody>
</table>
Example 4 – Python Code

```python
def hping_flood(count, ip, port):
    call = subprocess.Popen(['hping3', '-i', '1', '-S', '-p', port, '-c', count, str(port)], stdout=subprocess.PIPE)
    output, error = call.communicate()
    return(output)

def hping_packet_loss(output):
    for line in output:
        if "loss" in line:
            my_list = line.split(',')
            end_pos = my_list[2].find('%')
            loss = my_list[1:end_pos]
            return(int(loss))

def load_file_and_execute(connect, file_name):
    status = connect.send_config_set(open(file_name).readlines())
    if "ERROR" in status:
        return False
    return True

def should_be_greater_than(loss, number):
    if int(loss) > int(num):
        return True
    return False
```
Summary

• Robot Automation Framework provides several use case scenarios for network automation
• Keyword based acceptance driven tests
• Reuse generic test libraries
• Separation of components allow customization and ease of understanding
• Simplify automation of workflows
Resources

- [https://github.com/robotframework/robotframework/blob/master/INSTALL.rst](https://github.com/robotframework/robotframework/blob/master/INSTALL.rst)
- [http://www.slideshare.net/pekkaklarck/robot-framework-introduction](http://www.slideshare.net/pekkaklarck/robot-framework-introduction)
- [http://robotframework.org/robotframework/#user-guide](http://robotframework.org/robotframework/#user-guide)
- [https://github.com/robotframework/HowToWriteGoodTestCases/blob/master/HowToWriteGoodTestCases.rst](https://github.com/robotframework/HowToWriteGoodTestCases/blob/master/HowToWriteGoodTestCases.rst)
- [http://robotframework.org/robotframework/#standard-libraries](http://robotframework.org/robotframework/#standard-libraries)
Thank You

- Questions?
Backup slides