TC Generator: USER GUIDE

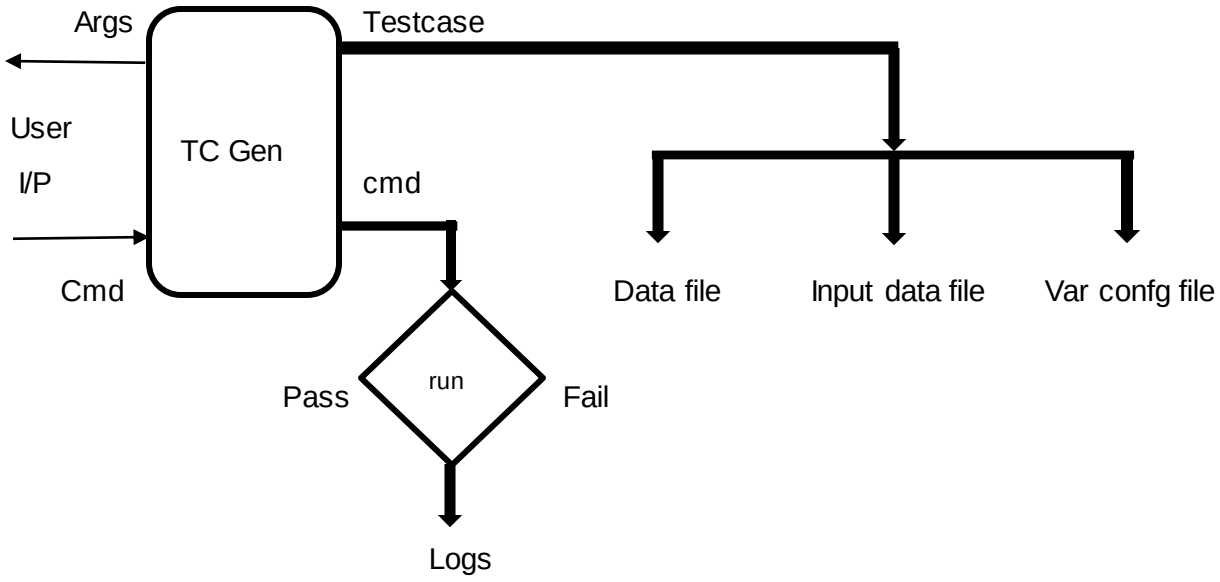
TABLE OF CONTENTS

1.0 WHAT IS TC GENERATOR?

TC Generator is Warrior tool. It has the capability to create a testcase with warrior structure or without warrior structure according to the user requirements. Finally it will give the location of created xml files and executable command of the created testcase(This is explained in more detail in section 4.0 and section 5.0).

2.0 HOW DOES TC GENERATOR WORK:

TC Generator diagrammatically represented by this way.



(section 2.0 figure 1)

To write a testcase with TC Generator user have installed warriorframework, python3.6 or above version in a Linux machine.

The TC Gen mode in the diagram has been drawn as rectangle which will ask to user arguments as command from user and creates Testcase and dependent files shown above. Along with files TC Generator will gives you the executable command of the Testcase.

To know type of Testcases support TC Generator use below command:

./Warrior used for git clone way. And **Warrior** used for pip installed way.

```
Warrior –tc_gen - -help
```

The above command result shown below:

(section 2.0 figure 2)

3.0 PREREQUISITES

TC Generator requires the following prerequisites to run successfully:

1. A Linux system
2. Python – version 3.6 or above
3. Warriorframework_py3

Warrior by default would install all dependencies of TC Generator.

4.0 INSTALLATION:

TC Generator comes along with Warriorframework_py3. Warriorframework_py3 can be installed by two ways:

Steps to install **Warriorframework_py3**:

GIT MODE:

- Do git clone warriorframework in **command line** with command provided below:

```
git clone https://github.com/warriorframework/warriorframework_py3.git
```

PIP MODE:

- Python3 PIP should be installed in your machine.
- Do pip install warriorframework in **command line** with command provided below:

```
pip3 install warriorframework
```

Advantages:

- Installing the warriorframework through **PIP MODE** warrior will set everything for user. User can access across the machine by providing **Warrior** at the starting of the command.

5.0 USAGE OF TC GENERATOR:

The TC Generator supports four types of testcases are shown below.

- CLI
- SNMP
- NETCONF
- REST

Usage of this four types of testcases are shown below.

For general way of usage will use **./Warrior** and pip installable way will use **Warrior.** For every testcase **type** and **tc_name** is mandatory argument.

5.1 HOW TO RUN TC GENERATOR:

CLI Usage:

```
./Warrior -tc_gen - -type="cli" - -tc_name="cli_demo_with_dip.xml" - -ip="localhost" - -username="uname" - -password="pwd" - –cli_port="1234" - –dip_port="1234"

./Warrior -tc_gen - -type="cli" - -tc_name="cli_demo_with_dip.xml" --ip="localhost" - -username="uname" - -password="pwd" - -cli_port="1234" –dip_port="1234" -wrs

./Warrior -tc_gen --type="cli" --tc_name="cli_demo_without_dip.xml" --ip="localhost" --username="uname" --password="pwd" -cp="23009"

./Warrior -tc_gen --type="cli" --tc_name="cli_demo_without_dip.xml" --ip="localhost" --username="uname" --password="pwd" -cp="23009" –wrs
```

The command argument full names and usage as shown in (section 2.0 figure 2)

SNMP Usage:

```
./Warrior -tc_gen --type='snmp' --tc_name='tc_name.xml' --username='username' --password='pwd' --ip='localhost' -cp='1234' -sp='1234'

./Warrior -tc_gen --type='snmp' --tc_name='tc_name.xml' --username='username' --password='pwd' --ip='localhost' -cp='1234' -sp='1234' –wrs
```

The command argument full names and usage as shown in (section 2.0 figure 2)

NETCONF Usage:

```
./Warrior -tc_gen --type='netconf' --tc_name='tc_name.xml' --username=uname' --password='pwd' --ip='localhost' -np=1234'

./Warrior -tc_gen --type='netconf' --tc_name='tc_name.xml' --username='uname' --password='pwd' --ip='localhost' -np='23013' –wrs
```

The command argument full names and usage as shown in (section 2.0 figure 2)

REST Usage:

```
./Warrior -tc_gen --type='rest' --tc_name='tc_name.xml' --username ='uname' --password ='pwd'

./Warrior -tc_gen --type='rest' --tc_name='tc_name.xml' --username ='uname' and --password ='pwd' -wrs

./Warrior -tc_gen --type='rest' --tc_name='tc_name.xml'

./Warrior -tc_gen --type='rest' --tc_name='tc_name.xml' –wrs
```

The command argument full names and usage as shown in (section 2.0 figure 2)

5.2 DIRECTORY STRUCTURE:

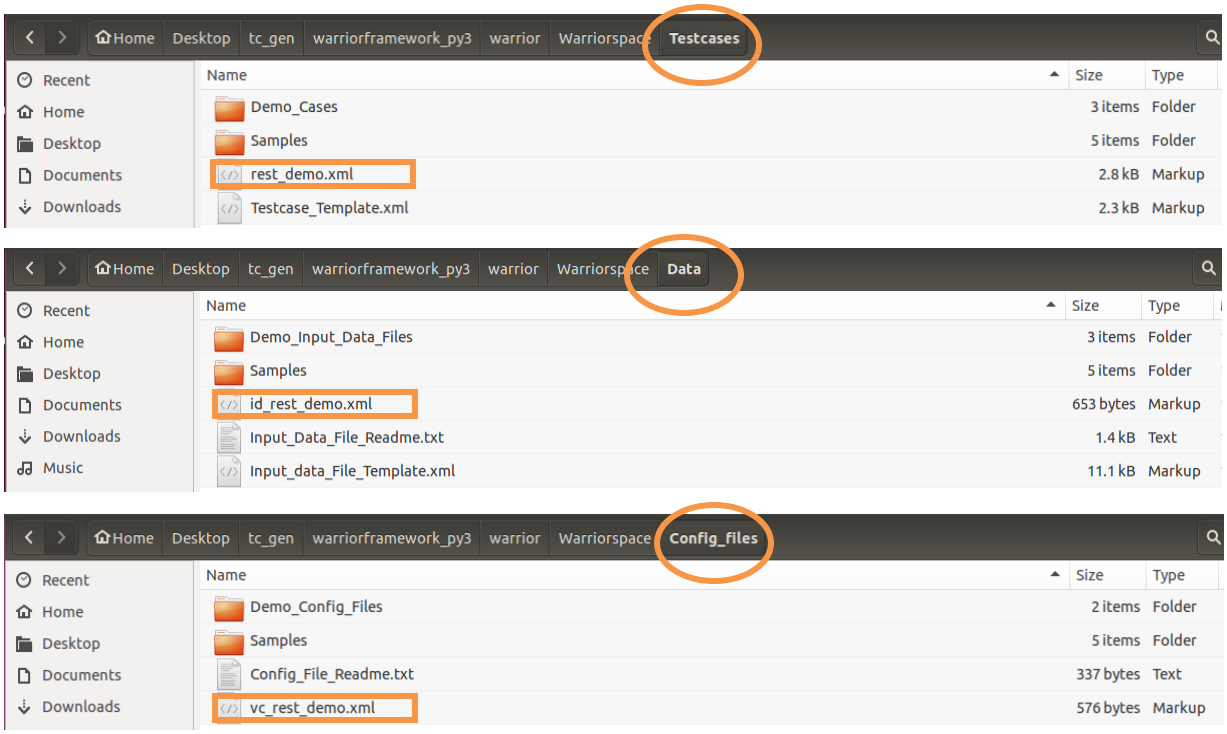
TC Generator provides the warrior structure when user provided **- - with_repo_structure** as a command line argument Shown in below figure.

If user did not provided **- - with_repo_structure** argument in command line TC Generator create files in current working directory.

Without **- - with_repo_structure :**

(Figure 5.2.1)

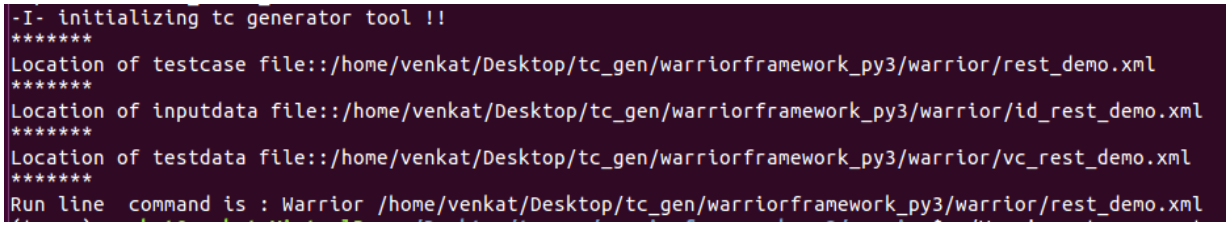
With **- - with_repo_structure :**



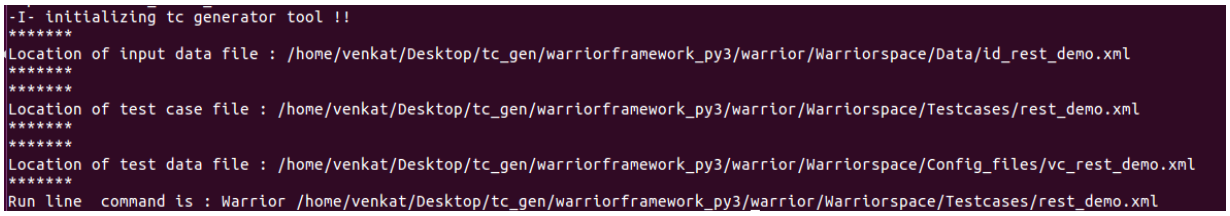
(Figure 5.2.2)

Results :

1. Without - - with_repo_structure :



2. With - - with_repo_structure :

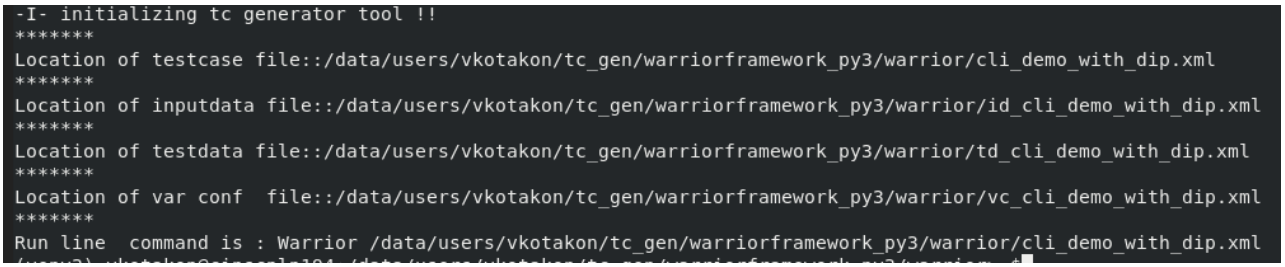


(Figure 5.2.3)

6.0 FILES

6.1 GENERATED XML FILES:

Files are generated each time when TC Generator is run. Log files are not generated when the TC Generator is used.



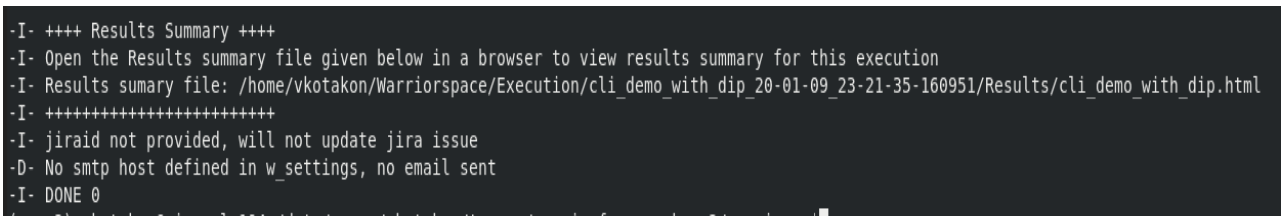
Generator files are shown in above figure.

6.2 LOG FILES:

As sown above figure **run line command**. After Run of run line command log files will be generated.

```
./Warrior /data/users/tc_gen/warriorframework_py3/warrior/cli_demo_with_dip.xml
```

Logfiles shown as below.



After running the command at last will have result summery as show in above figure.

Checkout directory till shown below:

/home/User/Warriorspace/Execution/cli_demo_with_dip_20_01_09_23_21_35_160951

Inside of this location you will find **Logs directory.**