Testkit-Lite User Guide

1. Introduction

This guide presents an overview of Testkit-lite, and describes the installation, deployment, results, and options list of Testkit-lite. Also a quick-start is included.

2. Overview

Testkit-Lite is a light-weight test tool with command-line interface. It provides following functions:

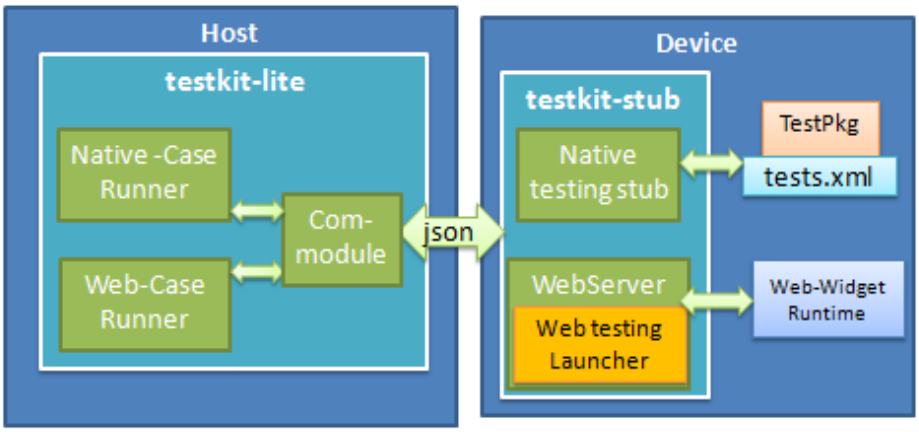
- Accepts test definition XML files as input.
- Drives automatic test execution and collects test result
- Provides multiple options to support test purpose, such as filters, external test launcher
- Supports test on cross-platforms, such as : TIZEN, android, Linux Distro like Ubuntu/Fedora

3. Deployment

Testkit-Lite is composed of 2 components testkit-lite and testkit-stub. The 2 components can be deployed on the same target machine, or deployed on target device and host machine

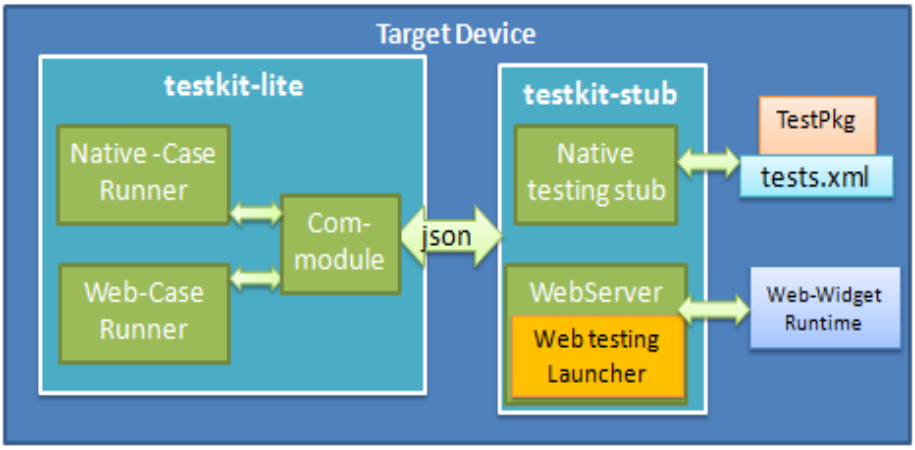
- Distributed deployment.

  In this deployment, the component testkit-lite is deployed on host machine, and testkit-stub is deployed on target device.



- Standalone deployment

  In this deployment, both testkit-lite and testkit-stub are deployed on target device.



4. Dependency

- Python 2.7 or above
- Python library: python-requests

- **Install request with pip (Ubuntu OS)**

  `$ sudo pip install requests`

- **Install request package from source code (TIZEN)**

  Download source from *https://github.com/kennethreitz/requests/archive/master.zip*

  `$ unzip master.zip`

  `$ python setup.py install`

5. Installation

To install testkit-lite

- **Download testkit-lite released package**

  For Ubuntu 12.04/Ubuntu 12.10, download the debian package “testkit-lite_<version>.deb”.

  For Tizen/Fedora, download the RPM package “testkit-lite_<version>.rpm”.

- **Install on Host(Ubuntu 12.04/Ubuntu 12.10/ Ubuntu 12.04 server/ Ubuntu 12.10 server):**

  Navigate to the work directory testkit-lite:

  Double click testkit-lite_<version>.deb file to install testkit-lite or execute command as below

  `$ sudo dpkg -i testkit-lite_<version>.deb`

- **Install on target (TIZEN):**

  Navigate to the work directory testkit-lite:

  `$ sudo rpm -ivh testkit-lite_<version>.rpm`

- **Download testkit-stub released binary.**

  The binary for ARM and IA architecture are both available.

- **Deploy testkit-stub binary.**

Copy the binary to the folder “/opt/home/developer/”, grant the execution permission.

```
$ sudo chmod +x /opt/home/developer/testkit-stub
```

Notice: Deploy Testkit-Lite on Ubuntu server edition, we still need add 2 extra environment variables as below:

```
export DBUS_SESSION_BUS_ADDRESS="unix:path=/run/dbus/system_bus_socket"
export DISPLAY=":0.0"
```

6. Quick Start

- **Distributed deployment**

- Run webapi test cases on TIZEN device with crosswalk runtime or WRT

```
$ testkit-lite -f "<somewhere>/<test_definition_webapi_file_xwalk>.xml"
```

- Run webapi test cases on android device with crosswalk runtime

```
$ testkit-lite -f "<somewhere>/<test_definition_webapi_file_xwalk>.xml" --comm androidmobile
```

- **Standalone deployment**

- Run core test cases:

```
$ testkit-lite -f "<somewhere>/<test_definition_core_file>.xml" --comm localhost
```

- Run webapi test cases with wrt-launcher

```
$ testkit-lite -f "<somewhere>/<test_definition_webapi_file>.xml" -e “wrt-launcher -s <wgt_id>” --comm localhost
```

- Run webapi test cases with crosswalk

```
$ testkit-lite -f "<somewhere>/<test_definition_webapi_file>.xml" -e “xwalk --allow-file-access-from-files <xwalk_webapp_id>” --comm localhost
```

- Run webapi test cases with browser, take chrome browser as instance:

```
$ testkit-lite -e "google-chrome --allow-file-access-from-files --disable-web-security --start-maximized --user-data-dir=<somewhere>/data <somewhere>/webrunner/index.html"
-f "/< somewhere>/<test_definition_file>.xml" --comm localhost
```

Notice: value of **“--user-data-dir”** should be a folder writeable for normal user.

7. Usage of Testkit-Lite

7.1 Options for basic function

- **Mandatory option**

Option	Description
-f [prefix:]"<test_definition_file_path>"	Specify one or more test cases definition file as input.

- Input a single test definition file:

  $ testkit-lite -f "<somewhere>/<test_definition_file_1>.xml"

- Input multi test suites (we use white-space as separator):

  $ testkit-lite -f " <somewhere>/<test_definition_file_1>.xml <somewhere>/<test_definition_file_2>.xml <somewhere>/<test_definition_file_3>.xml"

- Use prefix to extend the location of test suites in host-device mode,

  We can use the prefix **"device:"** to specify that test definition files are located in device:

  $ testkit-lite -f device:" <somewhere>/<test_definition_file_1>.xml <somewhere>/<test_definition_file_2>.xml <somewhere>/<test_definition_file_3>.xml"

  ■ **Optional options**

Option	Description
-e <external_test_instance>	Specify an external test instance; it will be launched during test. For webapi test, the "-e" usually provided with a web app launcher command-line.
--comm <comm_type>	Specify commodule type, "tizenmobile" is default value. So for tizen test, it can be omitted; "androidmobile" is for android device access "localhost" is for localhost access.
--deviceid <device_id_string>	Specify a device with its id when more than one device connected. If this option is omitted, lite will use the first recognized device by Host. For TIZEN, it uses "sdb devices" for full list. For Android, it uses "adb devices" for full list
--testprefix <prefix_for_location>	Set a prefix value for test case location. such as "http://127.0.0.1:8000"
--version	Show version information
-o <test_result_file>	Specify an customized output location of result file. Testkit-Lite output result file with location **/opt/testkit/lite/latest/tests.result.xml** by default.
--non-active	Disable the ability to set the result of core manual cases from the console

- Show the help information:
  $ testkit-lite --help/-h

- Show the version of lite:
  $ testkit-lite --version

- **“--comm”** option is provided to end-user to specify commutation type used for test

  - Run test with TIZEN mobile device, the communication over “sdb”
  $ testkit-lite -f deivce :”<somewhere>/<test_definition_webapi_file>.xml” --comm tizenmobile
  It is default communication type, so usually, “--comm” option can be omitted.
  $ testkit-lite -f device :”<somewhere>/<test_definition_webapi_file>.xml”

  - Run test with TIZEN IVI device, the communication over “ssh”
  $ testkit-lite -f device:"<somewhere>/<test_definition_file>.xml” --comm tizenivi

  - Run test with android mobile device, the communication over “adb”
  $ testkit-lite -f device:"<somewhere>/<test_definition_file>.xml” --comm androidmobile

  - Run test with localhost device, the communication is local access (file location, shell invoking)
  Notice: in localhost mode, the prefix **“device:”** is illegal for test definition location string
  $ testkit-lite -f ”<somewhere>/<test_definition_webapi_file>.xml” --comm localhost

- **“-e”** option is provided to end-user to customize an external test instance launcher for test,
  It will override the test launcher already defined in test definition file (tests.xml)

  - Run test with TIZEN mobile device with “crosswalk”

  $ testkit-lite -f deivce :”<somewhere>/<test_definition_webapi_file>.xml” -e “xwalk --allow-file-access-from-files aabbccddeeffgghh”

  - Run test with android mobile device with “crosswalk”
  $ testkit-lite -f deivce :”<somewhere>/<test_definition_webapi_file>.xml” -e “am start -n org.xwalk.test_service_tests/.test_service_testsActivity” –comm androidmobile

  - Run test with chrome browser in localhost
  $ testkit-lite -f "/< somewhere>/<test_definition_webapi_file>.xml" -e "google-chrome --allow-file-access-from-files --disable-web-security --start-maximized --user-data-dir=<somewhere>/data <somewhere>/webrunner/index.html"--comm localhost
  Notice: value of **“--user-data-dir”** should be a folder writeable for normal user.

- **“--testprefix”** option is very useful for end-user to add a prefix for each **“test_script_entry”** already defined in test definition file.

  - Specify a prefix to make test runner access test script from remote server.

  For instance, if we want to access test scripts from http://127.0.0.1:8080, you can use

  $ testkit-lite -f deivce :”<somewhere>/<test_definition_webapi_file>.xml”

  --testprefix http://127.0.0.1:8080

7.2 Options for filtering function

Testkit-lite provides several useful filter conditions to support execute a subset in test definition files. When use combined conditions, the conditions perform “AND” logic together.

Filter	Description
-A	A shortcut of filter auto test cases. It equals filter “execution_type = auto”
-M	A shortcut of filter manual test cases. It equals filter “execution_type = manual”
--type	Filter test cases by test case type: • functional_positive • functional_negative • security • performance • reliability • portability • maintainability • compliance • user_experience
--priority	Filter test cases by test case priority: • P0 • P1 • P2
--status	Filter test case by test case status: • ready • approved • designed

--set	Filter test case by test set
--id	Filter test case by test case id
--component	Filter test case by test case component
--capability	Specify hardware capability file to filter test case

- Assign multiple values to each filter.
  To select test cases of both P0 and P1 priority, run the following command:

$ testkit-lite --priority P0 P1 -f device:" <somewhere>/<test_definition_file>.xml"

- Use a group of filters at one time, because Testkit-Lite performs the AND logic.
  For example, to select test cases by both priority and component filters, run the following command:

$ testkit-lite --priority P0 P1 --component comp1 comp2
-f device:"<somewhere>/<test_definition_file>.xml"

- Freely combine multi-filters in one command to meet complex requirement.
  For example, to select and run test cases from two test descriptor files by filters of priority, component and status:

$testkit-lite --priority P0 P1 --component comp1 comp2 --status ready
-f device:" <somewhere>/<test_ definition_file_1>.xml <somewhere>/<test_definition_file_2>.xml"

- Especially, Testkit-lite provides an option "--capability" to support filtering TCs based on an xml file which contains capabilities information of a device. Please refer to Chapter 8 for details.

$testkit-lite --capability "<somewhere>/capability.xml"
-f device:" <somewhere>/<test_ definition_file_1>.xml <somewhere>/<test_definition_file_2>.xml"

8. Capability Filter

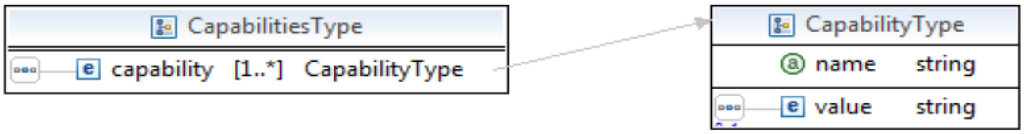
8.1 Use Capability File With Lite

Some test sets depend on a hardware capability or several hardware capabilities of a device.
When launch a test cycle includes these test sets, Testkit-Lite will decide to execute or skip a Test set of based on actual hardware capabilities of device under test:

- If any one of hardware capabilities required not met, the test set will be skipped.
  In the result file, the set's numbers (all/pass/fail/block) will all be recorded as zero.
- If all the hardware capabilities required are met, the test set will be executed.

Testkit-Lite accepts a XML file which contains capabilities definition of a device with "--capability" option.
Any instance of the capability XML file should follow schema definition as below.

Structure:



One or more sub-element “capability” is contained. The sub-element is a simple string element, only 1 identifier attribute “name” and a pure text sub-element is contained.

Limitation:

The attribute “name” is the identifier of sub-element “capability”. Testkit-lite will filter “set” with the capability list announced in this element and the capability list extracted from devices.

8.2 Generate Capability File With GetCap

For TIZEN platform, we implement a widget “getCap” to generate capability file as description in **8.1.** Widget “getCap” invokes ***tizen.systeminfo*** APIs to obtain the hardware capabilities of a TIZEN device and invokes ***tizen.filesystem*** APIs to write these capabilities information in an xml file by location *“/opt/usr/media/Documents/tct/capability.xml” in* device.

- When you launch test with web_tct tool, tct-mgr or tct-shell will launch “getCap” and get capability file automatically, so you don’t need to touch capability file directly.
- When you launch test with testkit-Lite directly, you need to launch “getCap” and get capability file manually. Please follow the steps as below:

1. Install getCap widget in device.
   sdb push /path/to/getCap.wgt /opt/usr/media/tct/
   sdb shell wrt-installer -i /opt/usr/media/tct/getCap.wgt
2. Launch getCap by clicking the app icon and wait till it exit.
3. Pull the capability file from device with sdb command.
   sdb pull /opt/usr/media/Documents/tct/capability.xml <somewhere>/capability.xml

Notice: “getCap” widget install package usually released in the web_tct tar ball.

9. Checking Test Reports

After Testkit-Lite completes executing all test cases successfully, you can obtain an .xml test report from /opt/testkit/lite/latest

10. Viewing Test Reports

Test report can be viewed in HTML format, so the data in the xml result file looks more human friendly. Please follow the following steps to view test report:

- copy files: application.js, back_top.png, jquery.min.js, testresult.xsl, tests.css under directory /opt/testkit/lite/xsd/
- put the files from step 1) under the same directory as the xml result file
  open xml result file with a web browser (IE, Chrome or Firefox)