Testkit-Lite Quick Start

Contents

1. Introduction

Testkit-Lite is a test runner, on which you can automatically run test cases for either middleware components or WebAPIs. This document describes how to write and structure a test case to be run on Testkit-Lite.

2. Test Case Descriptor File

You need to write an .xml test case descriptor file for Testkit-Lite to learn which test cases to run. Below is an example of test case descriptor file for the test case named "HelloWorldTest".

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name=" HelloWorldSuite">
  <set name="HelloWorldSet">
   <testcase component=”HelloWorldComponent" execution_type="auto"
purpose=”Test HelloWorld” id="HelloWorldTest">
    <description>
     <test_script_entry test_script_expected_result="0"
timeout="90">/PATH/TO/HelloWorld
      </test_script_entry>
    </description>
  </testcase>
 </set>
</suite>
```

- **<suite>** and **<set>**

Both **<suite>** and **<set>** function as container element in a test case descriptor file. **<suite>** is the root element of a test case descriptor file, which is parent of one or more **<set>** elements. A **<set>** element in turn contains one or more **<testcase>** child elements.

- **<testcase>**

The **<testcase>** element describes the key information of a test case, including ID, targeting component, test purpose, pre-conditions, test steps, and expected results. For an auto test case, you need to assign **auto** to **execution_type**, and provide at least one <**test_script_entry>** element to tell which commands to run and with what arguments. Be aware that you need to provide the absolute path of test scripts, programs, or HTML web test page, if it is not accessible with the system default **PATH** environment variable. Testkit-Lite verdicts test result to PASS or FAILURE by comparing the real result with value of the attribute **test_script_expected_result.**

- **<test_script_entry>**

<test_script_entry> supports two types of commands:

- Executable programs or scripts developed in programming or scripting languages, such as Python, C/C++, and Java.
- HTML web test page. It can either embed JavaSript test code in page content or link to separated files that contain the JavaScript test code.

A test case descriptor file can contain as many test cases as you want and organize them in test sets, allowing batch execution of test cases in one test cycle. You need to update the test case descriptor file when new test cases are ready.

3. Testkit-Lite Constraints

Testkit-Lite has to make assumptions on test cases to reduce the complexity of running them. Constraints are as follows:

- Testkit-Lite uses system built-in normal account, which is a non-root account, to run test executable files by default.
- The test script, program, or HTML web test page in <**test_script_entry>** should be accessible with the system default PATH environment variable. If not, you need to provide full path to make it accessible by Testkit-Lite.
- Testkit-Lite requires a **Web-Runtime Environment** to load web test pages for running WebAPI test cases. A utility tool, **WRTLauncher,** is available for launching and running WebAPI test cases easily.

  Note: WRTLauncher takes the name of widget achieve as input. To launch a WebAPI test case, run the following command:

```
$ WRTLauncher  <widget_name>
```

4. Running Test Case

To run a test case, perform the following steps:

1. Check that Testkit-Lite has already been installed on the target test device. On terminal, run the following command:

```
$ testkit-lite   --help
```

The Testkit-Lite help information displays, as shown in Figure 3-1.

```
Usage: testkit-lite [options]
examples: testkit-lite -E <tizen|meego> -f <somewhere1>/tests.xml <somewhere2>/tests.xml
          testkit-lite -E <tizen|meego> --testxmlconfig <somewhere1>/testsxmlconfig <somewhere2>/testsxmlconfig
          testkit-lite -E <tizen|meego> -f tests.xml -V
          testkit-lite -E <tizen|meego> -f tests.xml -D
          testkit-lite -E <tizen|meego> -f tests.xml -A
          testkit-lite -E <tizen|meego> -f tests.xml -M
          testkit-lite -E <tizen|meego> -f tests.xml -S
          testkit-lite -E <tizen|meego> -f tests.xml -C
          testkit-lite -E <tizen|meego> -f tests.xml --level level1 level2 --type type1 type2 ...
          testkit-lite -E <tizen|meego> -f tests.xml -f tests1.xml tests2.xml --testxmlconfig config1 config2 -D -A -S -C --level level1 level2 --type type1 type2 ...
          testkit-lite -E <tizen|meego> -f tests.xml -w --webtestid=1 --webhidestatus --priority P0 ...
Note:
        1) TestLog is stored to /opt/testkit/lite/latest
        2) testkit-lite enables only automatic tests by default

Options:
  -f, --testxmls        Specify the test.xml, support multi test.xml
                        Refer to --testxmlconfig to add by filelist
  --testxmlconfig       Specify the file listing group of testxmls
                        Support multi testxmlconfig files
  -D, --dryrun          Dry-run the selected test cases
  -V, --validate-only   Do only input xml validation, do not execute tests
  -M, --manual-only     Enable only manual tests to be executed
  -A, --auto-and-manual
                        Enable both automatic and manual tests
  -S, --significant-only
                        Enable only significant tests to be executed
  -C, --compatibleresultxml
                        use nokia compatible result xml format
  -T, --stripresultxml  to strip those non-executed cases from result xml
  -o RESULTFILE, --output=RESULTFILE
                        specify output file for result xml
  -r REPORTERNAME, --reporter=REPORTERNAME
                        specify external reporter for publishing result
  -E ENGINE, --engine=ENGINE
                        select testcase parser engine
  -w                    run web API test with chromium-browser envriorment
  --webruntime=WEBRUNTIME
                        Run into web API test envriment
  --webtestid=TESTID    Run the web API test ID=webtestid
  --webhidestatus       Run web API test in full screen mode
  --webserver=SERVER    Server for web API test
  --testcase            Select the specified white-rules filter: testcase
  --testsuite           Select the specified white-rules filter: testsuite
  --testset             Select the specified white-rules filter: testset
  --status              Select the specified white-rules filter: status
  --type                Select the specified white-rules filter: type
  --priority            Select the specified white-rules filter: priority
  --category            Select the specified white-rules filter: category
  --component           Select the specified white-rules filter: component
  --execution_type      Select the specified white-rules filter:
                        execution_type
  --Ntestcase           Select the specified black-rules filter: testcase
  --Ntestsuite          Select the specified black-rules filter: testsuite
  --Ntestset            Select the specified black-rules filter: testset
  --Nstatus             Select the specified black-rules filter: status
  --Ntype               Select the specified black-rules filter: type
  --Npriority           Select the specified black-rules filter: priority
  --Ncategory           Select the specified black-rules filter: category
  --Ncomponent          Select the specified black-rules filter: component
  --Nexecution_type     Select the specified black-rules filter:
```

Figure 3-1 Testkit-Lite help information

2. Deploy the following information to the target test device:
   - Test case descriptor file
   - Test scripts, programs, and HTML web test page
   - Dependency files or test data
3. Run test cases.

- To run **Non-WebAPI** test cases:

$ testkit-lite -f /PATH/TO/<test_descriptor_file>.xml

- To run **WebAPI** test cases:

$ testkit-lite -e "WRTLauncher <widget _name>" -f /PATH/TO/<test_descriptor_file>.xml

For details on more options and usages, see the *Testkit-Lite User Guide*.

5. Checking Test Report

Testkit-Lite creates both .xml and text test reports and locate them under **/opt/testkit/lite/latest** after it completes executing all test cases successfully. Below are examples.

- **.xml Test Report**

Figure 5-1 .xml test report

- **Text Test Report**

```
==================================TestReport====================================
                                                          TYPE PASS FAIL  N/A
--/usr/share/blts-bluetooth-tests/tests.xml                XML    2    0    0
  `---blts-bluetooth-tests                                SUITE    2    0    0
      `---bt-1dev-tests                                     SET    2    0    0
          |---HAL-Bluetooth drivers and userspace check    CASE    1    0    0
          `---HAL-Bluetooth scan                           CASE    1    0    0
```

Figure 5-2 Text test report