Testkit-Lite User Guide

Contents

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:

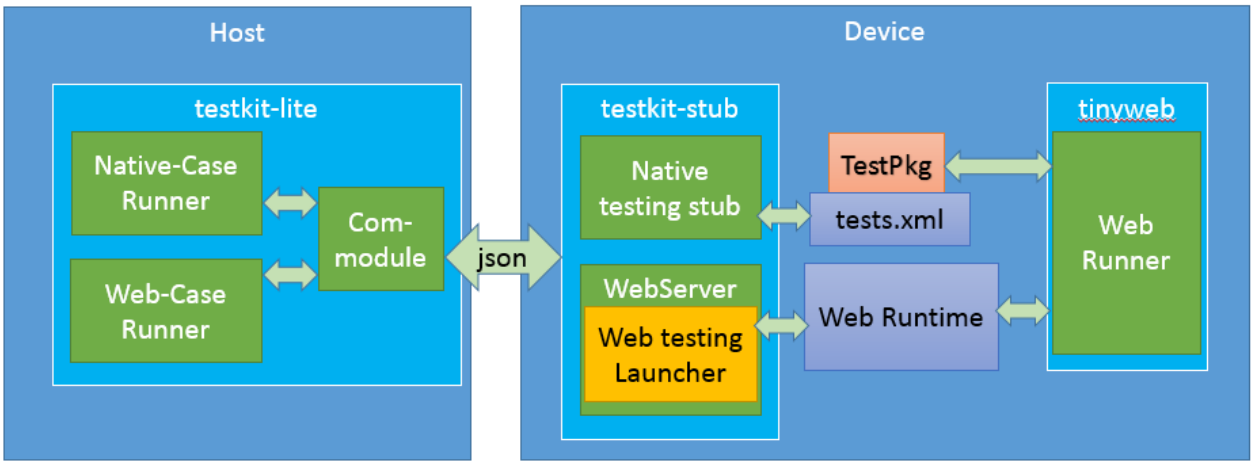
- Accepts test definition XML files as input.
- Drives automatic test execution and collects test result
- Provides multiple options to support test purpose, such as filters, external test launcher
- Supports test on cross-platforms, such as : TIZEN, android, Linux Distro like Ubuntu/Fedora/Deepin, Windows,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

3. Deployment

Testkit-Lite is composed of 3 components testkit-lite, tinyweb and testkit-stub. The 3 components can be deployed on the same target machine, or deployed on target device and host machine

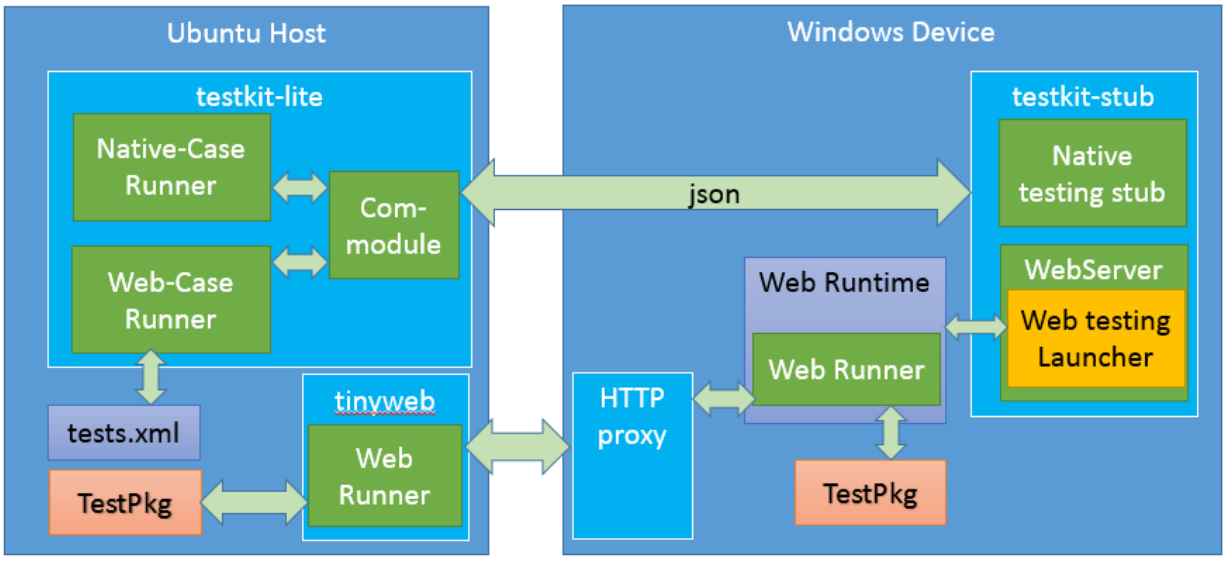
- Distributed deployment.

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



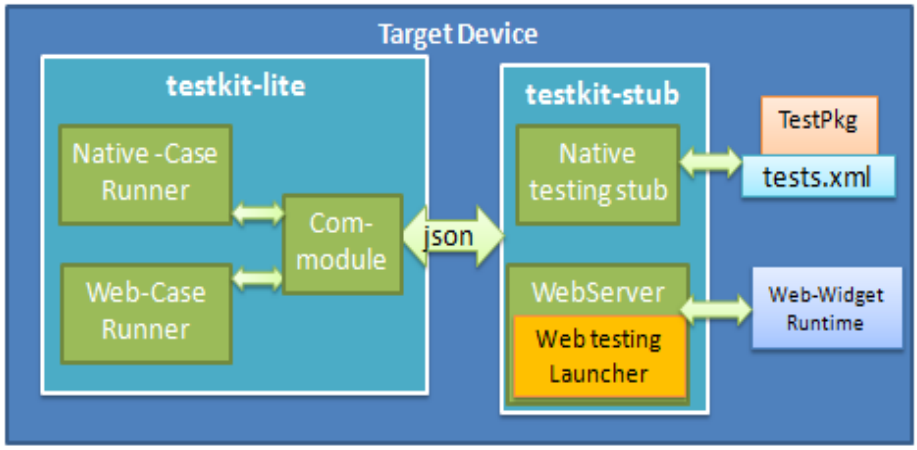
- Distributed deployment for Windows device.

  In this deployment, testkit-lite and testkit-stub are deployed as same as previous deployment mode. Tinyweb is deployed on host machine. And a “HTTP Proxy” will be deployed on Windows 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 libraries:

1. python-setuptools, python-support, python-pip

- **Ubuntu(Deepin)**

  $ sudo apt-get install python-setuptools python-support python-pip

- **Fedora(RHEL)**

  $ sudo yum install python-setuptools python-support python-pip

- **OpenSUSE(TIZEN)**

```
$ sudo zypper install python-setuptools python-support python-pip
```

- **Windows**

  Download python-setuptools, python-pip packages from https://pypi.python.org/pypi/, use

  ```
  $ python setup.py install
  ```

  to install setuptools and pip

2. python-requests(>=1.1)

- **Windows**

  Download python-requests package from https://pypi.python.org/pypi/, use

  ```
  $ python setup.py install
  ```

  to install requests

- **Others**

  ```
  $ sudo pip install requests
  ```

3. pywin32 ***only*** for Windows

   Download pywin32-219.win-amd64-py2.7.exe from http://sourceforge.net/projects/pywin32/files/pywin32, and install it

5. Install

5.1 Install testkit-lite

- **Download testkit-lite released package**

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

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

  For Windows/Mac OS, download the zip package of testkit-lite

- **On Host Ubuntu/Deepin:**

  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
  ```

- **On Host TIZEN/Fedora/RHEL/OpenSUSE:**

  Navigate to the work directory testkit-lite:

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

- **On Host Windows/Mac OS:**

  Download and unzip the zip package of testkit-lite

5.2 Install testkit-stub

- **On Android devices**

  $ adb install testkit-lite-<version>/web-test-utilities/testkit-stub/android/testkit-stub_all.apk

- **On TIZEN deives**

  $ sdb push testkit-lite-<version>/web-test-utilities/testkit-stub/tizen/arm(ia32/x64)/testkit-stub /somepathofdevice, eg: /opt/home/developer/

  $ sdb shell “chmod +x /opt/home/developer/testkit-stub”

- **On Windows device**

  Copy the binary of testkit-stub into the folder “c:\stub\” with “pthreadGC2.dll”.

  For “pthreadGC2.dll”,

  1\ Dowload the release package of “pthread-w32” from https://www.sourceware.org/pthreads-win32/

  2\ extract the release package, the dll is in <PTHREAD-W32_HOME>\Pre-built.2/dll\x86\pthreadGC2.dll

  For deploying powershell script

  1\ Copy the powershell script “download.ps1” and “uninstall.ps1” into “c:\stub\powershell\”

  2\ Launch “command prompt” as administrator. (Right-click the command app, and click “Run as administrator”)

  3\ Execute command “powershell” in command line.

  4\ Allow system to execute script:

  ps c:\> Set-ExecutionPolicy RemoteSigned

  choose A to allow executing Powershell scripts.

  Create a folder “packages” for downloading msi files.

  After deploying, the tree of stub will be

  C:\stub
  | pthreadGC2.dll
  | testkit-stub.exe
  |
  ├─packages
  └─powershell
  download.ps1
  uninstall.ps1

5.3 Install tinyweb for those test who needs Web Service

- **On Android devices**

  $ adb install testkit-lite-<version>/web-test-utilities/tinyweb/tinyweb_all.apk

- **On TIZEN deives**

```
$ cd testkit-lite-<version>/web-test-utilities/tinyweb/<arch>
$ sdb push tinyweb /opt/home/developer/
$ sdb shell "chmod a+x /opt/home/developer/tinyweb"
$ sdb push cgi-getcookie /opt/home/developer/
$ sdb shell "chmod a+x /opt/home/developer/cgi-getcookie"
$ sdb push cgi-getfield /opt/home/developer/
$ sdb shell "chmod a+x /opt/home/developer/cgi-getfield"
$ sdb push libmongoose.so /opt/home/developer/
$ sdb shell "chmod 666 /opt/home/developer/libmongoose.so"
$ sdb push echo.so /opt/home/developer/
$ sdb shell "chmod 666 /opt/home/developer/echo.so"
$ sdb push server.pem /opt/home/developer/
$ sdb shell "chmod 666 /opt/home/developer/server.pem"
```

When TIZEN OS is x86_64:

```
$ sdb shell "ln -s /usr/lib64/libssl.so.1.0.0 /opt/home/developer/libssl.so"
$ sdb shell "ln -s /usr/lib64/libcrypto.so.1.0.0 /opt/home/developer/libcrypto.so"
```

When TIZEN OS is i386:

```
$ sdb shell "ln -s /usr/lib/libssl.so.1.0.0 /opt/home/developer/libssl.so"
$ sdb shell "ln -s /usr/lib/libcrypto.so.1.0.0 /opt/home/developer/libcrypto.so"
```

- **On Ubuntu host (Windows device testing)**

```
$ cd testkit-lite-<version>/web-test-utilities/tinyweb/<arch>
$ cp -r tinyweb /opt/home/developer/
$ chmod a+x /opt/home/developer/tinyweb
$ cp -r cgi-getcookie /opt/home/developer/
$ chmod a+x /opt/home/developer/cgi-getcookie
$ cp -r cgi-getfield /opt/home/developer/
$ chmod a+x /opt/home/developer/cgi-getfield
$ cp -r libmongoose.so /opt/home/developer/
$ chmod 666 /opt/home/developer/libmongoose.so
$ cp -r echo.so /opt/home/developer/
$ chmod 666 /opt/home/developer/echo.so
$ cp server.pem /opt/home/developer/
$ chmod 666 /opt/home/developer/server.pem
$ ln -s /usr/lib/x86_64-linux-gnu/libssl.so.1.0.0 /opt/home/developer/libssl.so
$ ln -s /usr/lib/x86_64-linux-gnu/libcrypto.so.1.0.0 /opt/home/developer/libcrypto.so
```

In the “docroot” folder, create a new folder <DOCROOT>/packages/. All test suites (.zip files) which will be auto deployed on windows should be deployed in this folder first.

5.4 Use config-device.sh script to install testkit-stub and tinyweb automatically

```
$ cd testkit-lite-<version>/web-test-utilities
$ ./config-device.sh -m [adb|sdb|ssh|local] -a username -s deviceid -o [install|purge]
```

5.5 Install “HTTP proxy”or TCP/IP monitor

1\ Java is supported on Windows device

Install “Apache TCPMon”.

Download the jar from https://code.google.com/p/tcpmon/

2\ Java is not supported

TBD. Candidate tool

micro-proxy: http://acme.com/software/micro_proxy/

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

- Run webapi test cases on Windows device with crosswalk runtime

$ testkit-lite -f "<somewhere>/<test_definition_webapi_file_xwalk>.xml" –comm windowshttp –deviceid <IP address of windows device>

- Launch testkit-stub (only for Windows device) as administrator

c:\> cd <TESTKIT-STUB_HOME>

c:\>testkit-stub

- Run “HTTP proxy”

Launching Tcpmon: double-click on the downloaded executable jar file.

Alternatively, execute below command.

c:\> cd <TCPMON_HOME>

c:\> java –cp tcpmon-1.1.jar com.codegoogle.tcpmon.MainWindow

Listen and forward ports: 8080, 8081, 8082, 8083 and 8443. Target host is the IP of Ubuntu host.

- **Standalone deployment**

  - Run core test cases:

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

  - Run webapi test cases with crosswalk

$ testkit-lite -f "<somewhere>/<test_definition_webapi_file>.xml" -e “XWalkLauncher” --comm localhost

- Run webapi test cases with cordova

$ testkit-lite -f "<somewhere>/<test_definition_webapi_file>.xml" -e “CordovaLauncher” --comm localhost

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 XWalkLauncher (default, can be omitted) or CordovaLauncher.
--comm <comm_type>	Specify commodule type, “tizenmobile” uses sdb mode to access tizen device, it is default value. “tizenivi” uses ssh mode to access tizen device “tizenlocal” is for localhost access on tizen device. “androidmobile” is for android device access “deepin” is for deepin 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” or /somepath/
--testenvs=TEST_ENV	set environs for test case execution, use ';' to separator multi option
--version	Show version information of testkit-lite
--internal-version	Show internal version information of testkit-lite
-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 on Ubuntu/Deepin etc. And for Windows and Mac OS, result file would save on testkit-lite-<version>/result floder
--capability	Specify hardware capability file to filter test case
--non-active	Disable the ability to set the result of core manual cases from the console, it needs when testing on Mac OS/Windows

- Show the help information:

$ testkit-lite --help/-h

- Show the version of lite:

  $ testkit-lite --version

- Show the internal version of lite:

  $ testkit-lite --internal-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 deivce :"<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

  - Run test with TIZEN device with "crosswalk" use **sdb** to access

  $ testkit-lite -f deivce :"<somewhere>/<test_definition_webapi_file>.xml" -e "XWalkLauncher" --comm tizenmobile

  - Run test with TIZEN device with "crosswalk" use **ssh** to access

  $ testkit-lite -f deivce :"<somewhere>/<test_definition_webapi_file>.xml" -e "XWalkLauncher" --comm tizenivi

  - Run test with android mobile device with "crosswalk"

  $ testkit-lite -f deivce :"<somewhere>/<test_definition_webapi_file>.xml" -e "XWalkLauncher"

--comm androidmobile

- Run test with android mobile device with “crodova”

$ testkit-lite -f deivce :”<somewhere>/<test_definition_webapi_file>.xml” -e “CordovaLauncher” --comm androidmobile

- Run test with deepin device with “crosswalk”

$ testkit-lite -f deivce :”<somewhere>/<test_definition_webapi_file>.xml” -e “XWalkLauncher” --comm deepin

- Run test on **Mac OS/Windows** device with “crosswalk”

$ cd testkit-lite-<vesion>

$ python testkit-lite -f deivce :”<somewhere>/<test_definition_webapi_file>.xml” -e “XWalkLauncher” --comm localhost

- **“--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.

Single condition, or combined conditions are both supported.

When use combined conditions, the conditions perform “AND” logic.

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.

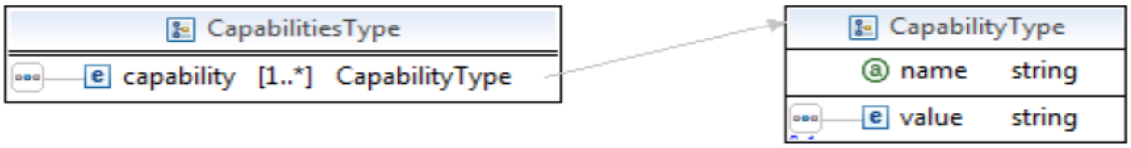
When launch a test cycle includes these test sets, Testkit-Lite will decide to execute or skip a Test set of based on actual hardware capabilities of device under test:

- If anyone 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_tcttool, tct-mgror 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 indevice.
   $ sdb push /path/to/getCap.wgt /opt/usr/media/tct/
   $ sdb shell wrt-installer -i/opt/usr/media/tct/getCap.wgt
2. Launch getCapby clicking the app iconand 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_tcttar 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)