FirefoxOS Certification Testsuite Documentation

Release 1.0

Mozilla Corporation and Individual Contributors

June 13, 2014

CONTENTS

Tests and tools to verify the functionality and characteristics of Firefox OS on real devices.

Contents:

CHAPTER

ONE

REQUIREMENTS

The test suite is designed to run on unprivileged devices and does not rely on high level instrumentation. Instead human interaction is needed throughout the test suite to perform various instructions in order to assert that conditions are met on the device.

The certification test suite is intended to run on a host computer attached to the device via USB cable. Currently the host requires the Linux or Mac OS operating systems with *adb* (Android Debug Bridge) installed.

If you need to install adb, see https://developer.mozilla.org/en-US/Firefox_OS/Debugging/Installing_ADB.

Once installed, adb must be on your PATH. If you use the bash shell emulator you can modify your *~/.bashrc* or equivalent file, by adding the following line to the file (replacing $SDK_HOME with the location of the Android SDK:

```
export PATH=$SDK_HOME:$PATH
```

The device must have a SIM card with a functioning phone subscription to receive SMS messages for a subset of the tests to pass.

1.1 Network

The device must be connected to WiFi and must have network access to the host machine.

On the host machine, the following ports are required, and must not have any existing servers running on them:

- 2828
- 6666
- 8000
- 8001
- 8888

Any network firewall must be configured to allow the device to access the host computer on the above ports.

Additionally, if you run the test suite on Mac you need to disable the system firewall so that the servers used as part of the test suite can listen to the ports mentioned above. To do this, head to the *Security & Privacy* pane in *System Preferences* and click the *Turn Off Firewall* button if present.

CHAPTER

TWO

SETUP AND USAGE

There are two methods for setting up and running the test suite: The “quick” method and the “virtualenv” way. For either to work you must enable adb access to the device.

2.1 Enabling ADB

For Firefox OS version 1.3: Launch *Settings*, and navigate to *Device Information* → *More Information* → *Developer*, then check *Remote Debugging*.

For version 1.4: Launch *Settings*, and navigate to *Device Information* → *More Information*, then check *Developer Options*. Next, hold down the *Home* button, and close the *Settings* app (press the *X*). Finally, launch *Settings* again, and navigate to *Developer*, then select *ADB only* in *Remote Debugging*.

Once this is done, go to *Settings* → *Display* and set the *Screen Timeout* to “never”. You need this because adb will not work when the device is locked.

2.2 Quick Setup and Usage

You can setup your environment and run the tests by running:

```
./run.sh --version=<some version>
```

The *–version* argument is optional. If passed, *–version* must be one of our supported release versions, either 1.3 or 1.4. If you don’t pass a version, 1.3 will be assumed.

This command sets up a virtual environment for you, with all the proper packages installed, activates the environment, runs the tests, and lastly deactivates the environment.

You may call *run.sh* as many times as you like, and it will run the tests using its previously set up virtual environment.

Some of the tests for Web APIs require manual user intervention. At this point a browser will open on your host computer. Simply follow the instructions given on screen.

2.3 Setup and Usage With virtualenv

If the quick setup doesn’t work, then follow these instructions. You can set up and run this tool inside a virtual environment. From the root directory of your source checkout, run:

```
virtualenv .
./bin/pip install -e .
```

Then activate the virtualenv:

```
source bin/activate
```

Once the virtualenv is activated, the certification test suite can be run by executing:

```
runcertsuite
```

To get a list of command-line arguments, use:

```
runcertsuite --help
```

For example it is possible to list logical test groups and to run filtered test runs of only a subset of the tests:

```
runcertsuite --list-test-groups
```

2.4 Submitting Results

Once the tests have completed successfully, they will write a file containing the results to disk; by default this file is called *firefox-os-certification.zip* and will be put in your current working directory. Please e-mail this file to fxos-cert@mozilla.com.

2.5 Known Issues

- Tests fail if re-run on a device they've already been run on [https://bugzilla.mozilla.org/show_bug.cgi?id=995455]. Workaround: re-install gaia or re-flash the device before re-running the tests.

CHAPTER

THREE

INTERPRETING RESULTS

After running the FxOS Certification Suite, a result file will be generated (firefox-os-certification.zip by default) in the current directory. Inside this file you will find a file named cert/results.json. It contains a number of top-level fields, which are explained below.

3.1 omni_results

This section contains the output of the omni_analyzer tool. The omni_alayzer compares all the JS files in omni.ja on the device against a reference version. If any differences are found, the entire file containing the differences is base-64 encoded and included in the result file.

To see the diffs between the files on the device and the reference versions, use the omni_diff.py tool inside the certsuite package. To run this tool:

> source certsuite_venv/bin/activate # this will exist after you run the tests cd certsuite python omni_diff.py /path/to/cert_results.json expected_omni_results/omni.ja.1.3 results.diff

You can then view results.diff in an editor.

Differences in omni.ja files are not failures; they are simply changes that should be reviewed in order to verify that they are harmless, from a branding perspective.

3.2 application_ini

This section contains the details inside the application.ini on the device. This section is informative.

3.3 headers

This section contains all of the HTTP headers, including the user-agent string, that the device transmits when requesting network resources. This section is informative.

3.4 buildprops

This section contains the full list of Android build properties that the device reports. This section is informative.

3.5 kernel_version

This section contains the kernel version that the device reports. This section in informative.

3.6 processes_running

This section contains a list of all the processes that were running on the device at the time the test was performed. This section is informative.

3.7 [unpriv|priv|cert]_unexpected_webidl_results

This section, if present, represents differences in how interfaces defined in WebIDL files in a reference version differ from the interfaces found on the device in an (unprivileged|privileged|certified) context. For example:

> **{** "message": "assert_true: The prototype object must have a property "textTracks" expected true got false", "name": "HTMLMediaElement interface: attribute textTracks", "result": "FAIL"
>
> },

This means that the HTMLMediaElement interface was expected to expose a textTracks attribute, but that attribute was not found on the device.

3.8 [unpriv|priv|cert]_added_window_functions

This section, if present, lists objects descended from the top-level 'window' object which are present on a reference version, but not present on the device, in an (unprivileged|privileged|certified) context.

3.9 [unpriv|priv|cert]_missing_window_functions

This section, if present, lists objects descended from the top-level 'window' object which are present on the device, but not on a reference version, in an (unprivileged|privileged|certified) context.

3.10 [unpriv|priv|cert]_added_navigator_functions

This section, if present, lists objects descended from the top-level 'navigator' object which are present on a reference version, but not present on the device, in an (unprivileged|privileged|certified) context.

3.11 [unpriv|priv|cert]_missing_navigator_functions

This section, if present, lists objects descended from the top-level 'navigator' object which are present on the device, but not on a reference version, in an (unprivileged|privileged|certified) context.

3.12 [unpriv|priv|cert]_added_navigator_unprivileged_functions

This section, if present, lists objects descended from the top-level ‘navigator’ object which are reported as null on a reference version, but reported as not-null on the device. This could indicate a permissions problem; i.e., the object belongs to an API which a reference version reports as null because the API is only available to privileged contexts, and the test is run in an unprivileged context, but which is available in an unprivileged context on the device. This test is performed in an (unprivileged|privileged|certified) context.

3.13 [unpriv|priv|cert]_missing_navigator_unprivileged_functions

This section, if present, lists objects descended from the top-level ‘navigator’ object which are reported as not-null on a reference version, but reported as null on the device. This could indicate a permissions problem; i.e., the object belongs to an API which should be available to unprivileged contexts, but which is not available to an unprivileged context on the device. This test is performed in an (unprivileged|privileged|certified) context.

CHAPTER

FOUR

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*