gaiatest Documentation

Release master

Mozilla Automation and Tools team

Oct 08, 2017

Contents

gaiatest is a Python package based on Marionette, which is designed specifically for writing tests against Gaia.

CHAPTER 1

Installation

You will need a Marionette enabled Firefox build that you can successfully connect to.

Before installing gaiatest you may want to consider creating a virtual environment:

```
virtualenv env_name
source env_name/bin/activate
```

If you only want to use gaiatest without making changes:

```
pip install gaiatest
```

However, if you want to modify gaiatest, first clone the Gaia repository before running setup.py:

```
git clone https://github.com/mozilla-b2g/gaia.git
cd gaia/tests/python/gaia-ui-tests
python setup.py develop
```

CHAPTER 2

Command line interface

A helpful command line tool is provided for interacting with Gaia. For full usage details run `gcli --help` and for help on a specific command use `gcli <command> --help`.

For example, to unlock the device, set brightness to 100%, connect to an unsecured network, and launch the Settings app:

```
gcli unlock
gcli setsetting screen.brightness 1
gcli connectwifi MozillaGuest
gcli launchapp Settings
```

CHAPTER 3

Running tests

To run tests using gaia test, your command-line will vary a little bit depending on what device you're using. The general format is:

```
gaiatest [options] /path/to/test_foo.py
```

For full usage run:

```
gaiatest --help
```

Risks

The gaiatest runner restores the target Firefox OS instance to a 'clean' state before every test. This means that any personal data such as contacts, SMS/MMS messages, emails, photos, videos, music, etc. will be removed. This will include data on the microSD card. The tests may also attempt to initiate outgoing calls, and may connect to services such as cellular data, WiFi, GPS, bluetooth, etc. Therefore, running tests using the gaiatest runner may cause unintentional data loss and may incur unexpected charges. In order to acknowledge these risks, you must enter the `acknowledged_risks` key in your test variables file with a true value. Note that this is a JavaScript boolean, so the value true should not be enclosed in quotes.

Disabling the Warning

Once you have acknowledged the risks, you will still be presented with a warning every time you run the test suite. If you are absolutely sure that you will only ever run the tests against suitable test devices, this can be disabled by setting an environment variable named `GAIATEST_SKIP_WARNING`. For example, you might run the gaiatest command like so:

```
GAIATEST_SKIP_WARNING=1 gaiatest ... params
```

Testing on a device

You must run a build of B2G on the device that has Marionette enabled. The easiest way to do that is to flash your device with a nightly 'engineering' build, like this one for our reference device (access to these builds is currently restricted).

If you are running the tests on a device connected via ADB (Android Debug Bridge), you must additionally set up port forwarding from the device to your local machine. You can do this by running the command:

```
adb forward tcp:2828 tcp:2828
```

ADB can be downloaded as part of the Android SDK.

Testing on desktop

If you don't have a Firefox OS device, you can download the latest build of the desktop client from here, but make sure you download the appropriate file for your operating system.

- **Linux (32bit)**: b2g-[VERSION].multi.linux-i686.tar.bz2
- **Linux (64bit)**: b2g-[VERSION].multi.linux-x86_64.tar.bz2
- **Mac**: b2g-[VERSION].multi.mac64.dmg
- **Windows**: b2g-[VERSION].multi.win32.zip

Once downloaded, you will need to extract the contents to a local folder. `$B2G_HOME` refers to the location of the local folder for the remainder of the documentation.

If a profile is specified when running the tests (recommended), a clone of the profile will be used. This helps to ensure that all tests run in a clean state. However, if you also intend to launch and interact with the desktop build manually, we recommend making a copy of the default profile and using the copy for your tests. The location of the default profile is `$B2G_HOME/gaia/profile`.

Note: The test framework only supports running the tests and B2G instances locally, although in theory it can be used against a running desktop build accessible over the network.

To run the full suite of tests, use the following command:

```
cd gaia/tests/python/gaia-ui-tests
gaiatest --restart --type b2g --binary $B2G_HOME/b2g-bin --profile $B2G_HOME/gaia/
↪profile \
  --testvars path/to/testvars.json gaiatest/tests/manifest.ini
```

Filtering tests

Tests can be filtered by type, and the types are defined in the manifest files. Tests can belong to multiple types, some types imply others, and some are mutually exclusive - for example a test cannot be both 'online' and 'offline' but a test that is 'lan' is by definition 'online'. Be warned that despite these rules, there is no error checking on types, so you must take care when assigning them. Default types are set in the `[DEFAULT]` section of a manifest file, and are inherited by manifest files referenced by an include.

Here is a list of the types used, and when to use them:

- antenna - Antenna (headphones) must be connected.
- b2g - This means the test is a B2G (Firefox OS) test. All tests must include this type.
- bluetooth - Bluetooth is required.
- camera - Camera is required.
- carrier - Active SIM card with carrier connection is required.
- external - The test requires access to resources outside the host and device, eg the internet.
- flash - Camera flash is required.
- lan - Local area connection (not cell data) is required by these tests (see note below).
- offline - Specifically requires no online connection.
- online - Connection (lan or carrier) is required.
- qemu - These tests require the Firefox OS emulator to run.
- sdcard - Storage device (such as an SD card) must be present.
- wifi - WiFi connection is required.
- sanity - Tests exercising core device features (dialer, camera, browser, sms).

Note: You may be thinking that there is only WiFi or cell data, and why the need for the 'lan' test type. Well, these tests aren't only run on physical devices. We also run then on desktop builds, which share the host computer's connection. It is for this reason that we need 'lan' to indicate a connection that is not cell data. For an example of where online/lan/carrier are used take a look at the browser tests.

Test variables

The –testvars option is used to pass in local variables, particularly those that cannot be checked into the repository. For example when running the Gaia UI tests, these variables can be your private login credentials, phone number or details of your WiFi connection.

To use it, copy `gaia/tests/python/gaia-ui-tests/gaiatest/testvars_template.json` to a different filename but add it into .gitignore so you don't check it into your repository.

When running your tests add the argument:

```
--testvars /path/to/testvars.json
```

Expected variables

- local_phone_numbers - Array of phone numbers that are attached to the phone.
- imei - The 12 digit IMEI code of the test phone.
- remote_phone_number - Phone number that your device can call during the tests (try not to be a nuisance!). Prefix the number with '+' and your international dialing code.
- wifi - These are the settings of your WiFi connection. Currently this supports WPA/WEP/etc. You can add WiFi networks by doing the following (remember to replace `KeyManagement` and the password with the value your network supports):

```
'wifi': {
  'ssid': 'MyNetwork',
  'keyManagement': 'WEP' or 'WPA-PSK',
  'wep': 'MyPassword',
}
```

```
'wifi': {
  'ssid': 'MyNetwork',
  'keyManagement': 'WPA-PSK',
  'psk': 'MyPassword'
}
```

Note: Due to bug 775499, WiFi connections via WPA-EAP are not capable at this time.

- email - Login information used by the email tests. It can contain different types of email accounts:

```
'gmail': {
  'name': '',
  'email': '',
  'password': ''
}
```

```
'IMAP': {
  'name': '',
  'email': '',
  'password': '',
  'imap_hostname': '',
  'imap_name': '',
  'imap_port': '',
  'smtp_hostname': '',
  'smtp_name': '',
  'smtp_port': ''
}
```

```
'ActiveSync': {
  'name': '',
  'email': '',
  'password': '',
  'active_sync_hostname': '',
  'active_sync_username': ''
}
```

- settings - Custom settings to override the Gaia default settings. These will be set before each test run but are not mandatory.

```
'settings': {
  '<setting>': <value>
}
```

When running with no SIM card or offline the timezone may not be automatically updated to match the local timezone. In that case you may need to force the timezone to match the desired timezone using settings in `testvars.json` which will set it during the test setup:

```
'settings': {
  'time.timezone': '<value>',
  'time.timezone.user-selected': '<value>'
}
```

- prefs - Custom preferences to override the Gecko default preferences. These will be set before each test run but are not mandatory.

```
'prefs': {
  '<name>': <value>
}
```

Test data prerequisites

Occasionally a test will need data on the hardware that cannot be set during the test setup. The following tests need data set up before they can be run successfully:

- test_ftu - Requires a single record/contact saved onto the SIM card to test the SIM contact import.

CHAPTER 4

Writing tests

Test writing for Marionette Python tests is described here.

At the moment we don't have a specific style guide. Please follow the prevailing style of the existing tests. Use them as a template for writing your tests. We follow PEP 8 for formatting, although we're pretty lenient on the 80-character line length.

CHAPTER 5

API usage

App manager

Data manager

Device manager

File manager

Test cases

CHAPTER 6

Apps

Base

Browser

Regions

Calendar

Regions

Clock

Regions

Contacts

Regions

Cost control

Regions

Email

Regions

Emergency call

FM radio

CHAPTER 7

Indices and tables

- genindex
- modindex
- search