Ubuntu Test Cases Touch Documentation

Release latest

March 18, 2015

Contents

The CI touch testing execution framework was written so that it's very easy to run tests from home in the exact same way tests are run for the CI dashboard and CI MP testing. The only things you need are:

- This bzr branch: lp:ubuntu-test-cases/touch
- The phablet-tools and ubuntu-device-flash packages
- An Ubuntu Touch supported device

There are two pieces to touch testing, provisioning and test execution. The provisioning step is required, but once provisioned, you can proceed with any of the test methods described in this document.

Note: Re-creating test results to match those on http://ci.ubuntu.com/ is not always possible. The Ubuntu touch images are created using the Ubuntu archive which changes over time as packages are constantly being updated. Even when using identical images, test dependencies may change making it impossible to recreate an identical test environment.

For best results, it's recommended to always use the most recent image.

CHAPTER 1

Prerequisites

Before proceeding, a network configuration file is required. This is a one time setup and can be used for all future provisioning and testing as long as your wifi network does not change. This file is the same network configuration file used by *phablet-network* and can be found by running *phablet-network* with no options. For example:

```
$ phablet-network
Network file is /etc/NetworkManager/system-connections/my-network-ssid
Provisioning network on device

Network setup complete
PING launchpad.net (91.189.89.222) 56(84) bytes of data.
```

Once the network configuration file is identified, it can be copied into the default location to be used by the provisioning scripts:

```
mkdir ~/.ubuntu-ci
sudo cp /etc/NetworkManager/system-connections/my-network-ssid ~/.ubuntu-ci/wifi.conf
sudo chown $USER:$USER ~/.ubuntu-ci/wifi.conf
```

CHAPTER 2

Provisioning

Provisioning is a required step before performing CI dashboard or MP testing. It is necessary to configure the device into a known state so that the tests can run in a predictible manner.

> **Warning:** This provisioning step will completely erase your device. Be sure to backup any data before proceeding.

The provisioning script is a simple wrapper to commands from phablet-tools and some additional device setup to get it ready for testing. Provisioning is performed with the scripts/provision.sh command. Running:

```
./scripts/provision.sh -w
```

will install the latest ubuntu-touch/devel-proposed image. If you wish to install the latest ubuntu-rtm image instead, use:

```
export IMAGE_OPT="--bootstrap --developer-mode --channel=ubuntu-touch/ubuntu-rtm/14.09-proposed"
./scripts/provision.sh -w
```

Provisioning using this script requires that you start off with the device booted and accessible via ADB. The device will be rebooted automatically and completely reinstalled - **ALL DATA WILL BE LOST**.

Note: provision.sh requires a path to a network-manager wifi configuration file that can be copied to the target device as described above in the Prerequisites section. By default this is set to ${HOME}/.ubuntu-ci/wifi.conf. This can be overridden with the -n parameter.

Other customizations can be performed during provisioning, for a full list of options, use:

```
./scripts/provision.sh -h
```

CHAPTER 3

Executing Tests

The touch testing tools are intended to provide CI Dashboard and CI MP testing for specific applications. The supported applications and test suites are visible in the CI Dashboard. These tests include both autopilot and UTAH test definitions. The following sections describe how to use these tools for testing on a local device.

Note: These tools will *only* work on a device that has been provisioned using the methods described above.

Note: Some of the tests generate subunit result files. These subunit result files provide richer content and potentially more test artifacts over the xml result files. To view the contents of these files, use the trv viewer application or your favorite subunit viewer.

3.1 Executing Autopilot Tests

One or more autopilot tests can be run on the target device using the command:

```
./scripts/run-autopilot-tests.sh
```

This is a small wrapper that uses phablet-tools to drive the tests. The script can run one or more autopilot tests. By default it will reboot the device between each test and ensure the device is settled using the *system-settle* script. Both of those options can be disabled via command line options. By default the script will create a directory named *clientlogs* and then a subdirectory for each testsuite with result files. These sub-directories include a xUnit XML formatted file, *test_results.xml*, a subunit result stream, *test_results.subunit*, as well as several log files from the device to help with debugging failures.

Note: run-autopilot-tests.sh will call a script that installs unity8-autopilot if it is not already installed, to allow the device to be unlocked automatically.

3.1.1 Examples

An example testing two applications:

```
./scripts/provision.sh -w
./scripts/run-autopilot-tests.sh -a dropping_letters_app -a music_app
```

And an example for running all dashboard autopilot tests:

```
./scripts/provision.sh -w
./scripts/run-autopilot-tests.sh
```

3.2 Executing UTAH Tests

Executing UTAH tests locally will require you to install the UTAH client package from a PPA:

```
sudo add-apt-repository ppa:utah/stable
sudo apt-get update
sudo apt-get install utah-client
```

With that package installed UTAH tests can be run with:

```
./scripts/jenkins.sh
```

This script runs one test at a time and will put its test artifacts under the *clientlogs* directory similar to the autopilot runner. The UTAH result file will be named clientlogs/utah.yaml.

An example of running the sdk test suite:

```
./scripts/jenkins.sh -a sdk
```

3.3 Provisioning and Executing Autopilot Tests for an MP

These scripts are used by jenkins for the testing of MPs that generate Debian packages. The *run-mp.sh* script used below includes provisioning, so for this one case, it is not necessary to call provision.sh separately.

To re-create the testing performed by jenkins, set the following environment variables based on the jenkins build parameters:

```
export package_archive=<from jenkins build parameter>
export test_packages=<from jenkins build parameter>
export test_suite=<from jenkins build parameter>
```

and set the variable:

```
export ANDROID_SERIAL=<adb id from your test device>
```

Then execute the following script:

```
./scripts/run-mp.sh
```

3.4 Running Autopilot Tests for a Modified Click Application

First provision the device with the desired image using the instructions in the "Provisioning" section of this README.

Once the image has been provisioned, install the click app to test. The dropping-letters application is used in this example:

```
adb push com.ubuntu.dropping-letters_0.1.2.2.67_all.click /tmp
adb shell pkcon --allow-untrusted install-local \
    /tmp/com.ubuntu.dropping-letters_0.1.2.2.67_all.click
```

Now install the test sources ('–wipe' will remove any previously installed test sources):

```
phablet-click-test-setup --wipe --click com.ubuntu.dropping-letters
```

The above phablet-click-test-setup command will install the standard test dependencies and the click application's test sources as specified in the manifest. This is usually the application's trunk branch. To override the test sources with local changes, replace the test sources that were copied to the device. This example assumes the application code is checked out under the 'dropping-letters' directory with the test sources under 'tests/autopilot':

```
adb shell rm -rf /home/phablet/autopilot/dropping_letters_app
adb push dropping-letters/tests/autopilot \
    /home/phablet/autopilot
```

Finally, run the application tests:

```
./scripts/run-autopilot-tests.sh -a dropping_letters_app
```

The test results are available under:

```
clientlogs/dropping_letters_app/test_results.subunit
clientlogs/dropping_letters_app/test_results.xml
```

3.5 Running Autopilot Tests for a Modified Debian Package

First provision the device with the desired image using the instructions in the "Provisioning" section of this README.

If the device is provisioned, and you have built the debian package you wish to test with locally, install it on the device. For instance, if you are building and installing dialer-app:

```
phablet-config writable-image -r 0000 --package-dir /path/to/packages -p dialer-app
```

Alternatively, if you have built the packages in a ppa, you could use:

```
phablet-config writable-image -r 0000 --ppa ppa:ci-train-ppa-service/landing-004 -p dialer-app
```

Note: If you have updates to the dependencies or tests in debian packages, make sure to also install packages for those if required for the change you are making. Some tests need a few extra dependencies installed for the tests to function correctly. To see a list of them, look at jenkins/testconfig.py.

Finally, run the application tests:

```
./scripts/run-autopilot-tests.sh -a dialer_app
```

The test results are available under:

```
clientlogs/dialer_app/test_results.subunit
clientlogs/dialer_app/test_results.xml
```