CTS Development

Initializing Your Repo Client

Follow the instructions to get and build the Android source code but specify `-b gingerbread` when issuing the `repo init` command. This assures that your CTS changes will be included in the next CTS release and beyond.

Setting Up Eclipse

Follow the instructions to setup Eclipse but execute the following command to generate the `.classpath` file rather than copying the one from the development project:

```
cd /path/to/android/root
./cts/development/ide/eclipse/genclasspath.sh > .classpath
chmod u+w .classpath
```

This `.classpath` file will contain both the Android framework packages and the CTS packages.

Building and Running CTS

Execute the following commands to build CTS and start the interactive CTS console:

```
cd /path/to/android/root
make cts
cts
```

Provide arguments to CTS to immediately start executing a test:

```
cts start --plan CTS -p android.os.cts.BuildVersionTest
```

Writing CTS Tests

CTS tests use JUnit and the Android testing APIs. Review the Testing and Instrumentation tutorial while perusing the existing tests under the `cts/tests/tests` directory. You will see that CTS tests mostly follow the same conventions used in other Android tests.

Since CTS runs across many production devices, the tests must follow these rules:

- Must take into account varying screen sizes, orientations, and keyboard layouts.
- Only use public API methods. In other words, avoid all classes, methods, and fields that are annotated with the "hide" annotation.
- Avoid relying upon particular view layouts or depend on the dimensions of assets that may not be on some device.

- Don't rely upon root privileges.

Test Naming and Location

Most CTS test cases target a specific class in the Android API. These tests have Java package names with a `cts` suffix and class names with the `Test` suffix. Each test case consists of multiple tests, where each test usually exercises a particular API method of the API class being tested. These tests are arranged in a directory structure where tests are grouped into different categories like "widgets" and "views."

For example, the CTS test for `android.widget.TextView` is `android.widget.cts.TextViewTest` found under the `cts/tests/tests/widget/src/android/widget/cts` directory with its Java package name as `android.widget.cts` and its class name as `TextViewTest`. The `TextViewTest` class has a test called `testSetText` that exercises the "setText" method and a test named "testSetSingleLine" that calls the `setSingleLine` method. Each of those tests have `@TestTargetNew` annotations indicating what they cover.

Some CTS tests do not directly correspond to an API class but are placed in the most related package possible. For instance, the CTS test, `android.net.cts.ListeningPortsTest`, is in the `android.net.cts`, because it is network related even though there is no `android.net.ListeningPorts` class. You can also create a new test package if necessary. For example, there is an "android.security" test package for tests related to security. Thus, use your best judgement when adding new tests and refer to other tests as examples.

Finally, a lot of tests are annotated with @TestTargets and @TestTargetNew. These are no longer necessary so do not annotate new tests with these.

New Test Packages

When adding new tests, there may not be an existing directory to place your test. In that case, refer to the example under `cts/tests/tests/example` and create a new directory. Furthermore, make sure to add your new package's module name from its `Android.mk` to `CTS_COVERAGE_TEST_CASE_LIST` in `cts/CtsTestCaseList.mk`. This Makefile is used by `build/core/tasks/cts.mk` to glue all the tests together to create the final CTS package.

Test Stubs and Utilities

Some tests use additional infrastructure like separate activities and various utilities to perform tests. These are located under the `cts/tests/src` directory. These stubs aren't separated into separate test APKs like the tests, so the `cts/tests/src` directory does not have additional top level directories like "widget" or "view." Follow the same principle of putting new classes into a package with a name that correlates to the purpose of your new class. For instance, a stub activity used for testing OpenGL like `GLSurfaceViewStubActivity` belongs in the `android.opengl.cts` package under the `cts/tests/src/android/opengl` directory.

Other Tasks

Besides adding new tests there are other ways to contribute to CTS:

- Fix or remove tests annotated with BrokenTest and KnownFailure.

Submitting Your Changes

Follow the Android Contributors' Workflow to contribute changes to CTS. A reviewer will be assigned to your change, and your change should be reviewed shortly!