Tema Android Adapter

Date: 07.12.2010

AndroidAdapter uses Android debug bridge (adb), window service and monkey service to run automatic keyword-based tests on Android GUI. Adapter works on the emulator, and on android development phones with security features disabled. Adapter can be used to run model-based tests with the TEMA test engine.

Table of Contents

TapDownObject object
TapUpObject object
LongTapObject [time,]object
IsTrue False|True
MoveToObject object
Drag object|x,y --> object|x,y
TouchScroll object|x,y --> object|x,y
SelectFromMenu 'menu_text'
Type 'text'
PressKey key[,times]
PressKeyDown key
PressKeyUp key
LongPressKey key[,time]
MoveTrackBall dx,dy
CheckProperty propertyName,'value',object
LaunchApp 'application'
ObjectVisible object
SetNetworkDelay 'gprs|edge|umts|none'
SetNetworkSpeed 'gsm|hscsd|gprs|edge|umts|hsdpa|full'
UpdateScreen
Delay
SendSMS

Requirements

- TEMA adapterlib
- Python 2.5 or later (not 3.0!).
- Adapter requires Android SDK. The tools folder of the SDK must be found from path.
- Supports Android devices and emulators with platform versions later than 1.6. (Support for versions over 2.2 is not guaranteed)
- Adapter does not support production devices due to security reasons. Run "adb shell getprop ro.secure" to check whether security is enabled. If the command prints 1, adapter will not work!

Installation

Software listed in requirements must be installed before installing adapter.

tema-androidadapter is a normal python program and can be installed with normal commands:

```
python setup.py build
python setup.py install
```

Alternatively Debian or RPM packages can be used.

Debian installation

Adapter requires package *python-adapterlib* from TEMA Toolset.

Adapter can be installed with following commands:

```
dpkg -i python-adapterlib-VERSION.deb
dpkg -i tema-androidadapter-VERSION.deb
```

Packaging

Source packages

Makefile is included that can be used to build source distribution packages:

```
make source
```

Alternatively setup.py can be used directly to generate source package:

```
python setup.py sdist
```

Generated packages will be in directory **dist**.

Debian packages

Source package includes necessary files for building Debian packages. Makefile is included that can be used to build Debian packages:

```
make builddeb
```

Generated packages will be in directory **debbuild**.

RPM packages

Source package includes necessary files for building RPM packages. Makefile is included that can be used to build RPM packages:

```
make buildrpm
```

Generated packages will be in directory **dist**.

API Documentation

API documentation can be generated with epydoc:

```
make apidoc
```

Epydoc writes html-documentation to directory *apidoc*.

Programs

- tema.android-adapter: Adapter for running tests

Usage

1. Setup adb connection with your emulator/device. When using emulator, use SDK setup to create and start one. For setting up a real device connection, check http://developer.android.com/sdk/win-usb.html .
2. Run adapter with command: tema.androidadapter [options] target_serial_number , where target_serial_number is the serial number of the target issued in adb (e.g. emulator-5554, check command "adb devices").
    1. For running an interactive test (run keywords from command line) run e.g. tema.android-adapter -i emulator-5554
    2. For running a test with the TEMA Model-based testing server, run e.g. tema.android-adapter -a SERVER_ADDRESS -p SERVER_PORT emulator-5554
    3. For more options, run tema.android-adapter -h
3. Logging (--record) requires screenshot-application from the android source for screenshot functionality. Compile and set up with following instructions:

Screenshot-application

1. copy <androidsource>sdkscreenshotsrccom folder to <androidsdk>tools
2. with command line, go to <androidsdk>tools
3. compile the code: javac -classpath libddmlib.jar comandroidscreenshotScreenshot.java
4. set the sdk's tools folder and the <androidsdk>toolslibddmlib.jar to the CLASSPATH environment variable
5. You should now be able to run the application from any path with the command: java com.android.screenshot.Screenshot

Keywords

This chapter describes the different keywords recognized by the adapter.

Object references

The GUI object that is used as a target for any action defined by a keyword, is needed in most of the keywords. The few different ways to specify (or to refer to) a gui object is explained in the following. Later, when the keywords are presented, the "object" - parameter is always interpreted in this same manner.

To find out the GUI Hierarchy and Object properties use the Hierarchy viewer application from Android SDK, or the "dump" -command when running the adapter in the interactive mode.

Direct reference

A component can be specified by giving its id or its text content directly in the following way:

```
id/plus
'+'
```

If the string is enclosed with single quote marks, an object is searched by its text content. Otherwise an object is searched by its id.

The type of the object can also be specified to separate objects with a same id or text. The type is separated from the id (or from the text content) with a semicolon. Either full type specification or a shortened version (final charaters of the type) can be used:

```
id/plus;Button
'+';Button
'+';com.android.calculator2.ColorButton
```

Hierarchical reference

Hierarchical references can be used to give more detailed refereces, e.g. by identifying the layout from where the object is found:

```
id/somelayout:::id/plus
id/somelayout:::'+'
```

Keyword specifications

VerifyText [<NoUpdate>][partial:]'text'[,object]

Verifies that a given text is found from some GUI object in the screen. The verification is exact, so that the text must be exactly same as the object's text (e.g. not a part of it). If the object-parameter is given, only text in that object is verified.

if <NoUpdate> -parameter is given, the keyword will not update the GUI hiearchy, but uses the previous update. This can be used to make execution faster.

if partial: -parameter is given, the search accepts components where the text is found somewhere inside its text content

Examples:

```
VerifyText '+',id/plus
VerifyText '+'
```

WaitText [partial:]'text',[timeout],[object]

Verifies that a given text is found from some GUI object in the screen during the given timeout (in seconds). The verification is exact, so that the text must be exactly same as the object's text (e.g. not a part of it). If the object-parameter is given, only text in that object is verified. Default timeout is 10 seconds

if partial: -parameter is given, the search accepts components where the text is found somewhere inside its text content

Examples:

```
WaitText '+',15,id/plus
```

WaitObject [timeout],object

Verifies that a given object is found from the screen during the given timeout (in seconds). Default timeout is 10 seconds

Examples:

```
WaitObject 15,id/plus
```

SelectFromList 'list_item'

Selects an item from a vertical list (such as Android settings). 'list_item' is the textual content of the searched list item.

Examples:

```
SelectFromList 'Call settings'
```

TapCoordinate x,y[,times]

Taps the given screen coordinate. The number of taps can be defined with the times-parameter (default is one).

Examples:

```
TapCoordinate 50,100
TapCoordinate 50,100,2
```

LongTapCoordinate x,y[,time]

Taps the given screen coordinate and holds it for a while. The default hold-time is 2 seconds.

Examples:

```
LongTapCoordinate 50,100
LongTapCoordinate 50,100,3
```

TapDownCoordinate x,y

Taps and holds the given coordinate

Examples:

```
TapDownCoordinate 50,100
```

TapUpCoordinate x,y

Releases a tap in a given coordinate

Examples:

```
TapUpCoordinate 50,100
```

MoveToCoordinate x,y

Moves the stylus in to given coordinate. Combine with TapDown and TapUp to simulate dragging.

Examples:

```
MoveToCoordinate 100,200
```

TapObject [times,]object

Taps an object once. If times-parameter is given, taps the object that many times.

Examples:

```
TapObject id/plus
```

TapDownObject object

Taps and holds the given object (from the middle)
Examples:

```
TapDownObject id/plus
```

TapUpObject object

Releases a tap on top of a give object (in the middle)
Examples:

```
TapUpObject id/plus
```

LongTapObject [time,]object

Taps the given screen object and holds it for a while. The default hold-time is 2 seconds.
Examples:

```
LongTapObject id/plus
LongTapObject 5,id/plus
```

IsTrue False|True

IsTrue returns True or False depending on the parameter
Examples:

```
IsTrue True
IsTrue False
```

MoveToObject object

Moves the stylus in to coordinates of the object. Combine with TapDown and TapUp to simulate dragging.
Examples:

```
MoveToObject id/plus
```

Drag object|x,y --> object|x,y

Performs a drag from the given object/coordinate to other object/coordinate. The coordinate/object from where the drag begins is held down 2 seconds before dragging
Examples:

```
Drag id/component --> id/component2
Drag 100,40 --> 200,40
Drag id/component --> 300,50
```

TouchScroll object|x,y --> object|x,y

Performs a touch gesture from the given object/coordinate to other object/coordinate. Simulates quick scrolling motions.
Examples:

```
TouchScroll id/component --> id/component2
TouchScroll 100,40 --> 200,40
TouchScroll id/component --> 300,50
```

SelectFromMenu 'menu_text'

The keyword will open the menu and select an item from there. Keyword can also handle situations where the item is found under the 'more'-menu. If item is not found, keyword will close the menu.
Examples:

```
SelectFromMenu 'Open'
```

Type 'text'

Types the given text
Examples:

```
Type 'Hello'
```

PressKey key[,times]

Presses the given hardware key. Hardware key codes and names can be found from http://developer.android.com/reference
The parameter can be either the key name or its integer value.
Examples:

```
PressKey KEYCODE_MENU
PressKey menu
PressKey 82
```

PressKeyDown key

Presses and holds down the given key. key-parameter specified in PressKey-keyword
Examples:

```
PressKeyDown KEYCODE_MENU
PressKeyDown menu
PressKeyDown 82
```

PressKeyUp key

Release a key. key-parameter specified in PressKey-keyword
Examples:

```
PressKeyUp KEYCODE_MENU
PressKeyUp menu
PressKeyUP 82
```

LongPressKey key[,time]

Presses and holds a key for a given time. By default the hold time is 2 seconds
Examples:

```
LongPressKey down,3
LongPressKey down
```

MoveTrackBall dx,dy

Simulates a trackball movement. dx and dy specify the amount of movement

Examples:

```
MoveTrackBall 0,3
MoveTrackBall 1,-2
```

CheckProperty propertyName,'value',object

Checks if object's property corresponds the give value. Different properties can be seen with the Hierarchy viewer application.

Examples:

```
CheckProperty mText,'Hello',id/display
```

LaunchApp 'application'

Launches the given application (activity). Activity can be specified as package.activityname when it will be launced using the am-utility or just application name, when the application is searched from the recent applications and the application menu.

If the application name begins with 'appmenu:', the application is searched only from application menu.

If the application name begins with 'recent:', the application is searched only from the recent applications list.

Mapping between the application names used in the keywords and the actual application names in the platform can be configured in the appconfig.ini file.

Examples:

```
LaunchApp 'com.android.calculator2.Calculator'
LaunchApp 'Calculator'
LaunchApp 'appmenu:Calculator'
LaunchApp 'recent:Calculator'
```

ObjectVisible object

Checks whether an object can be found from the GUI

Examples:

```
ObjectVisible id/plus
```

SetNetworkDelay 'gprs|edge|umts|none'

Sets the emulator network delay

Examples:

```
SetNetworkDelay 'gprs'
```

SetNetworkSpeed 'gsm|hscsd|gprs|edge|umts|hsdpa|full'

Sets the emulator network speed.

Examples:

```
SetNetworkSpeed 'full'
```

UpdateScreen

Delay

SendSMS