

# appiumphone

## **Introduction**

appiumphone is a appium wrapper, it follows the idea of warping selenium into seleniumpc. appiumphone's advantage is that it is easy to learn, appiumphone's disadvantage is that by now it only support native-view but not web-view.

## **Develop & Test Environment**

system: Windows 7 a64  
runtime: Python 2.7 i386  
executor: Appium 1.6.3  
phone: Android 4.1

## **Classes in selenium**

### Phone

Phone represents the instance of a phone.

### Element

Element represents a native-view element.

### Android

Element attribute collection belonging to a Android element.

### Ios

Element attribute collection belonging to a Ios element.

### Net\_Android

Android network type collection.

## **Properties & Definitions in Phone**

### Phone.identity

Android's device-name/Ios's UDID. Must be a string.

### Phone.platform

Must be a string, 'android/ios'.

### Phone.version

The phone's system version. Must be a string

### Phone.app

The app's absolute path. Must be a string.

### Phone.install

When set True, the app would be installed/re-installed. Must be a bool.

This property is optional.

### Phone.package

Android's app-package/Ios's BundleID. Must be a string.

### Phone.activity

Android's app-activity. Must be a string.  
This property is only required by Android.

#### `Phone.executor`

Appium's url address. Must be a string.

#### `Phone.log`

The absolute path of where you would like to save the log. Must be a string.

#### `Driver.delay`

The global delay to slow every step down during your test run. Must be an int above 0.  
The unit is millisecond, the default value is 700.

#### `Phone.attach()`

Connect the phone to Appium.

#### `Phone.detach()`

Disconnect the phone from Appium.

#### `Phone.applaunch()`

Launch the app.

#### `Phone.appclose()`

Close the app.

#### `Phone.appreset()`

Reset the app.

#### `Phone.apphide(duration)`

Put the app into background for a period of time.

##### **duration:**

How long you would like to hide the app in the background. Must be an int above 0.  
The unit is millisecond.

#### `Driver.find(attribute)`

Find all the elements who match the attributes you pass.  
Return a list of found `Element` instances.

##### **attribute:**

The attributes that are sufficient enough to identify your desired element. Must be an `Android/ios` instance, or a list of `Android/ios` instances.

#### `Phone.tap(x, y, count = 1)`

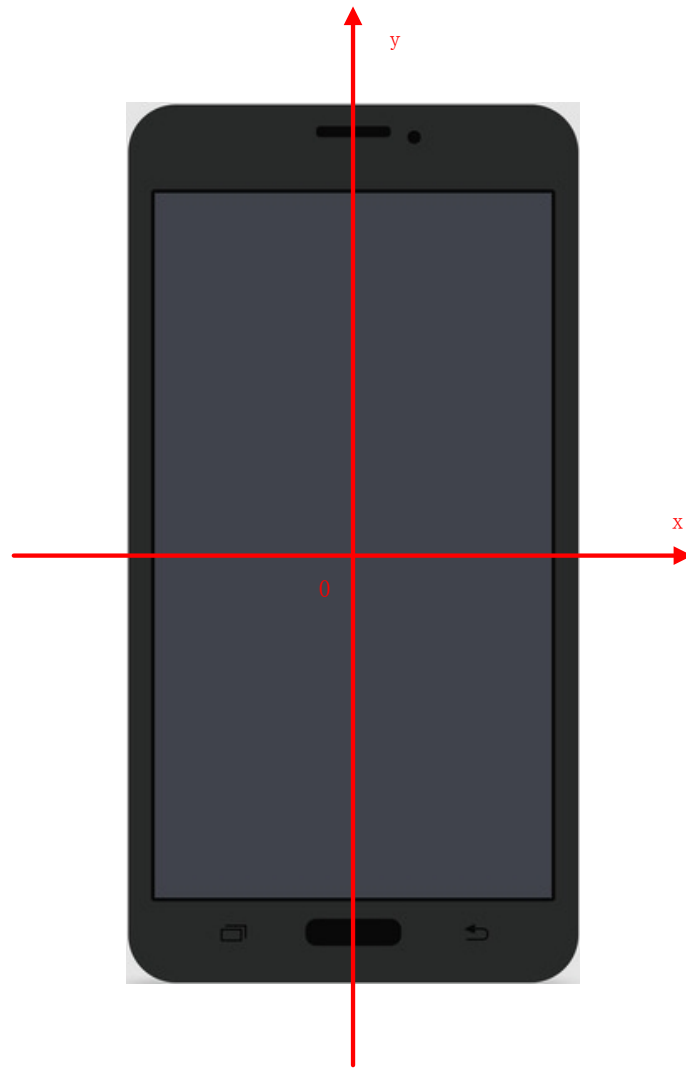
Tap on the very coordinates.

##### **x/y:**

The coordinates you would like to tap on. Must be an int.  
The unit is pixel.

##### **count:**

How many times of tap you would like to perform. Must be an int above 0.  
The default value is 1.



phone screen's origin point & coordinate system

`Phone.hold(x, y)`

Press down without release on the very coordinates.

**x/y:**

The coordinates you would like to press on. Must be an int.

The unit is pixel.

`Phone.release(x, y)`

Move to the very coordinates as the finger keeps pressing down, and then release.

**x/y:**

The coordinates you would like to press on. Must be an int.

The unit is pixel.

`Phone.press(duration, x, y)`

Press down on the very coordinates for a period of time.

**duration:**

How long you would like to keep pressing down. Must be an int above 0.

The unit is millisecond.

**x/y:**

The coordinates you would like to press on. Must be an int.

The unit is pixel.

`Phone.shake(count = 1)`

Shake the phone.

**count:**

How many times of shake you would like to perform. Must be an int above 0.

The default value is 1.

`Phone.locate(latitude, longitude, altitude)`

Set the phone's latitude, longitude & altitude.

**latitude:**

Must be an int/a float,  $\geq -90$  &  $\leq 90$ .

**longitude:**

Must be an int/a float,  $\geq -180$  &  $\leq 180$ .

**altitude:**

Must be an int/a float.

`Phone.width()`

Return the width of the phone screen.

`Phone.height()`

Return the height of the phone screen.

`Phone.topbottom()`

Flick from top edge to bottom edge.

`Phone.bottomtop()`

Flick from bottom edge to top edge.

`Phone.leftright()`

Flick from left edge to right edge.

`Phone.rightleft()`

Flick from right edge to left edge.

`Phone.imefold()`

Fold the keyboard.

`Phone.type_Android(key, meta = None)`

Type key-event.

This definition is Android only.

**Key/meta:**

key-event: <https://developer.android.com/reference/android/view/KeyEvent.html>

meta's default value is None, means not sending key-combine.

`Phone.toast_Android(toast, strict = True, timeout = 7000)`

Return True if the desired toast message were being displayed, otherwise return False.

This definition is Android only.

**toast:**

The toast message or partial toast message you would like to verify.

**strict:**

If strict were True, the desired toast message should be strictly the same as you pass. Or the toast message should be contained in that you pass. Must be a str.

The default value is True.

**timeout:**

The timeout of verifying the toast message. Must be an int about 0.

The unit is millisecond, the default value is 7000.

`Phone.net_Android(net)`

Set the phone's network type.

This definition is Android only.

**net:**

Must be an `Net_Android` instance.

### Definitions in `Element`

`Element.find(attribute)`

Under the current element, find all the elements who match the attributes you pass.

Return a list of found `Element` instances.

**attribute:**

The attributes that are sufficient enough to identify your desired element. Must be an `Android/ios` instance, or a list of `Android/ios` instances.

`Element.parent()`

Return an `Element` instance of the parent node of the current element.

`Element.tap(x = 0, y = 0, count = 1)`

Tap on the current element with offsets.

**x/y:**

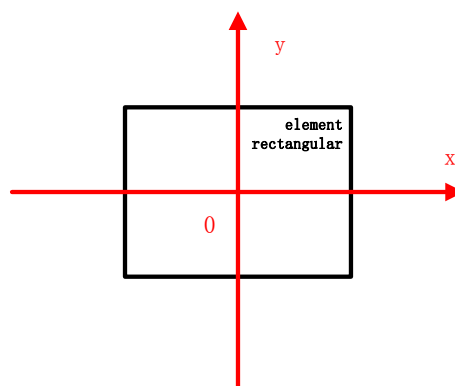
The offsets to the centre of the current element. Must be an int.

The unit is pixel, the default value is 0.

**count:**

How many times of tap you would like to perform. Must be an int above 0.

The default value is 1.



element's origin point & coordinate system

`Element.hold(x = 0, y = 0)`

Press down without release on the current element with offsets.

**x/y:**

The offsets to the centre of the current element. Must be an int.

The unit is pixel, the default value is 0.

`Element.release(x = 0, y = 0)`

Move to the current element with offsets as the finger keeps pressing down, and then release.

**x/y:**

The offsets to the centre of the current element. Must be an int.

The unit is pixel, the default value is 0.

`Element.press(duration, x = 0, y = 0)`

Press down on the current element with offsets for a period of time.

**duration:**

How long you would like to keep pressing down. Must be an int above 0.

The unit is millisecond.

**x/y:**

The offsets to the centre of the current element. Must be an int.

The unit is pixel, the default value is 0.

`Element.clear()`

Clear the content of the current element.

`Element.send(send)`

Send content into the current element.

**send:**

The content you would like to send into the current element. Must be a str.

`Element.waitexist(attribute, timeout = 7000)`

Under the current element, wait until the very element could be found.

**attribute:**

The attributes that are sufficient enough to identify your desired element. Must be an [Android/ios](#) instance, or a list of [Android/ios](#) instances.

**timeout:**

The timeout to wait for the element's appearing. Must be an int above 0.

The unit is millisecond, the default value is 7000.

`Element.waitextinct(attribute, timeout = 7000)`

Under the current element, wait until the very element could not be found.

**attribute:**

The attributes that are sufficient enough to identify your desired element. Must be an [Android/ios](#) instance, or a list of [Android/ios](#) instances.

**timeout:**

The timeout to wait for the element's disappearing. Must be an int above 0.

The unit is millisecond, the default value is 7000.

`Element.width()`

Return the width of the current element.

`Element.height()`

Return the height of the current element.

`Element.abcissa()`

Return the abscissa of the current element to the centre of the phone's screen.

`Element.ordinate()`

Return the ordinate of the current element to the centre of the phone's screen.

`Element.isdisplay()`

Return True if the current element were being displayed, otherwise return False.

`Element.isselect()`

Return True if the current element were being selected, otherwise return False.

`Element.isenable()`

Return True if the current element were enabled, otherwise return False.

### **Definition in [Android](#)**

`Android.class_(class_)`

Return an [Android](#) instance that could be passed to [Phone.find\(\)](#)/[Element.find\(\)](#)/[Element.waitexist\(\)](#)/[Element.waitextinct\(\)](#) as an attribute pattern.

**class\_:**

The class-value of your desired Android element's attribute. Must be a str.

`Android.text_(text_, strict = True)`

Return an [Android](#) instance that could be passed to [Phone.find\(\)](#)/[Element.find\(\)](#)/[Element.waitexist\(\)](#)/[Element.waitextinct\(\)](#) as an attribute pattern.

**text\_:**

The text-value of your desired Android element's attribute. Must be a str.

**strict:**

If strict were True, the desired element should have strictly the same text-value as you pass. Or the very element's text-value should be contained in that you pass. Must be a str.

The default value is True.

`Android.contentdesc_(contentdesc_, strict = True)`

Return an [Android](#) instance that could be passed to [Phone.find\(\)](#)/[Element.find\(\)](#)/[Element.waitexist\(\)](#)/[Element.waitextinct\(\)](#) as an attribute pattern.

**contentdesc\_:**

The content-desc-value of your desired Android element's attribute. Must be a str.

**strict:**

If strict were True, the desired element should have strictly the same content-desc-value as you pass. Or the very element's content-desc-value should be contained in that you pass. Must be a str.

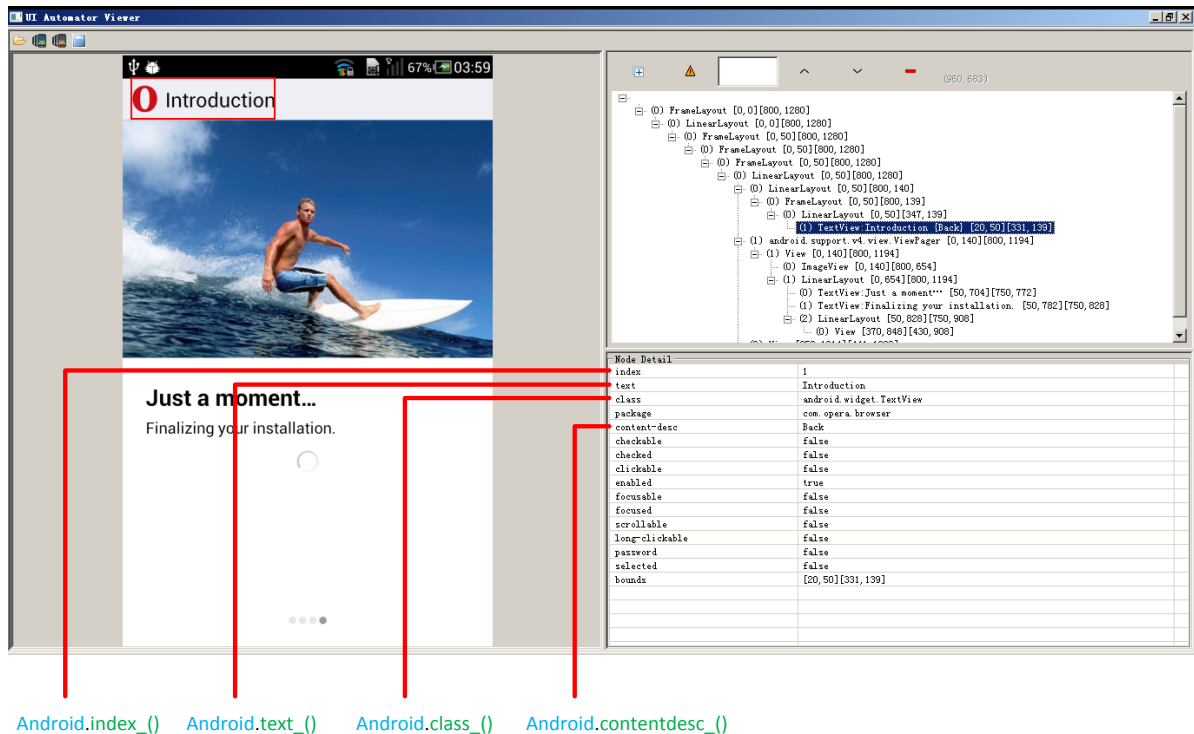
The default value is True.

`Android.index_(index_)`

Return an [Android](#) instance that could be passed to [Phone.find\(\)](#)/[Element.find\(\)](#)/[Element.waitexist\(\)](#)/[Element.waitextinct\(\)](#) as an attribute pattern.

**index\_:**

The index-value of your desired Android element's attribute. Must be a int  $\geq 0$ .



Android's class, text, content-desc & index

### Definition in *los*

*los.type\_(type\_)*

Return an *los* instance that could be passed to *Phone.find()/Element.find()/Element.waitexist()/Element.waitextinct()* as an attribute pattern.

*type\_*:

The type-value of your desired *los* element's attribute. Must be a str.

*los.name\_(name\_, strict = True)*

Return an *los* instance that could be passed to *Phone.find()/Element.find()/Element.waitexist()/Element.waitextinct()* as an attribute pattern.

*name\_*:

The name-value of your desired *los* element's attribute. Must be a str.

*strict*:

If *strict* were True, the desired element should have strictly the same name-value as you pass. Or the very element's name-value should be contained in that you pass. Must be a str.

The default value is True.

*los.label\_(label\_, strict = True)*

Return an *los* instance that could be passed to *Phone.find()/Element.find()/Element.waitexist()/Element.waitextinct()* as an attribute pattern.

*label\_*:

The label-value of your desired *los* element's attribute. Must be a str.

*strict*:

If *strict* were True, the desired element should have strictly the same label-value as you pass. Or the very element's label-value should be contained in that you pass. Must be a str.

The default value is True.



`ios.value_(value_, strict = True)`

Return an `ios` instance that could be passed to `Phone.find()/Element.find()/Element.waitexist()/Element.waitextinct()` as an attribute pattern.

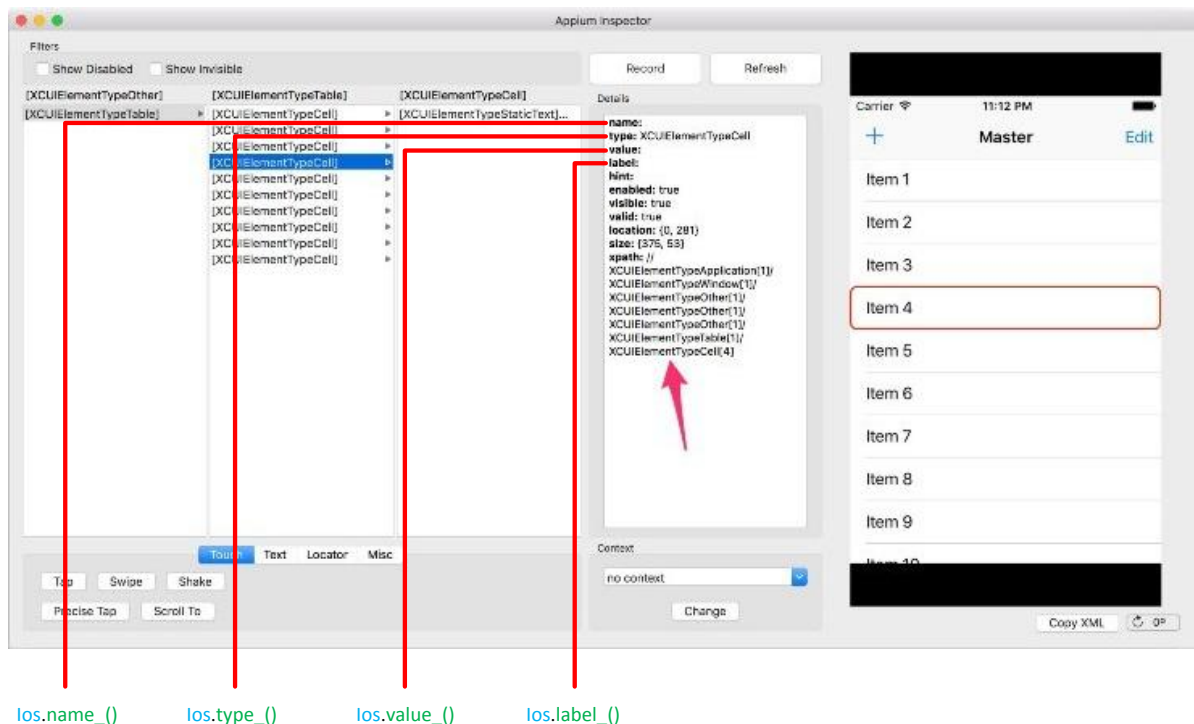
`value_:`

The value-value of your desired ios element's attribute. Must be a str.

`strict:`

If `strict` were `True`, the desired element should have strictly the same value-value as you pass. Or the very element's value-value should be contained in that you pass. Must be a str.

The default value is `True`.



ios's type, name, label & value