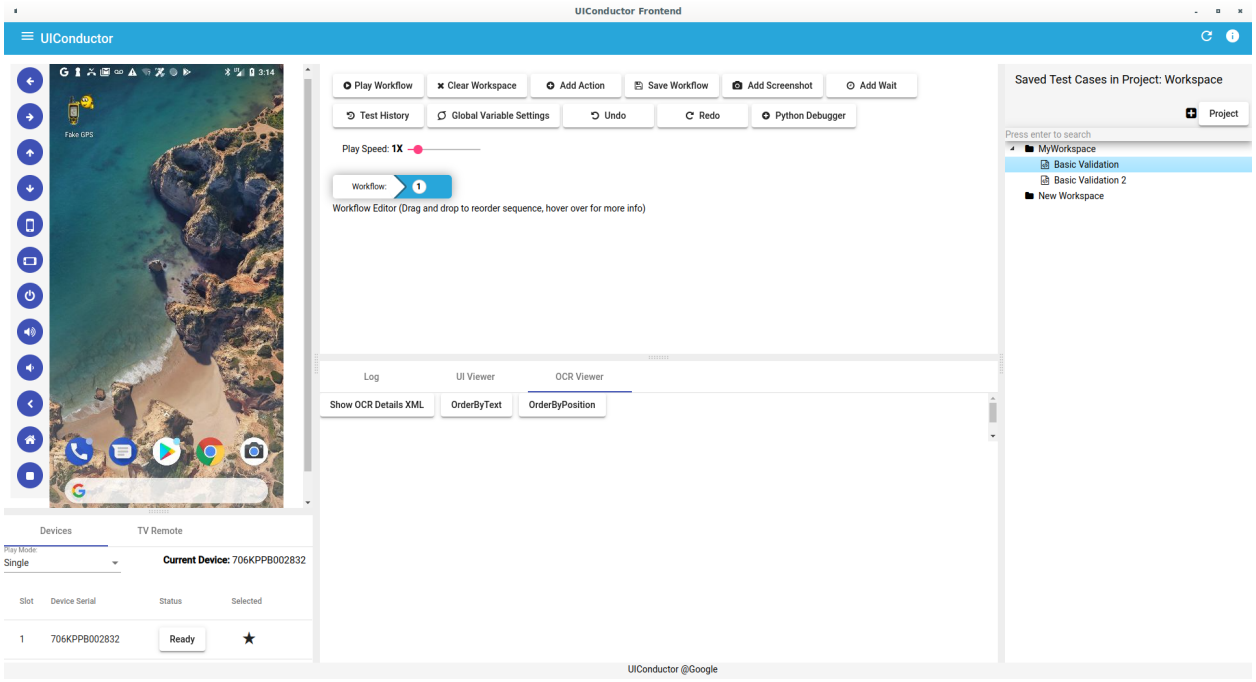
UIConductor User Guide

Android UIConductor

Android UIConductor is a platform that allows users to create E2E testing workflows very easily.

Screen Shot



Requirement

- Linux (Mac and Windows should also work, but need write the package and start script for each platform)

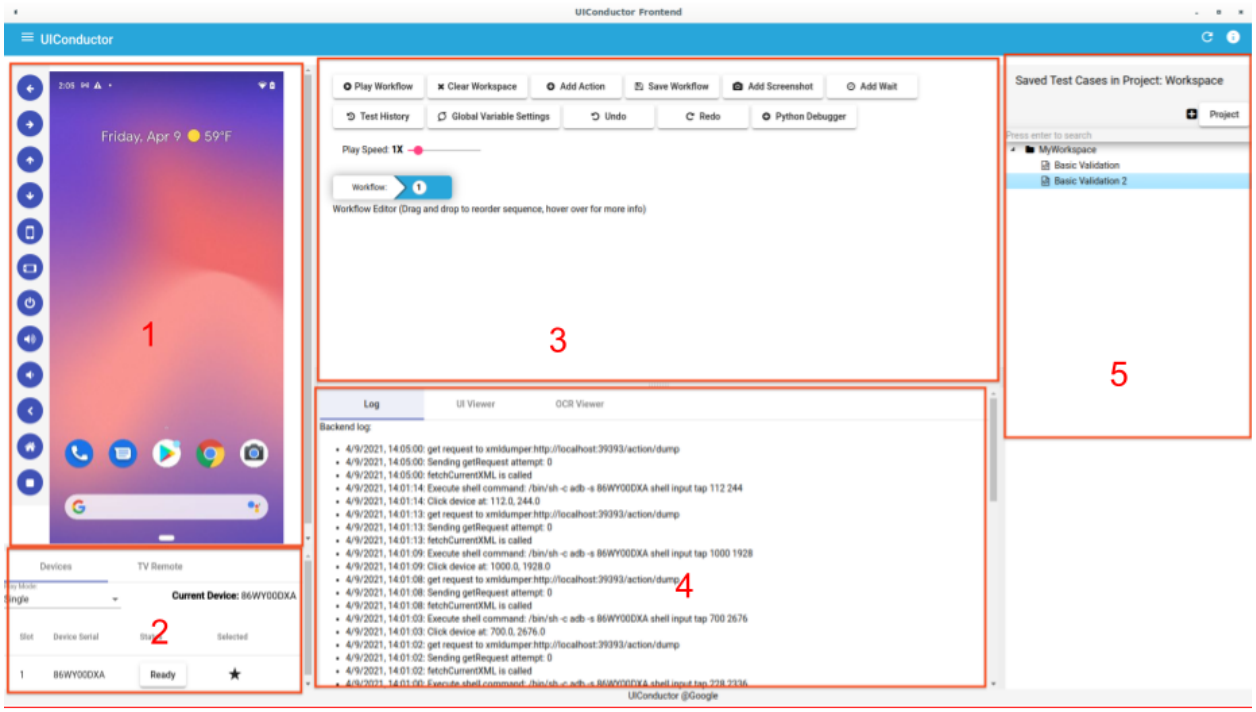
- Jdk 8+
- Android SDK 28
- adb installed and setup correctly in the env
- a local/remote MySql Database
- Angular CLI: 1.7.4 (Currently the higher verison of Angular CLI might not work. we are working ont the migration)
- Node: 10.15.0

Installation

1. Create an empty database schema called "uicddb"
2. Execute backend/src/com/google/uicd/backend/recorder/db/initdb.sql to init the database schema.
3. Set the ANDROID_HOME to your SDK location (you can skip step3 and step4 if you want to use the prebuild one)
4. run ./package.sh to build everything into the release folder
5. Set the db connection string in the release/uicd.cfg to something like this: jdbc:mysql://localhost:3306/uicddb?autoReconnect=true&user=root&password=admin&useUnicode=true&characterEncoding=utf-8 Please make sure change the username and password to your own username and password.

Basic Usage

1. cd release
2. ./start.sh to start the application. Please make sure the following things before run start.sh
    a. adb devices are in the right state
    b. No password or lock pattern for the devices.

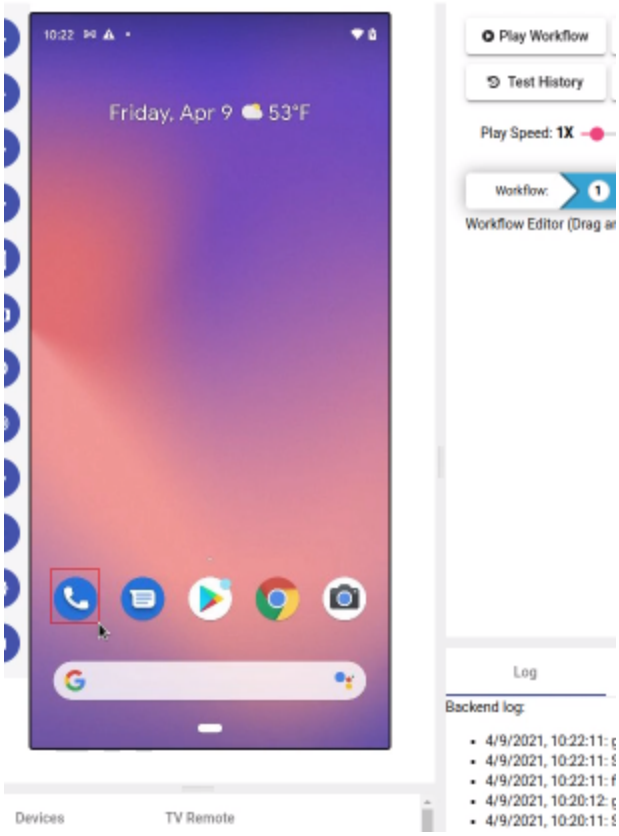


Section 1: Real Time streaming of a phone connected. Users can do actions, like ClickAction, SwipeAction etc. in this section. Includes the preset buttons such as swipe left/right, swipe up/down, etc.
Section 2: Contains devices and simulator of TV Remotes. Users can switch devices and it will show the real time streaming device in section 1.
Section 3: Test case edit section. Actions in a test case are editable here.
Section 4: Log section, UI XML tree viewer and OCR Viewer.
Section 5: Test case management section.

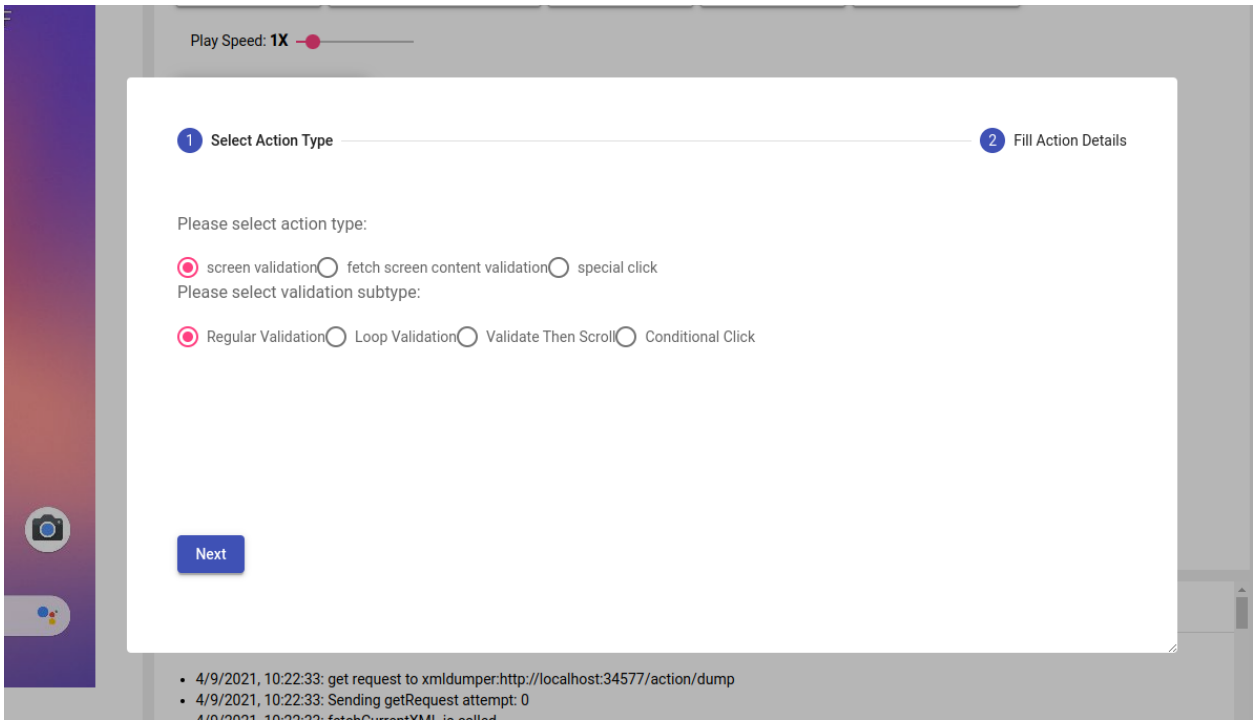
Basic Workflow and Validation

Make sure the Android device is connected to the computer and open UICD.

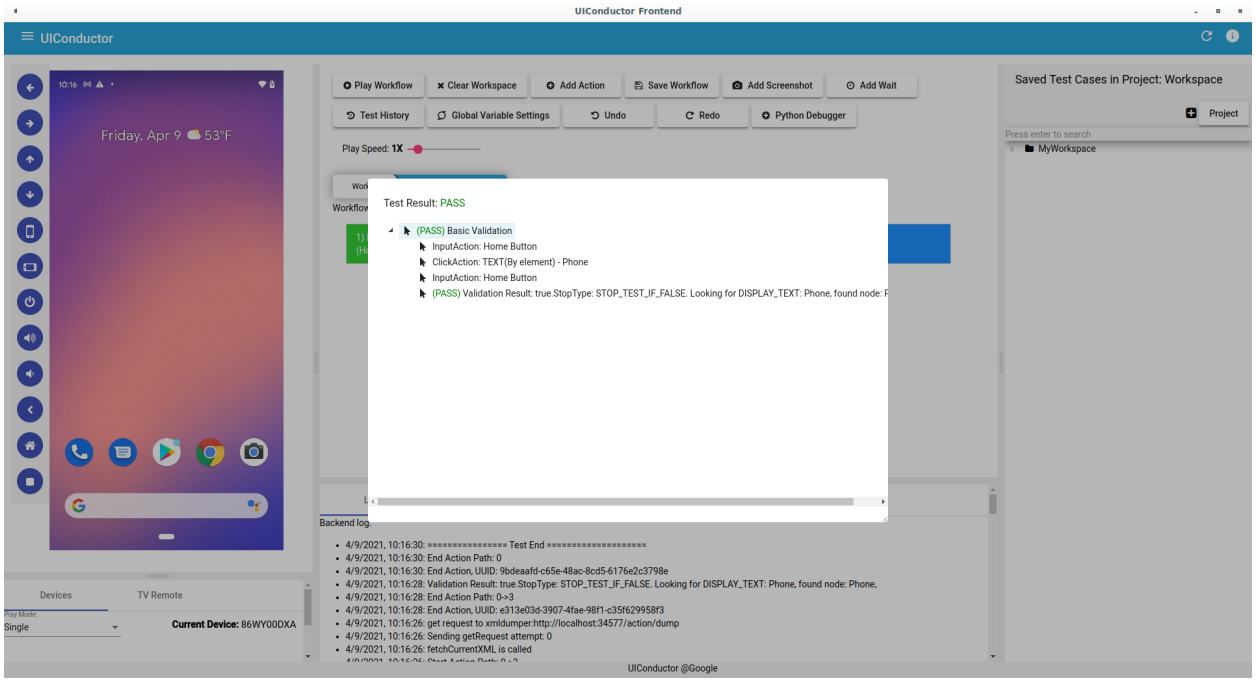
1. Click Home on the pre-made buttons to the left. 
2. Click Phone Icon on the home page using the mouse.
3. Click Home on the pre-made buttons to the left.
4. CTRL+ Mouse select phone icon on home screen. Here we are validating if the phone icon is on the screen.



5. Window Pops up. Screen Content Validate and Regular Validation is selected.



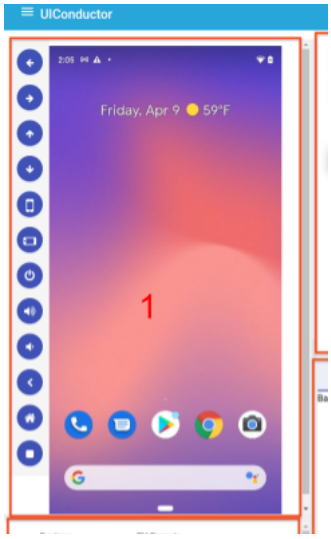
6. Click Next.
7. Click Play workflow.
8. Workflow should pass as it is validating “Phone”.



Basic Mouse Actions

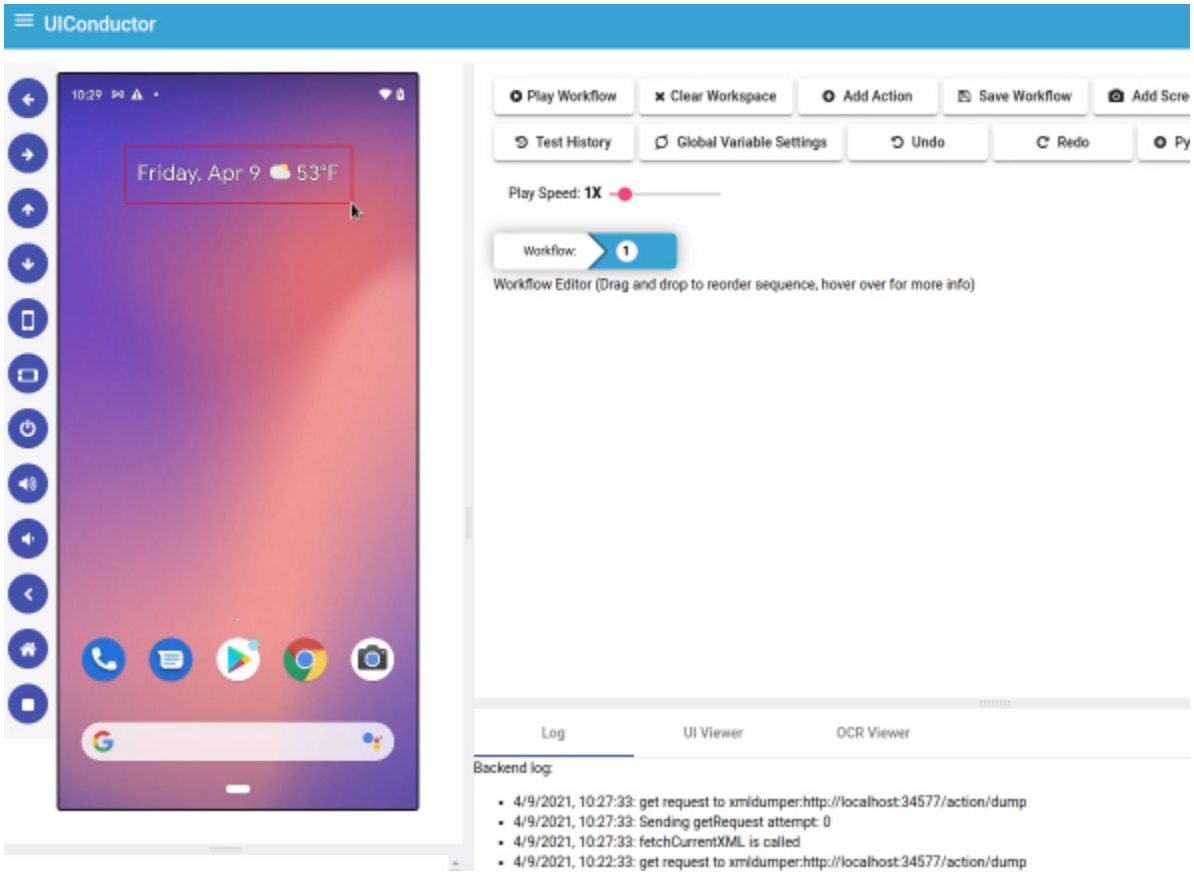
The mouse can be used to mimic basic actions on the UI Conductor.

Swipe - Hold down the left mouse button and swipe in a direction.
Long Click - Hold down the left mouse button.
Click - Press the left mouse button

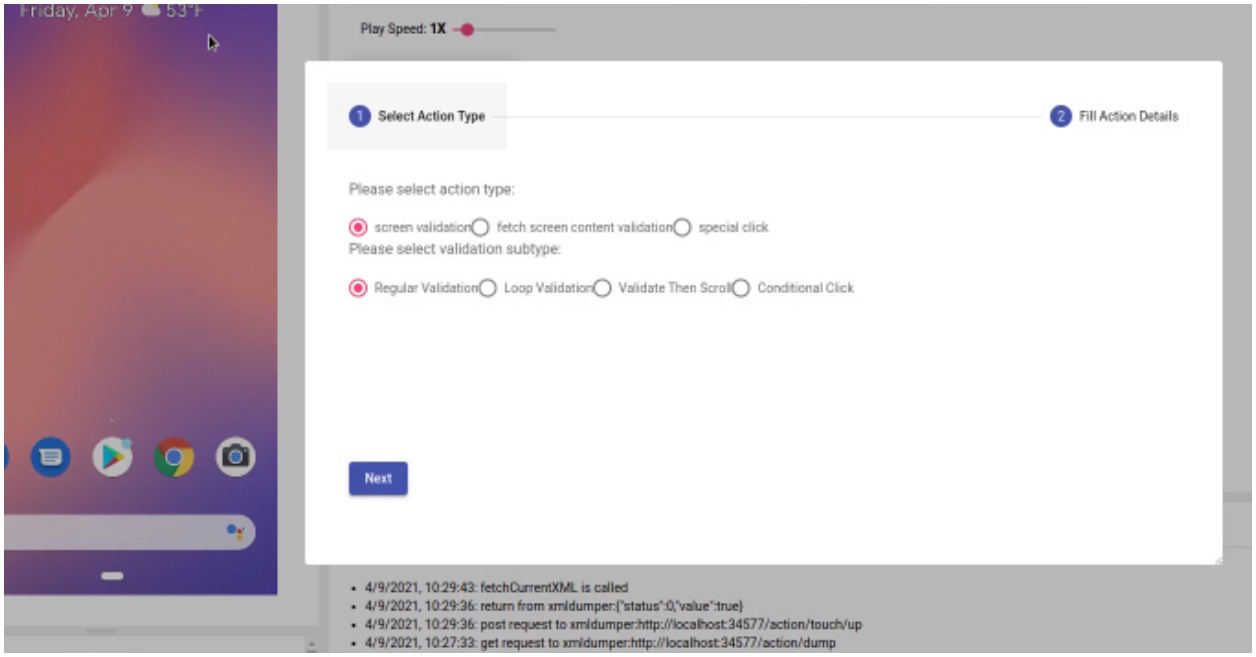


Validations

CTRL (Command on Mac) + Mouse Left Click to select area to validate.



A validation pop up will show.



Regular Validation

This is used to check whether the selected element is present on the screen or not while playing back the test case.

Loop Validation

Loop Validation is like Regular Validation, except that it will repeat the check multiple times. This is useful if the text you need to validate may not show up immediately. A timeout can be given to specify how long to wait before failing the validation. Notifications is a good example when to use loop validation.

Validate Then Scroll

Sometimes there may be an element that is outside the display's view you want to validate. This validation will attempt to scroll in a given direction to find the element.

Conditional Click

This is used to click on an element only if the validation being done is true. For example, you only want to click on the WiFi button if it is currently off, otherwise you do not want to click on it.

Note that Conditional Click will not report any failures if the validation is not true. It will only skip the action and proceed with the rest of the test case.

Validation Option Details

Validation Option Details

Match Node Context

Type	Description
True	Match Context(UI hierarchy based match, search the element not only by the text, but also by the element around it). Useful when doing the condition click.
False	Directly match the target text.

Search Range

Type	Description
Strict	Search for text within a 100px radius on the screen (Physical device resolution).
Around	**Default value**. Search for text within a 300px radius on the screen (Physical device resolution).
Full Screen	Search for text on the entire screen. Note: Only use this if the text is unique enough to only appear once on the screen.

Stop Type

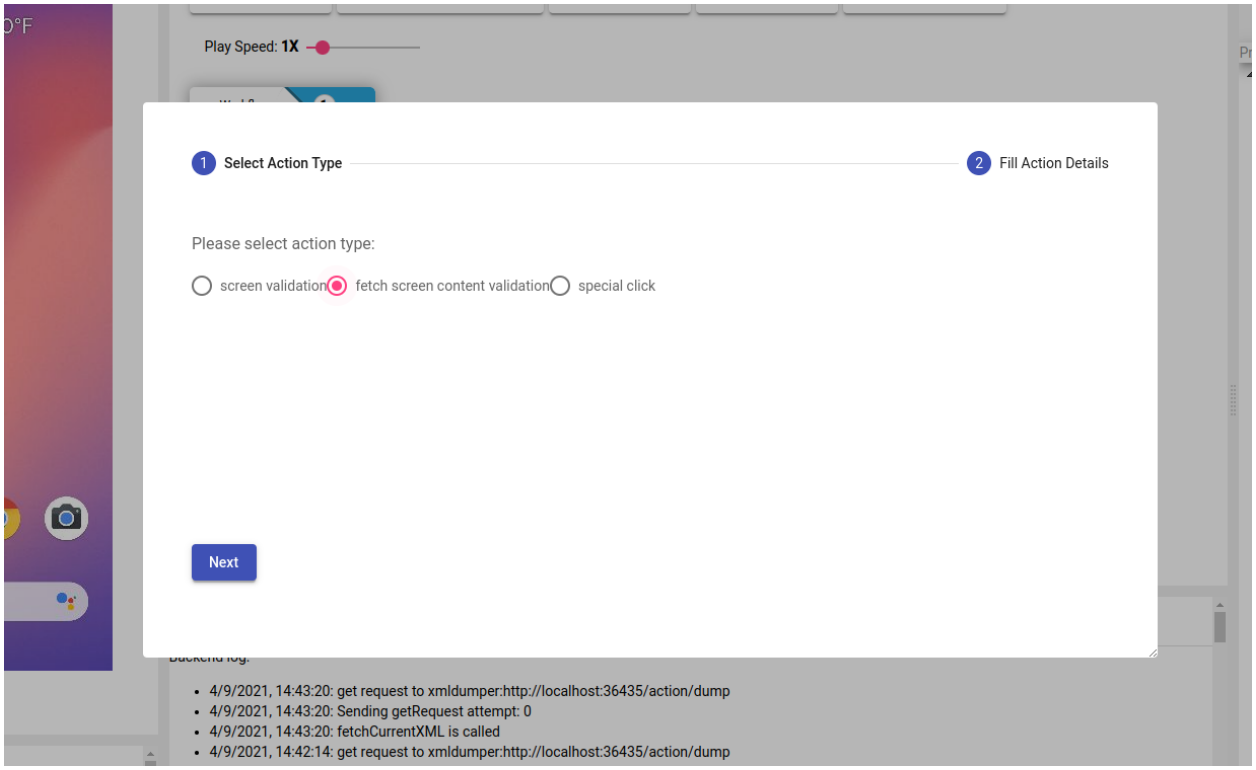
Type	Description
Stop All If False	Stop and fail the test if the validation is false.
Stop All If True	Stop and fail the test if the validation is true.
Stop Current Compound If False	Break the current compound action if the validation is false but continue test sequence.
Stop Current Compound If True	Break the current compound action if the validation is true but continue test sequence.

*More details on nuwa userguide

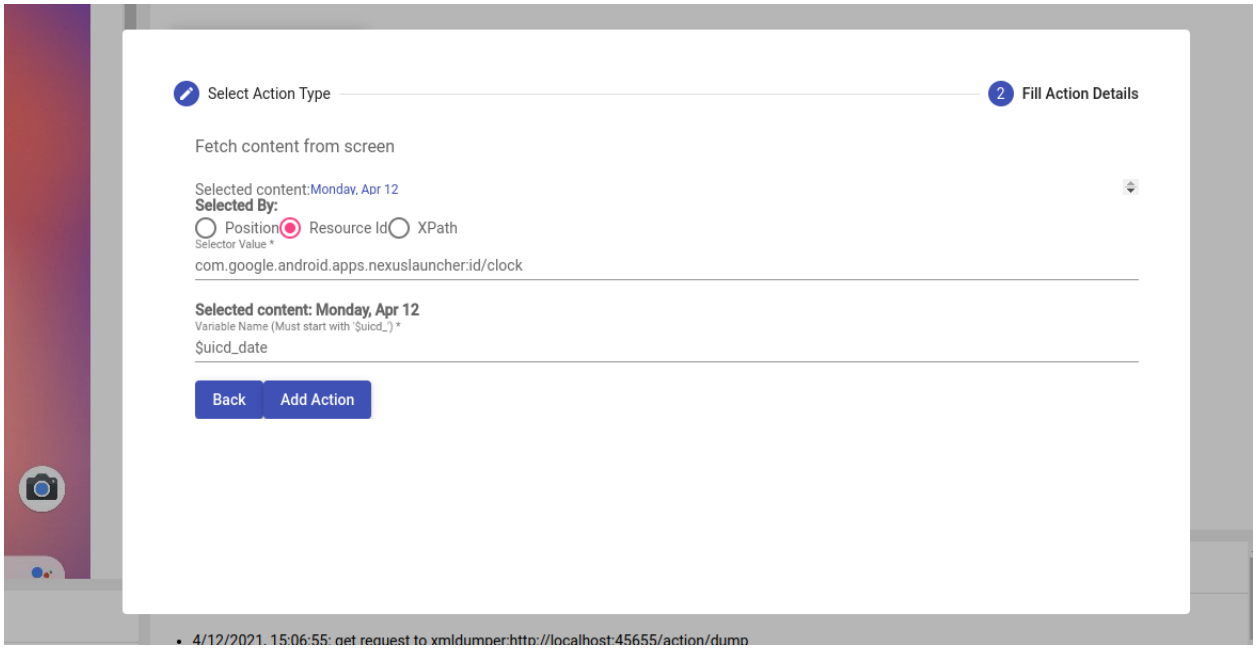
OK

Fetch Screen Content Validation

CTRL (Command on Mac) + Mouse Left Click to select area to fetch screen content.

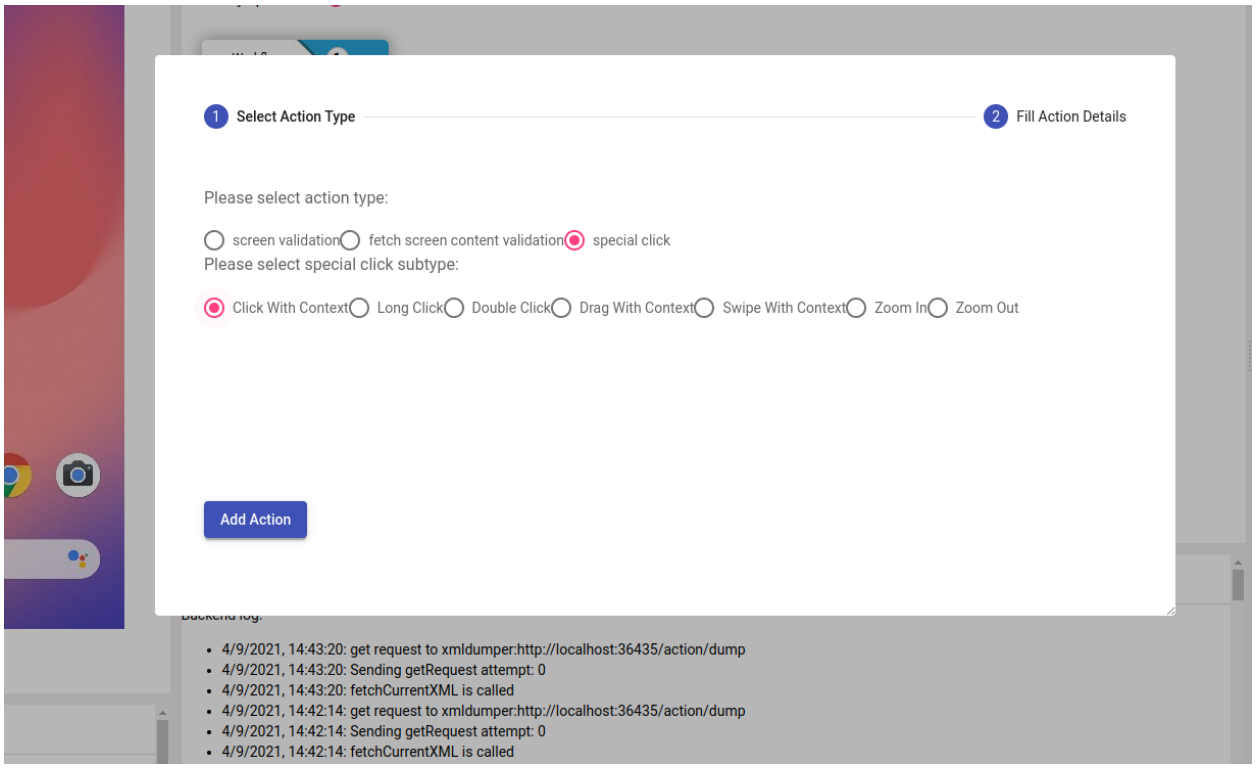


Fetch screen content validation fetches the screen content and creates a global variable. Variables should always start with $uicd_<variable>



Special Click

CTRL (Command on Mac) + Mouse Left Click to select area use a special click.



Click with Context

An action to click on an item with specific context.

Long Click

An action to perform a long click.

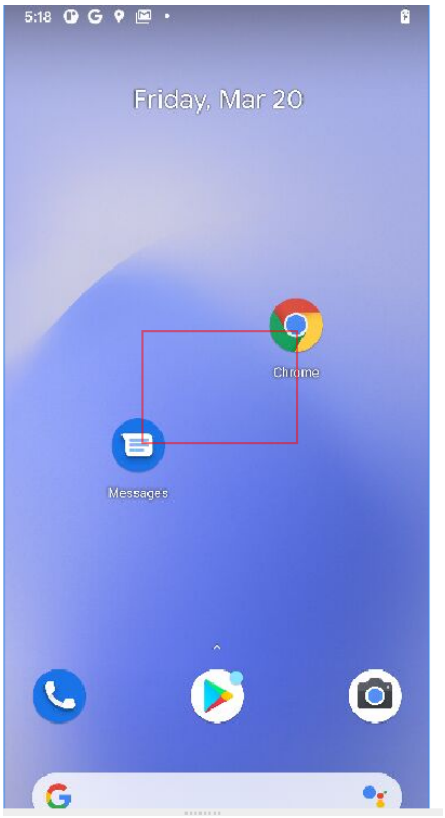
Double Click

An action to perform double click.

Drag with Context

Drag with context action drags an icon to another icon. To use this action press ctrl + mouse to make a square. One corner touching the first icon and the diagonal corner from it touching the second icon.

The action type window pops-up. Choose special click and Drag With Context.

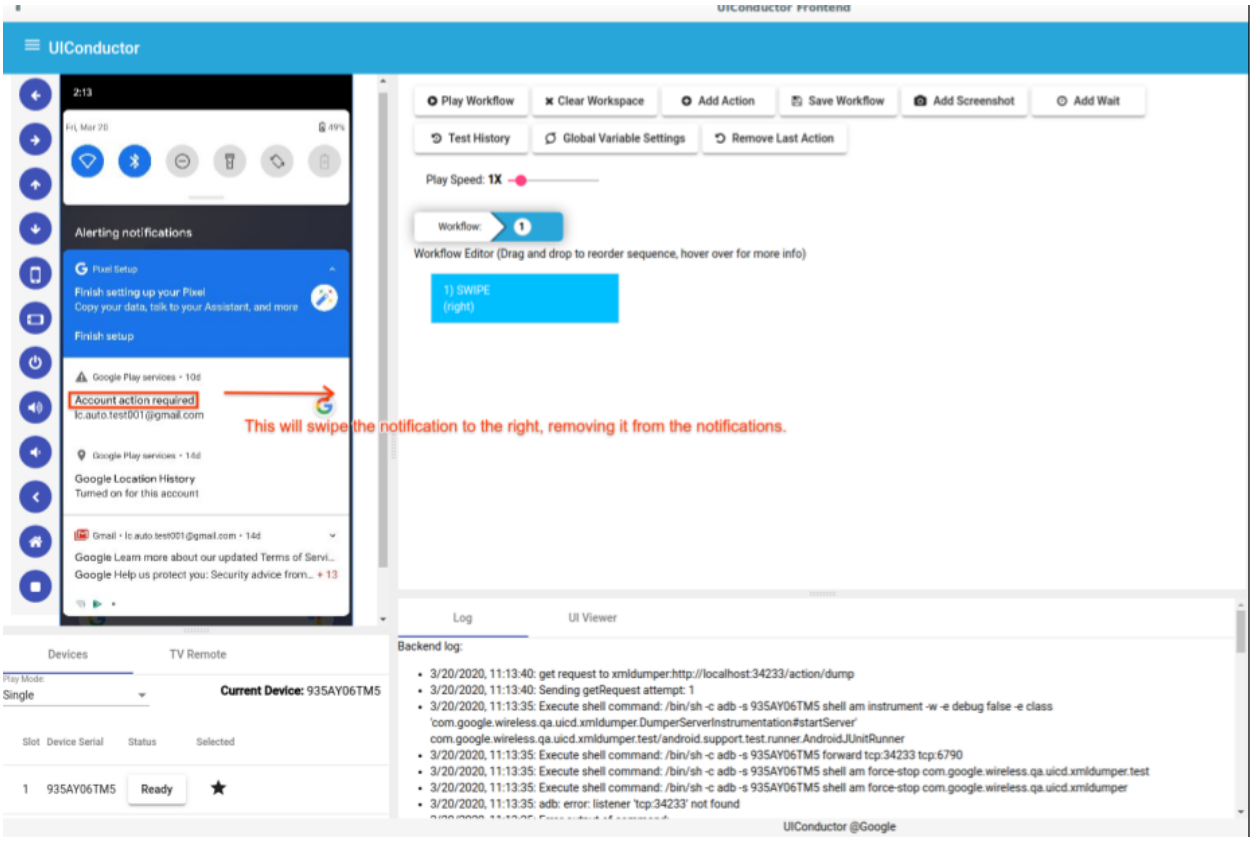


End result.

Swipe with Context

The swipe with context action swipes right on the context a user chooses.

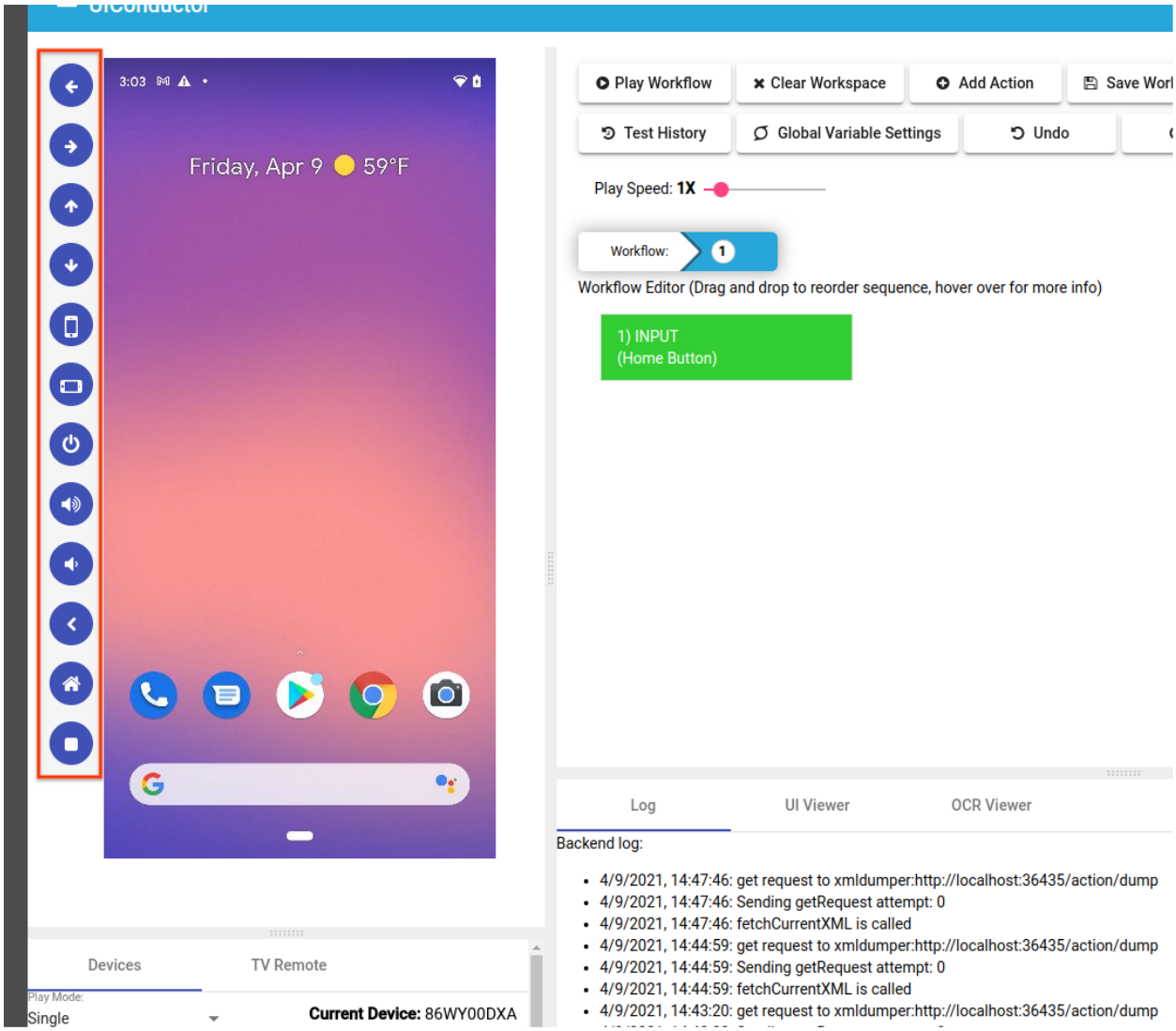
A good example when to use this action is on notifications. In this example "Account action required" is selected. When playing the action UICD will swipe right on the context, removing it from the notifications.



Zoom In/Zoom Out

Zoom In and Zoom out can be performed using a special click.

Preset Buttons

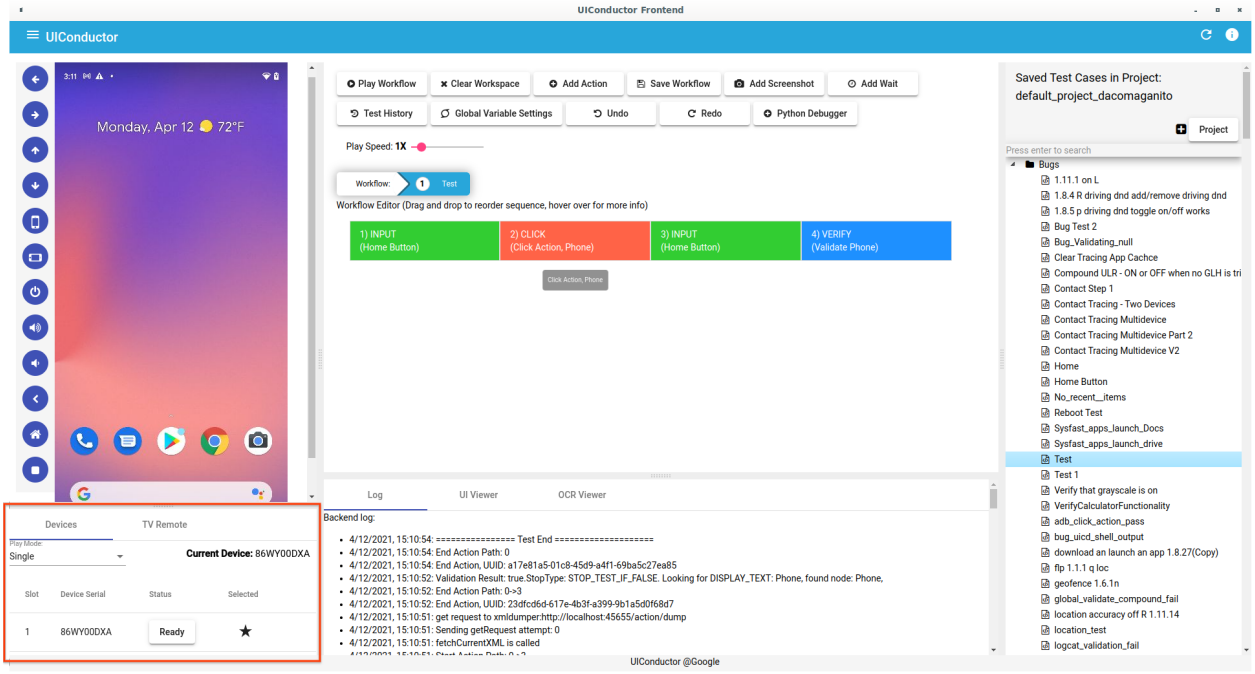


In order from top to bottom.

- Swipe Left
- Swipe Right
- Swipe Up
- Swipe Down
- Portrait Mode
- Landscape
- Power
- Volume Up
- Volume Down
- Home
- Overview

It is recommended to use the preset buttons first if it fits the use case. For example some devices have a physical home button so using the preset home button is better. Also if the device has a home button on the UI it is recommended to use the preset home button.

Devices and TV Remote



Devices

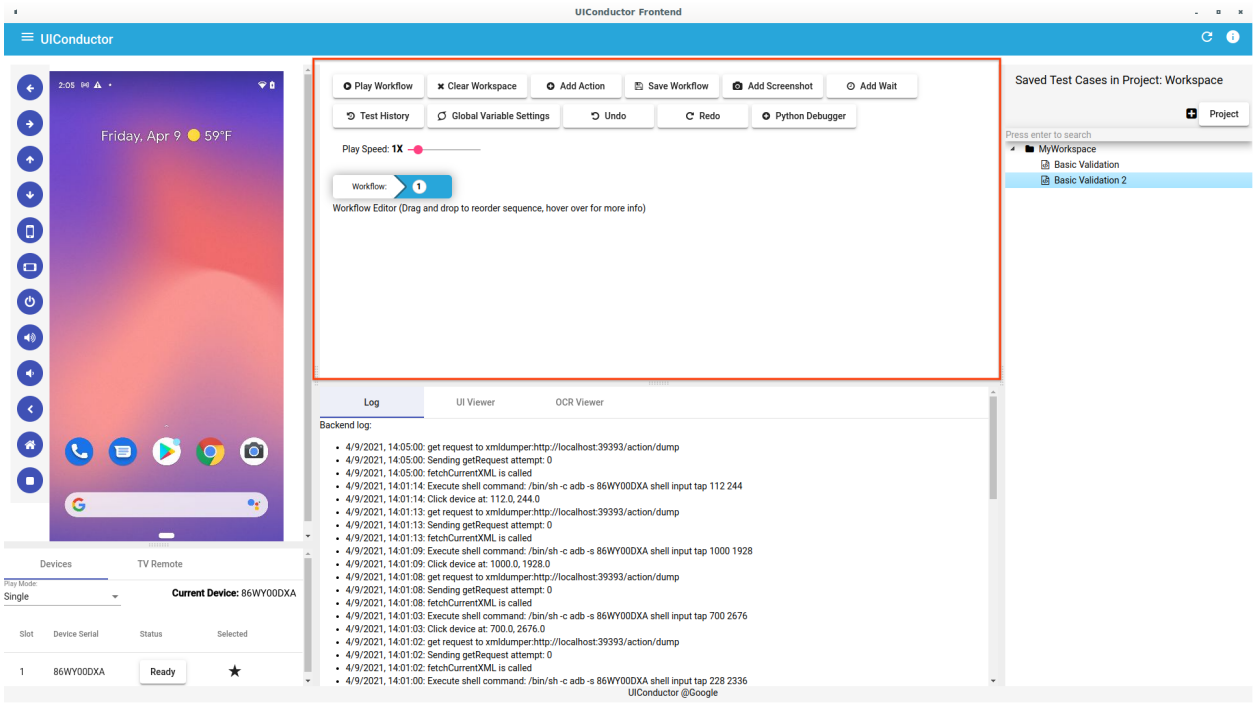
Play Mode

Single - Plays actions on a single device.
Multidevice - Plays actions on multiple devices. Interactions between two or more devices. For example if you want to check if a sms was sent to another device use this.
Play All - Plays actions simultaneously across all connected devices.

TV Remote

Simulator for remote tv.

Test Case Edit Section



Play Workflow

Plays the workflow in the workflow editor.

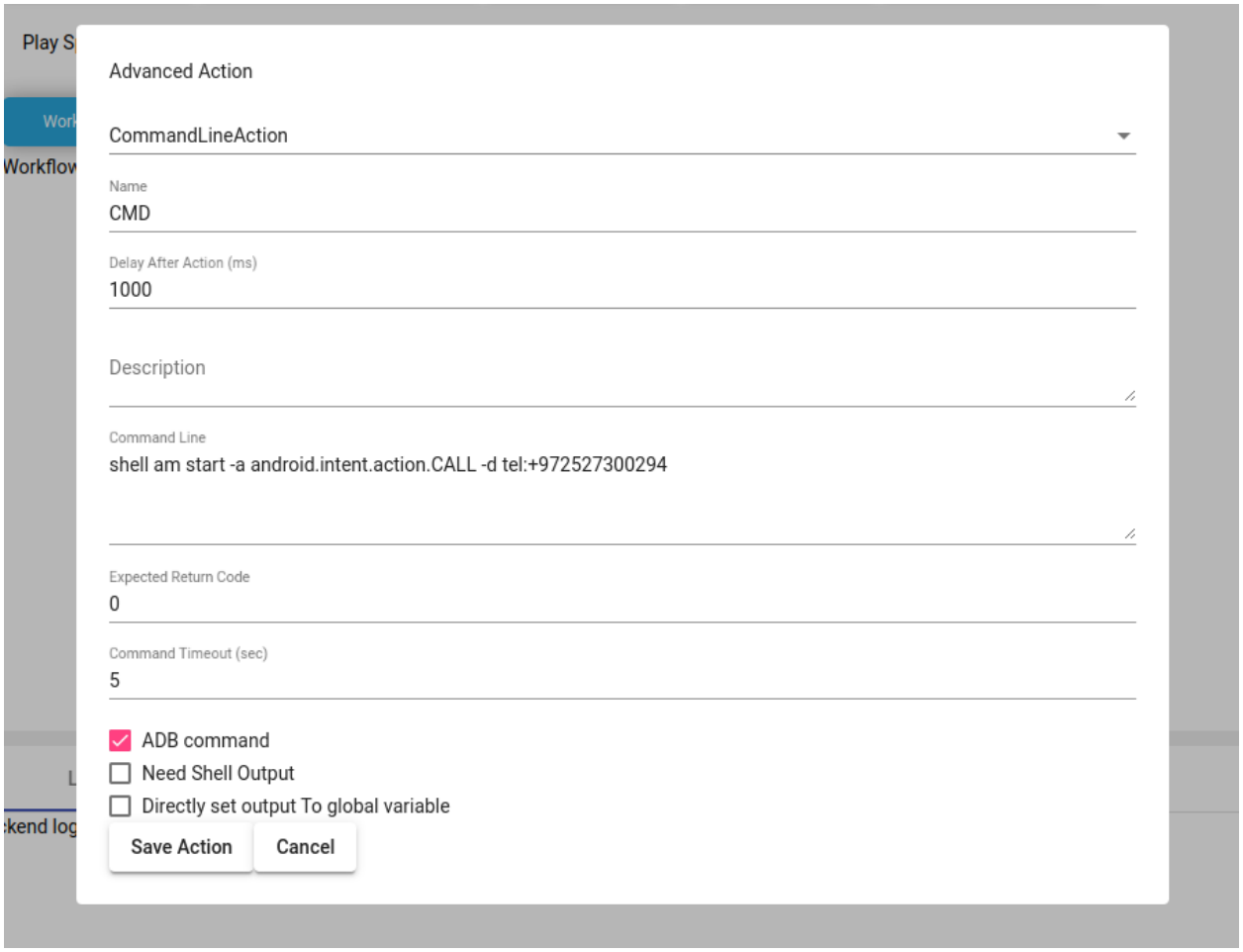
Clear Workspace

Clears the workspace in the workflow editor.

Add Action

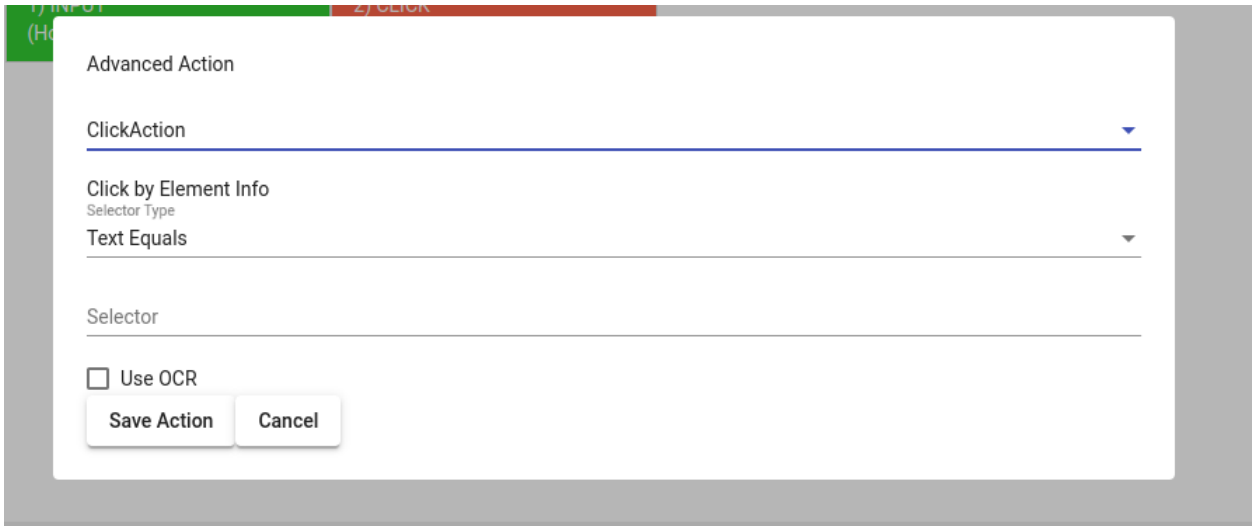
Command Line Action

Here we can make command line actions. For adb commands click the adb command checkbox and add the command in the command line. Example in the screenshot below.



Click Action

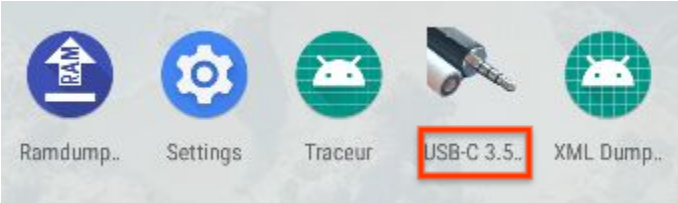
Click action is used to be more precise about the correct option to click. There are four kinds of options for selecting using the advance click actions. They are Text Contains, Text Equals, Resource ID, and Xpath.



Text Contains

In this example we use Text Contains to click on “USB-C 3.5..”.

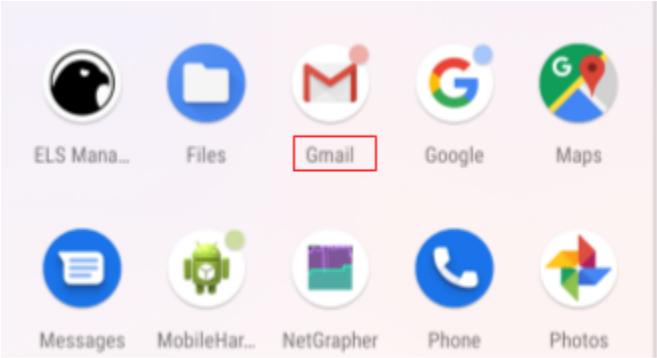
```
USB-C
```



Text Equals

In this example we use Text Equals to click on Gmail. This is one of the multiple ways to use a click text action.

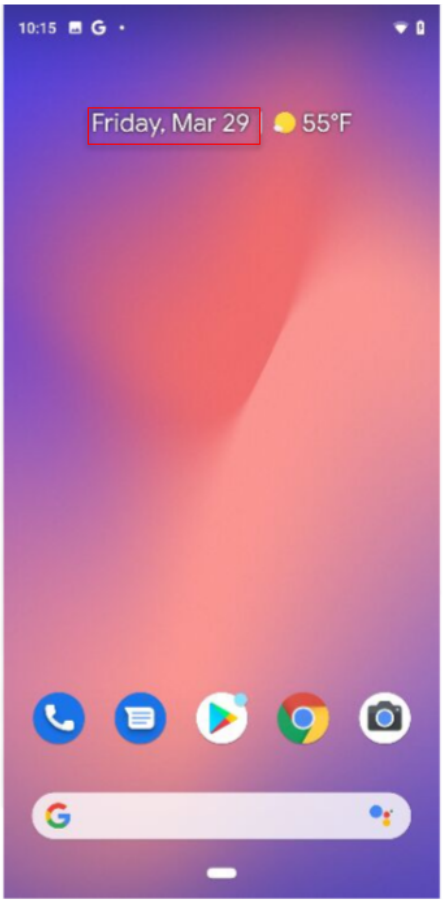
```
Gmail
```



Resource ID

In this example we use resource id to click the date. It is good practice to use this, especially if there is no text visible for the source we want to click. Uicd will automatically find the resource id on the screen and click it.

```
com.google.android.apps.nexuslauncher:id/clock
```



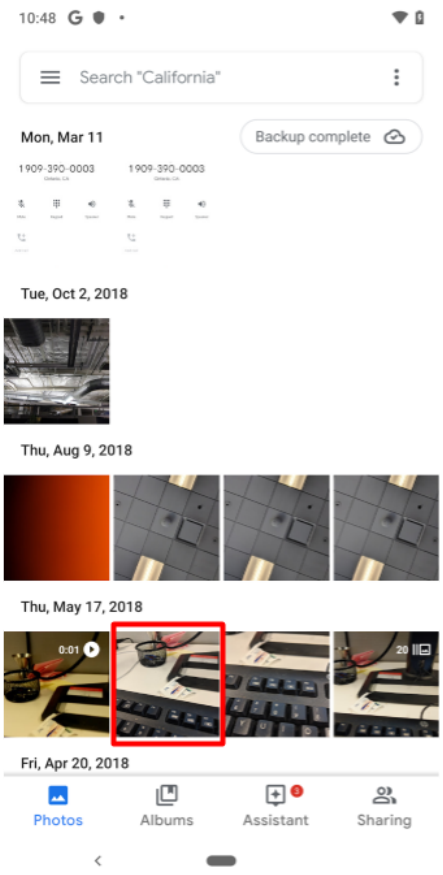
Xpath

In this example we want to click the second photo under "Thu, May 17, 2018". It is impossible just using text or the resource id. In this case XML will be very useful. Using Xath is more dynamic compared to using resource id or text clicks.

```
//*[@resource-id='com.google.android.apps.photos:id/photos_photogrid_date_scrubber_view']//*[@class='android.view.ViewGroup' and @index='12']
```

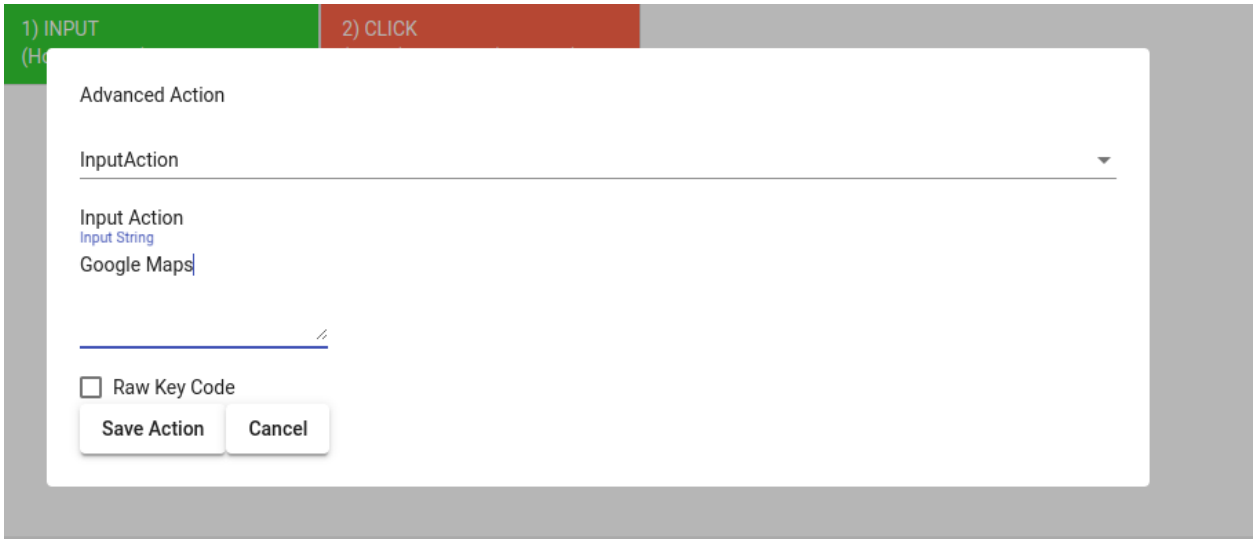
OR

```
//*[@resource-id='com.google.android.apps.photos:id/photos_photogrid_date_scrubber_view']//androidviewViewGroup[@index='12']
```



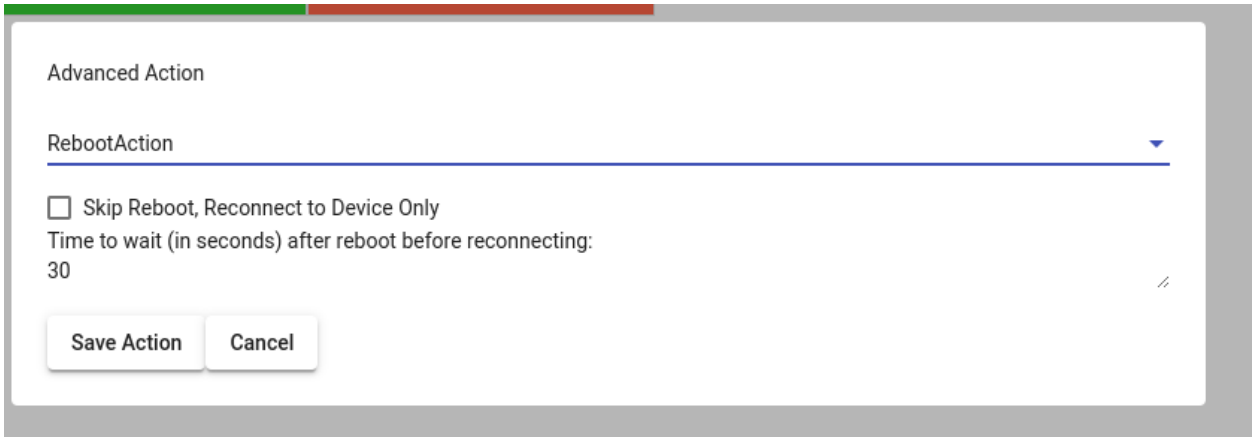
Input Action

The purpose of this action is to provide long input strings in the required fields instead of individual keyboard click, hence reducing the number of test steps.

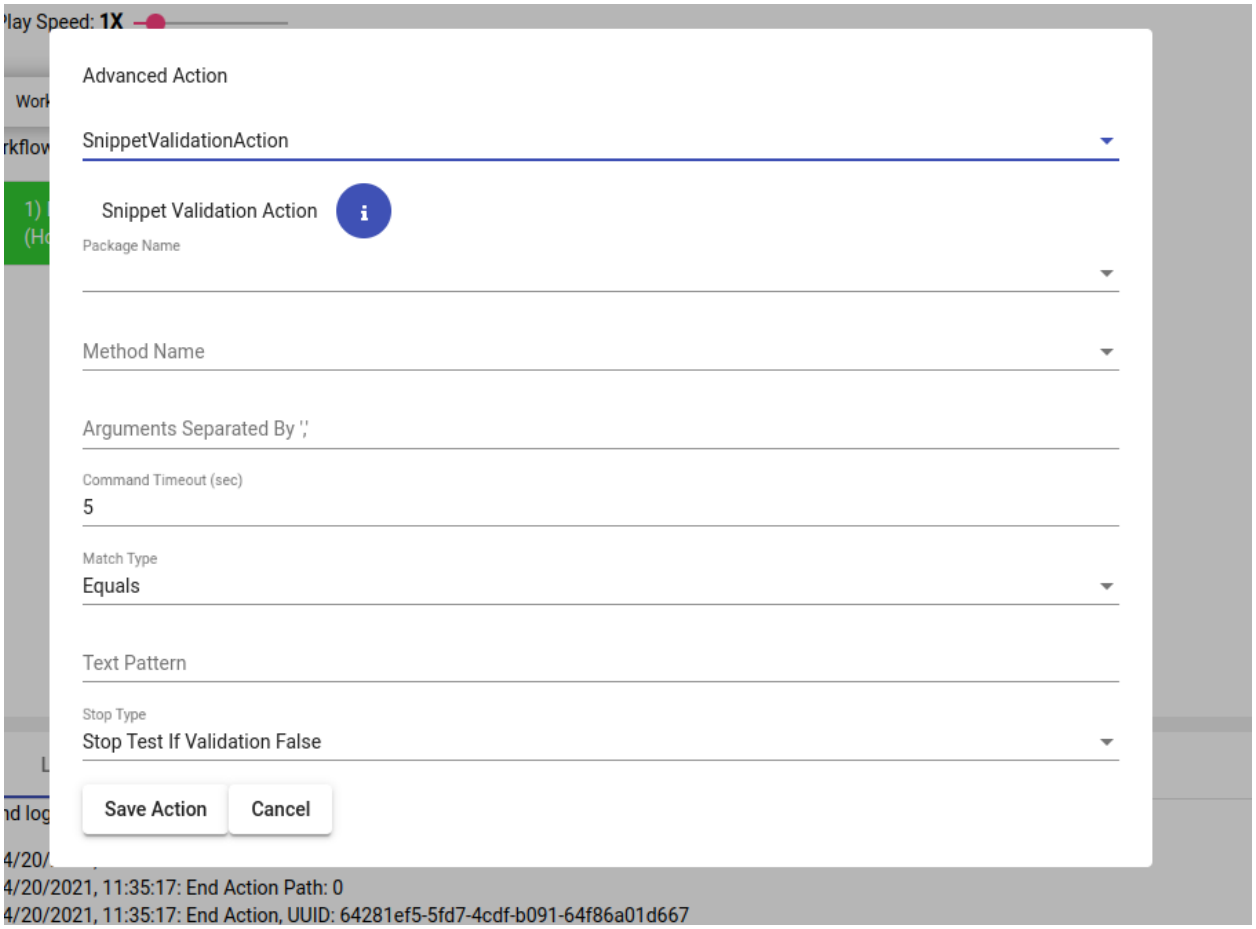


Reboot Action

As the name suggests, this action is used to perform a reboot action in the test. Some of the work flows need a reboot of the device in between, So this action can help the user to restart the device in such conditions.



Snippet Validation Action



Script Execution Action

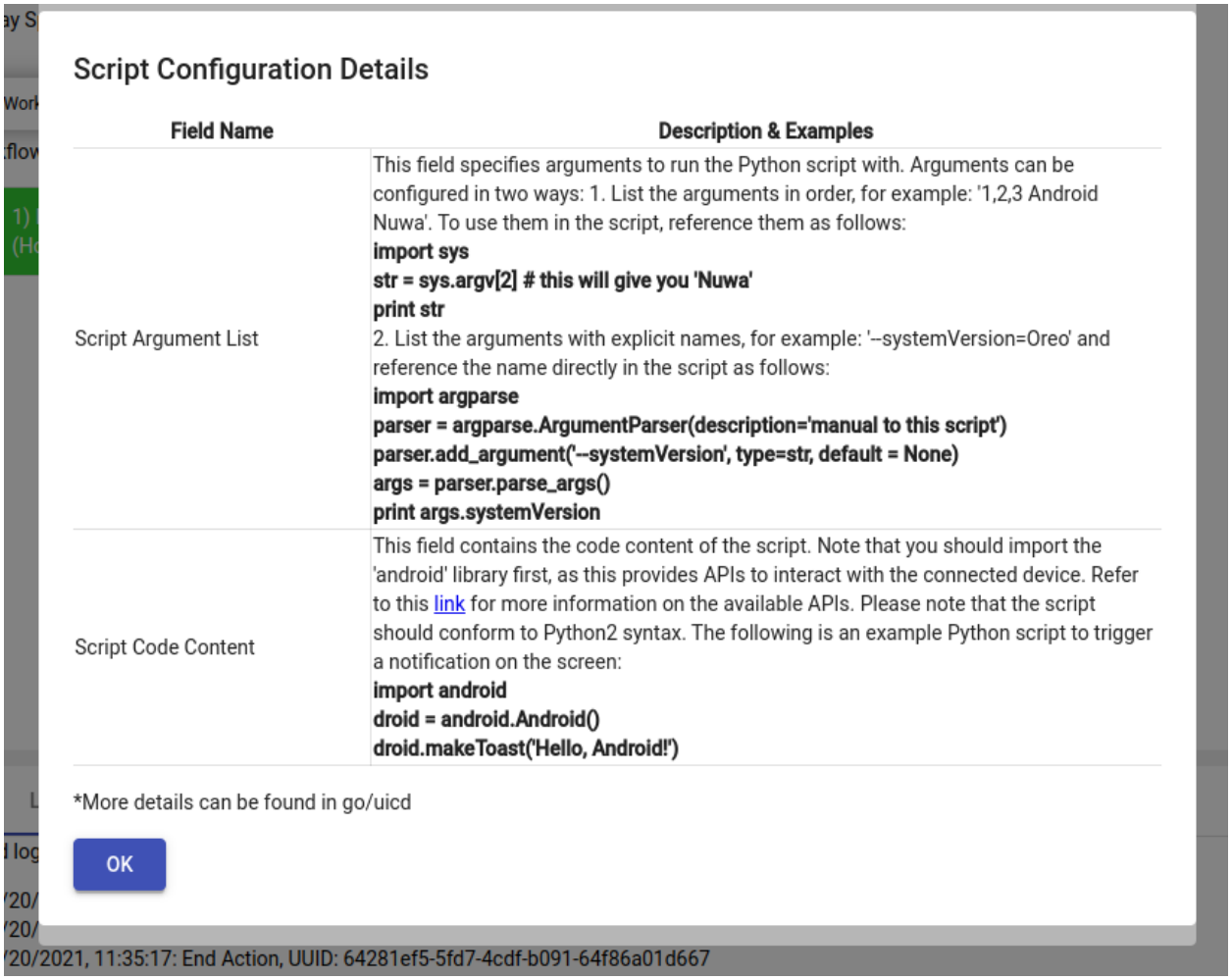
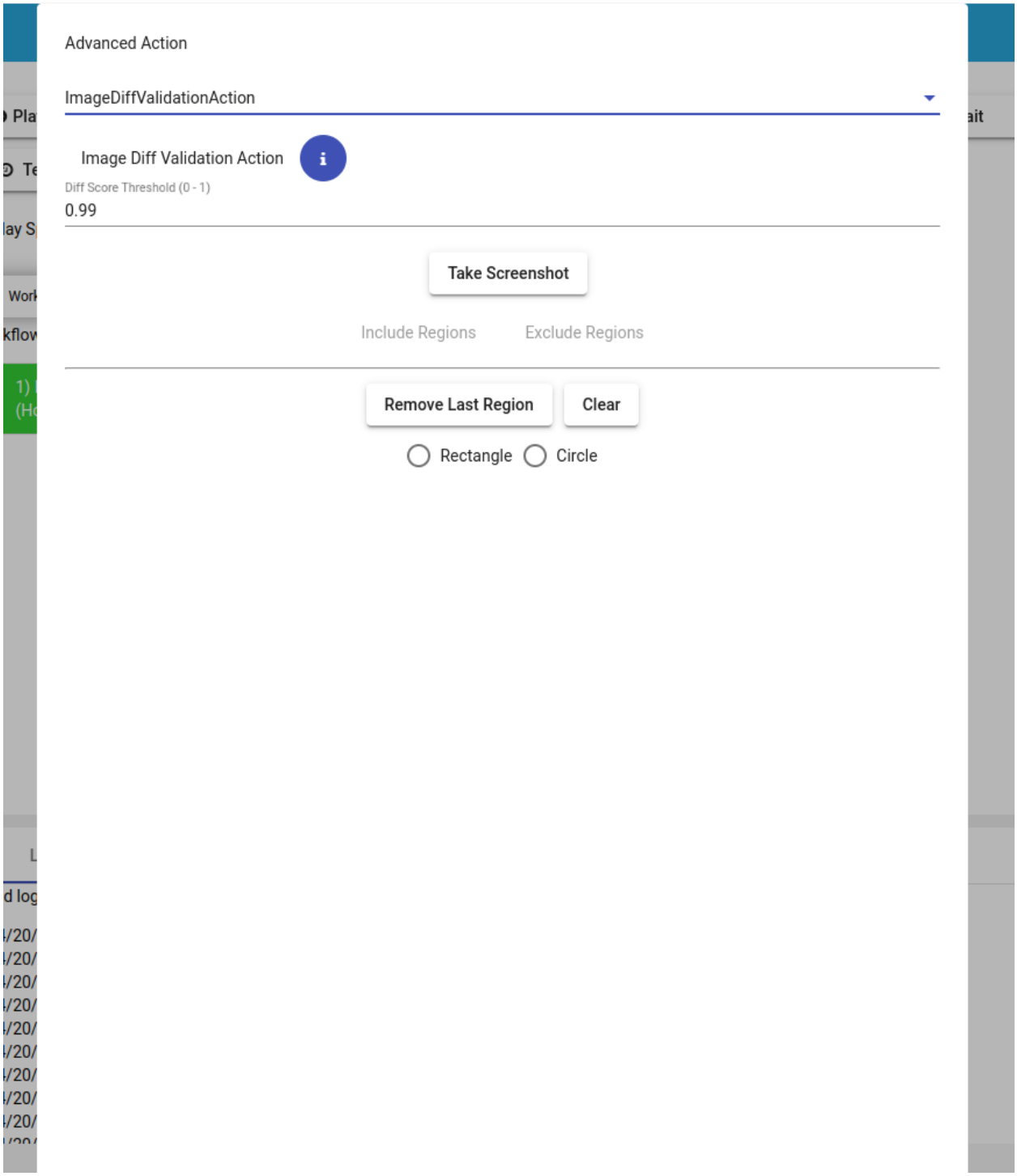


Image Diff Validation Action

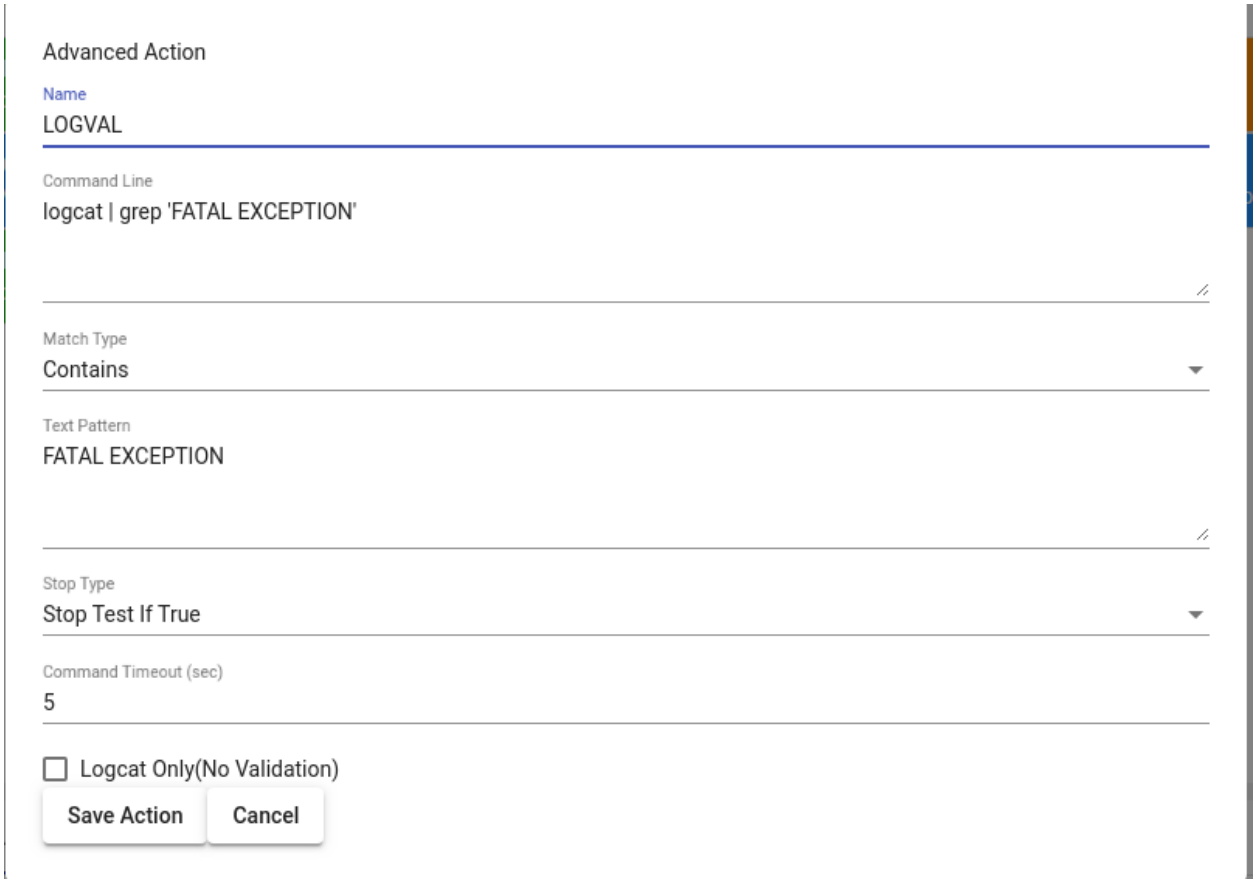
Validate an image.



Logcat Validation Action

Logcat validation is used when validation is needed on the logcat or dumpsys. The logic on the logcat validation treats each line as one string.

For example if searching for 'FATAL EXCEPTION' in the logcat it will look at one line in the logcat to check if it is in the log.

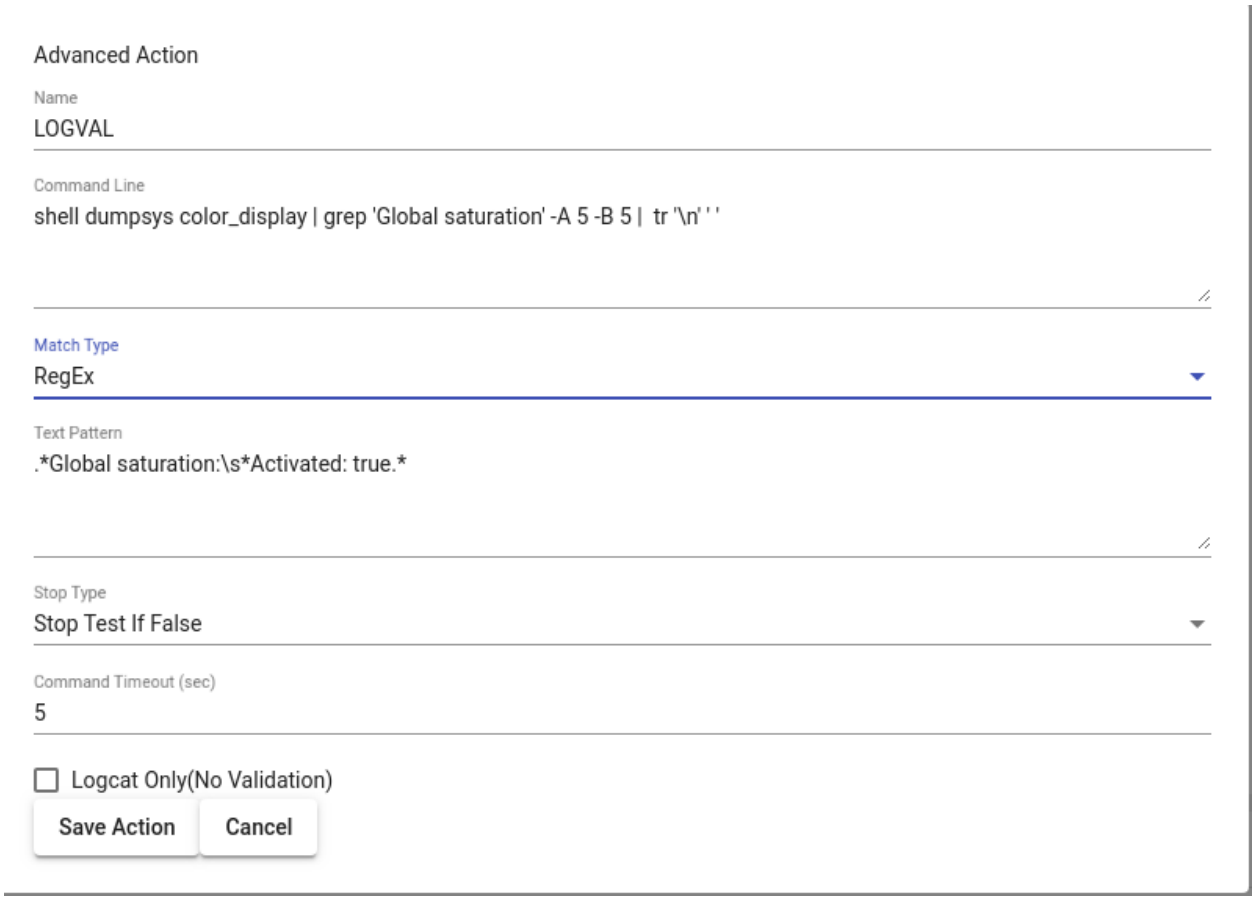


Because logcat validation treats each line as one string, command line manipulation is needed for certain validations. For example if you want to find the regex .*Global saturation:\s*Activated: true.* in the following text:

```
Activated: false
    Color temp: 2850
Global saturation:
    Activated: true
App Saturation:
    com.android.settings:
        0:
            mSaturationLevel: 100
            mControllerRefs count: 1
    com.google.android.apps.nexuslauncher:
        0:
            mSaturationLevel: 100
            mControllerRefs count: 1
```

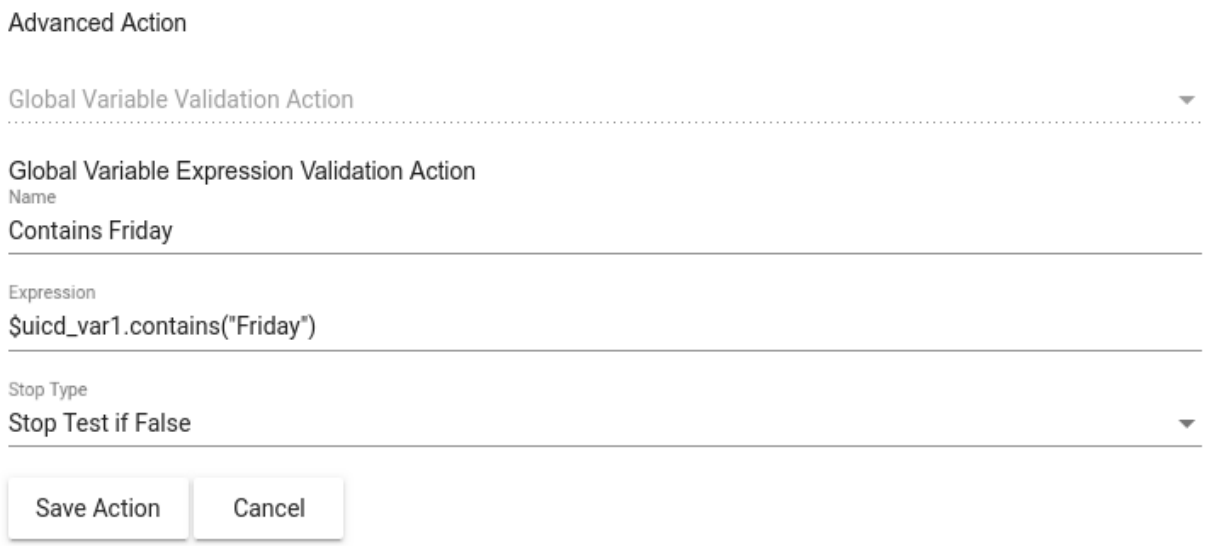
```
    com.google.android.apps.wellbeing:
        0:
            mSaturationLevel: 100
            mControllerRefs count: 4
    com.google.devtools.mobileharness.platform.android.app.binary.devicedaemon.v2:
        0:
            mSaturationLevel: 100
            mControllerRefs count: 1
Display white balance:
    Not available
Color mode: 3
```

manipulation would be needed on the text. In the following image, is the correct way to find the regex .*Global saturation:\s*Activated: true.*.



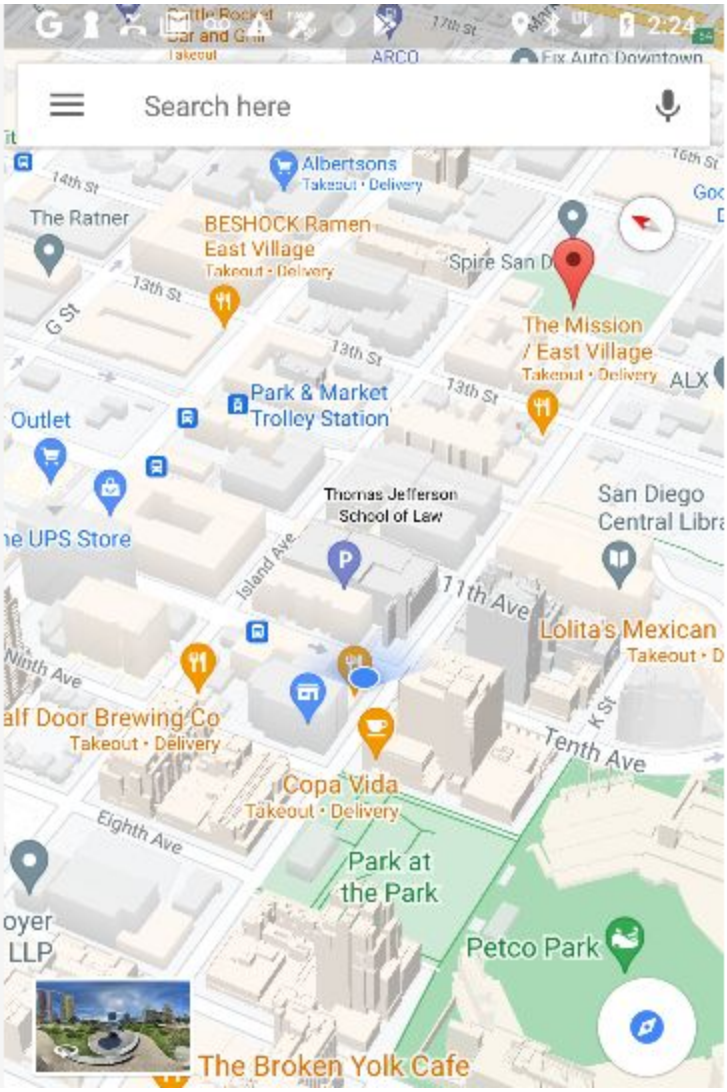
Global Variable Validation Action

Global variable validation action is used with common java methods. In this example we have set $uicd_var1 = "Friday". Then we use the global variable validation action to see if $uicd_var1 contains "Friday".



ML Image Validation Action

Validate if there is a blue dot on screen. A good example is when validating the blue dot on Google maps.



Condition Validation Action

Validate condition based on XML elements and attributes.

Double Tap Power Button Action

Mimics a double tap on the power button.

Python Script Action

Text/Resource-id

Click by text (text is visible name on the phone, for example Camera, Phone, etc):
Example - to open the camera app from a screen:

```
from python_uiautomator.device import Device
d = Device.create_device_by_serial("92KBA0000")

d.text("Camera").click()
```

If you find an error in this step which says Camera not found, make sure that the Camera app is actually present on the screen and that the xmlDumper app was installed.

Click by resource-id/content-desc:

```
from python_uiautomator.device import Device

d = Device.create_device_by_serial("92KBA0000")
d.resource_id("xxx").click()
d.content_desc("yyy").click()
```

Attributes

Similarly you can “select” element by any attribute in the xml node:

```
from python_uiautomator.device import Device

d = Device.create_device_by_serial("92KBA0000")
d.attributes("xxx").click()
```

More detail about the xml node can be found in UICD front end, click on UI Viewer, and click refresh XML and you will see all elements, at the very bottom you can see resource-id. See the section2 for more details.

MatchOption

We also support different types of match options to matching.
We support four types of match options: START_WITH, END_WITH, CONTAINS and REGEX.

Those options are fairly straightforward to use.

An example here is to select an element with text starting with “Settin”:

```
from python_uiautomator.device import Device
from python_uiautomator.constant import MatchOptions

d = Device.create_device_by_serial("92KBA0000")
d.text("Settin", MatchOptions.START_WITH).click()
```

And a regex example:

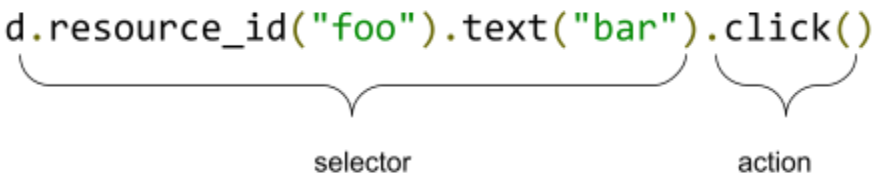
```
from python_uiautomator.device import Device
from python_uiautomator.constant import MatchOptions

d = Device.create_device_by_serial("92KBA0000")
d.text(".*etting$", MatchOptions.REGEX).click()
```

The above example will select the UiObject with text ending with “ettings”

Chain Selectors

In the previous example, we selected element by the Text:Camera


```
d.resource_id("foo").text("bar").click()
```



Which means find all elements whose resource_id is “foo” and later search all the descent elements of “foo”(note that there might be more than one “foo”), find any element that text is “bar”. Performs the Click at the first element to match the condition.

Users need to make sure to give a unique identifier to get an element. Otherwise will be unknown behavior.

Navigate the xml tree using selector

Python UIAutomator also provides the ability to allow the caller to directly “navigate” in the xml tree.
For example: We want to find the input box next to a particular text element. We can easily select it and do the following:

```
d.text("foo").parent().child(1).click()
d.input_text("bar")
```

Customized matching

Users can define their own validation function, users can write any function they want to “find” a particular xml node.

```
d.custom_matcher(lambda n: n.get("text") == 'Phone')
```

Verify elements exists

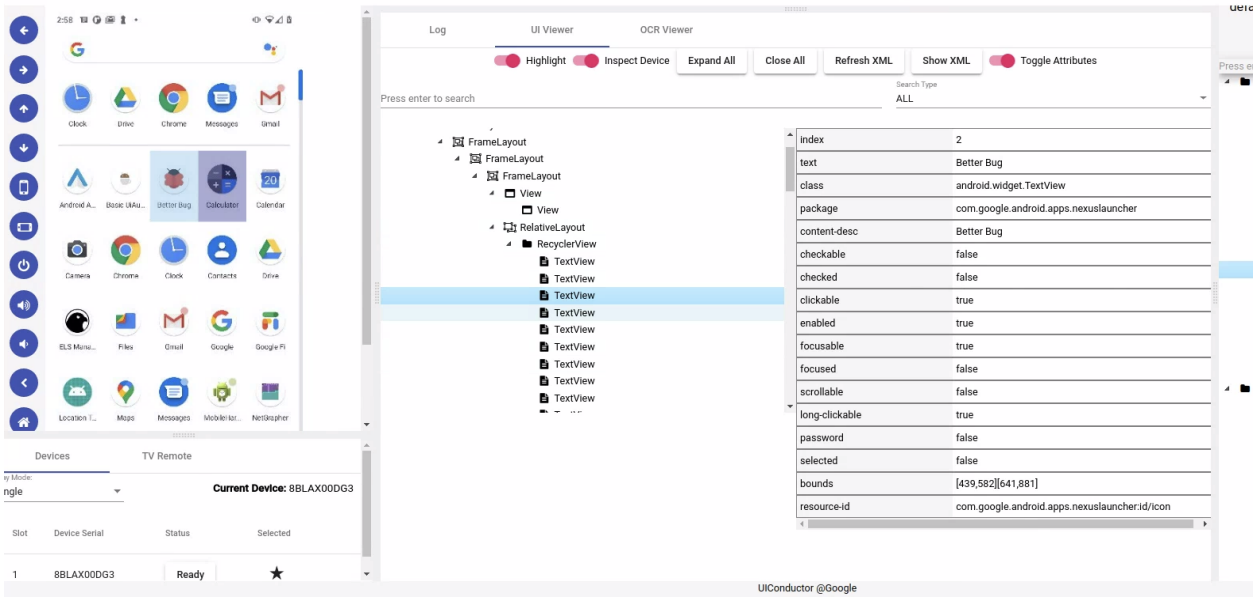
```
d.text("Clock").verify("class", "Clock")

d.text("Clock").verify_exist()

# simpler version
d.has_element("Text", "Clock")

# also you can do
d.has_text("Clock")
```

Wait for element

```
from python_uiautomator.ui_selector import UiSelector
phone_selector = UiSelector().text("Phone")
d.wait_for(selector=phone_selector)
```

UICD inspector

Use UICD inspector to understand more details of the xml hierarchy tree.



Click on Chrome Example

```
"""Sample MH test file for UICD PythonScriptAction.
 This is a demo to the simple click functionality.
"""
from python_uiautomator.device import Device
from python_uiautomator.uicd_python_util import UICDPythonUtil

uicd_util = UICDPythonUtil()
d = Device.create_device_by_slot(0)
d.home()
d.text("Chrome").click()
```

Verify element exists uicd.util.assert_true example

```
"""Sample MH test file for UICD PythonScriptAction.
 This is a demo to the simple assert functionality.
"""
from python_uiautomator.device import Device
from python_uiautomator.uicd_python_util import UICDPythonUtil

uicd_util = UICDPythonUtil()
d = Device.create_device_by_slot(0)
d.home()
result = d.has_text("Chrome")
uicd_util.assert_true(result, "Element not found")
```

Global variable example

```
"""Sample MH test file for UICD PythonScriptAction.
 This is a demo to the simple click functionality.
"""
from python_uiautomator.device import Device
from python_uiautomator.uicd_python_util import UICDPythonUtil

uicd_util = UICDPythonUtil()
d = Device.create_device_by_slot(0)
d.home()
d.text("Chrome").click()
uicd_util.set_variable("$uicd_var1", "abc")
uicd_util.save_uicd_global_variable()
```

Multi-device example

```
"""Sample MH test file for UICD PythonScriptAction.
 This is a demo to the multi device testing functionality.
"""
from python_uiautomator.device import Device
from python_uiautomator.uicd_python_util import UICDPythonUtil

uicd_util = UICDPythonUtil()
d1 = Device.create_device_by_slot(0)
d2 = Device.create_device_by_slot(1)
d1.home()
d1.text("Chrome").click()
d2.home()
d2.text("Chrome").click()
```

Save Workflow

Saves the workflow in the workflow editor.

Add Wait

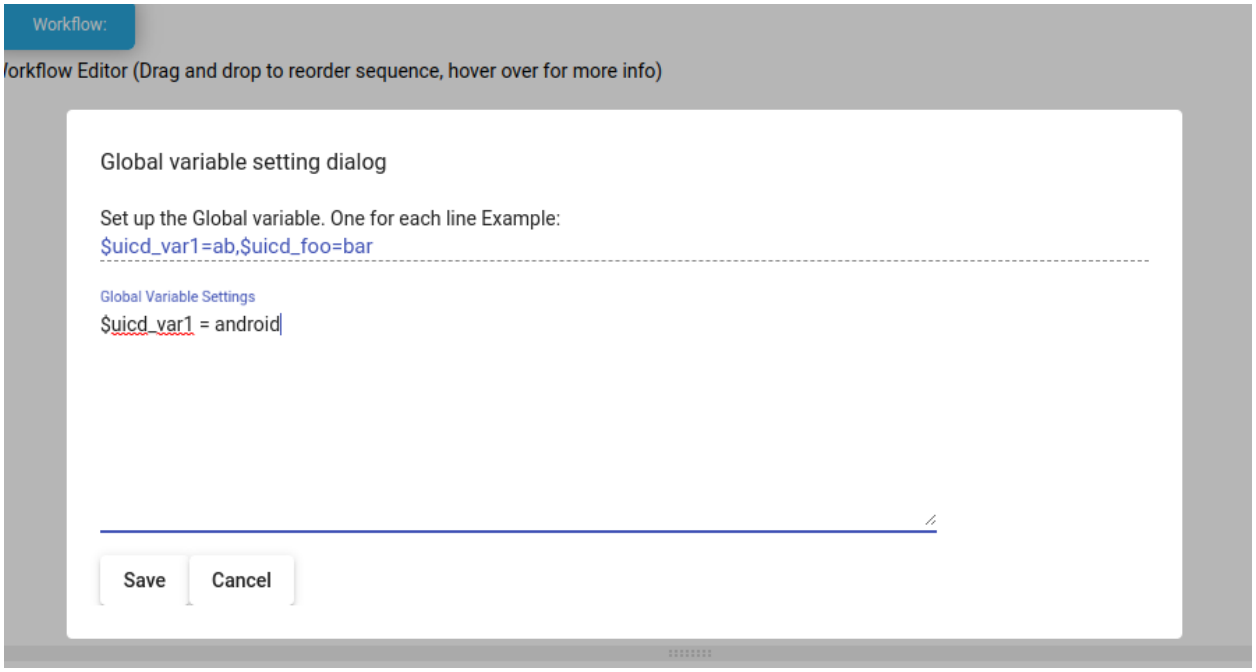
Add wait time in the workflow.

Test History

History of the tests that were run.

Global Variable Settings

Sets up a global variable to be used in the current workflow.



Undo

Undo an action in the workflow.

Redo

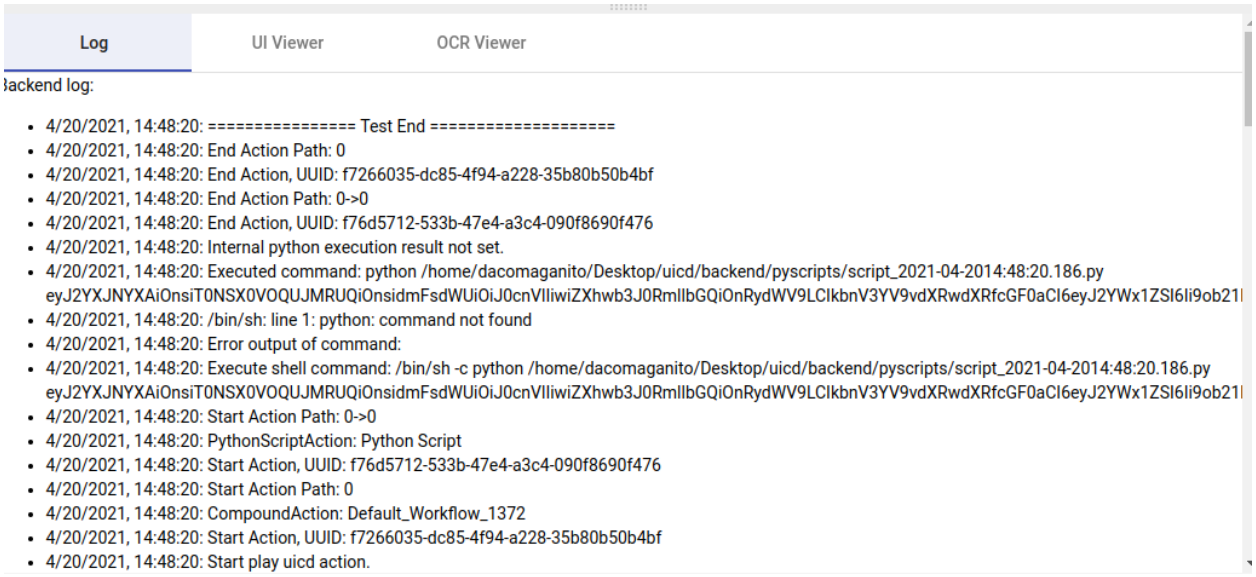
Redo an action in the workflow.

Python Debugger

Used for debugging python actions.

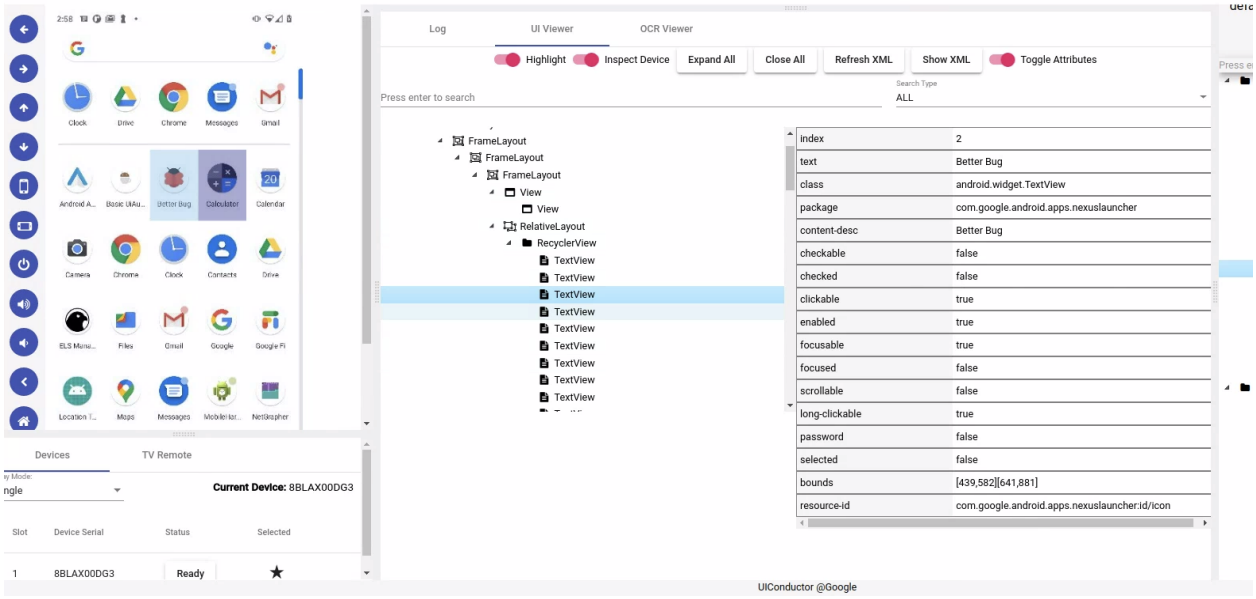
Log Section

Logs the actions in the workflow. Can be used for troubleshooting as well.

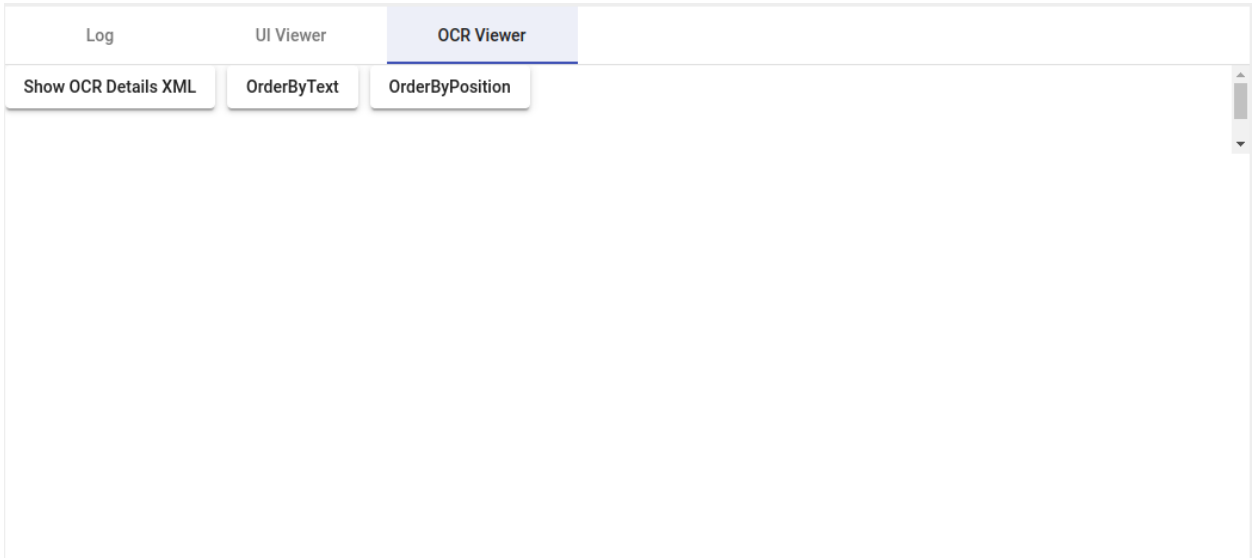


UI Viewer

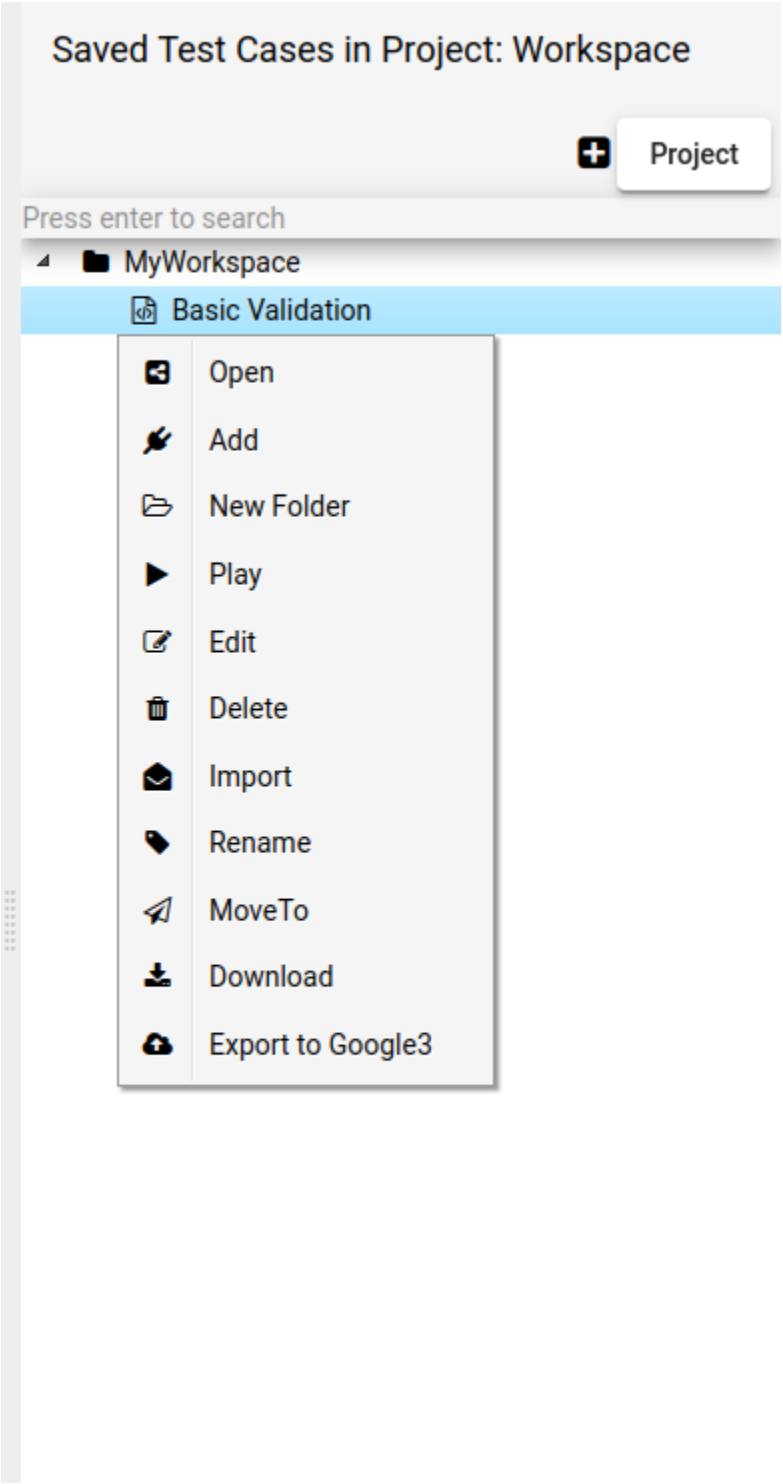
Shows the XML hierarchy tree information.



OCR Viewer


Log
UI Viewer
OCR Viewer
Show OCR Details XML
OrderByText
OrderByPosition

Test Case Management



+

Create a new workspace.

Project

Create a New project, open a new project, share current project, and export/import a project.

Add

This operation allows you to add another previously saved workflow into the current one. This allows for reusability of a test case. It is always good practice to create a small workflow where the steps of the workflow can be applied to many other test cases. For instance, you may have 20 test cases which will require you to toggle the WiFi settings. If you create a small workflow which only does this WiFi toggle routine, it can be added to any test case that requires this step.

Import

This is used to copy a test case from another user and put it into your list of test cases. You can import test cases by using the original test case's UUID, if you know it, or you can use the search by user method by entering the user name of the person who created the test case. Note that you can not modify the actions in an imported workflow but you can add your own actions to that flow and it doesn't show any impact on the other person's test.

Move to

This is used to move the saved test cases between the folders. Alternatively, you can just drag a test case into another folder.

Export

This function is used to generate a text file from a saved test case. This is mainly used to run the test in Moscar/Mobile Harness.

Other Options

Open, Edit, Play, Rename, and Copy which should be straightforward.

Reserved Keywords

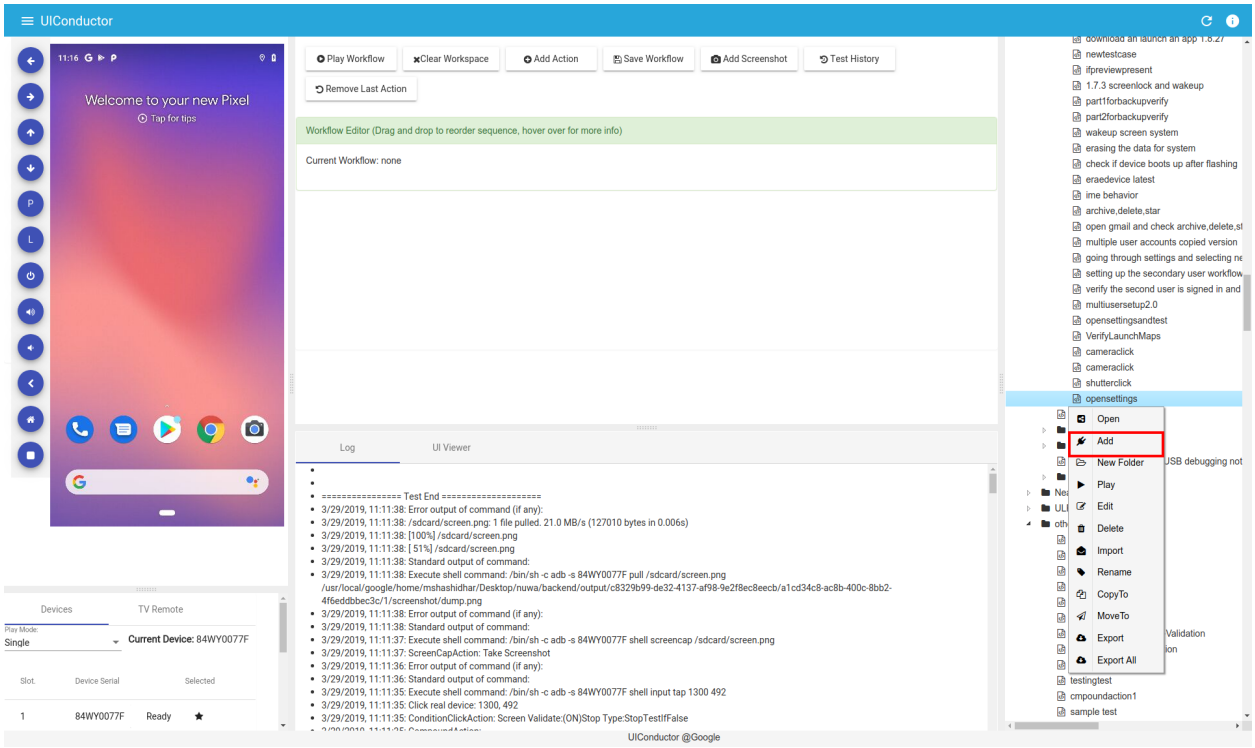
Name	Description
$uicd_adb_path	Adb path Uicd is using.
$uicd_input_path	Input path.
$uicd_output_path	Output path, the root path of current run, includes all the test details screenshots etc.
$uicd_deviceid	Current device id uicd is using.

Reuse the compound action

UICD gives you the option to reuse the test cases that have been recorded. For example you have multiple test cases which have a step to toggle the WiFi button. This step can be recorded once and then can be added to the other test cases where required instead of recording it everytime.

1. Go to the test case you want to use.

2. Right click on it and select the add option. This will add the test case to the new workflow.

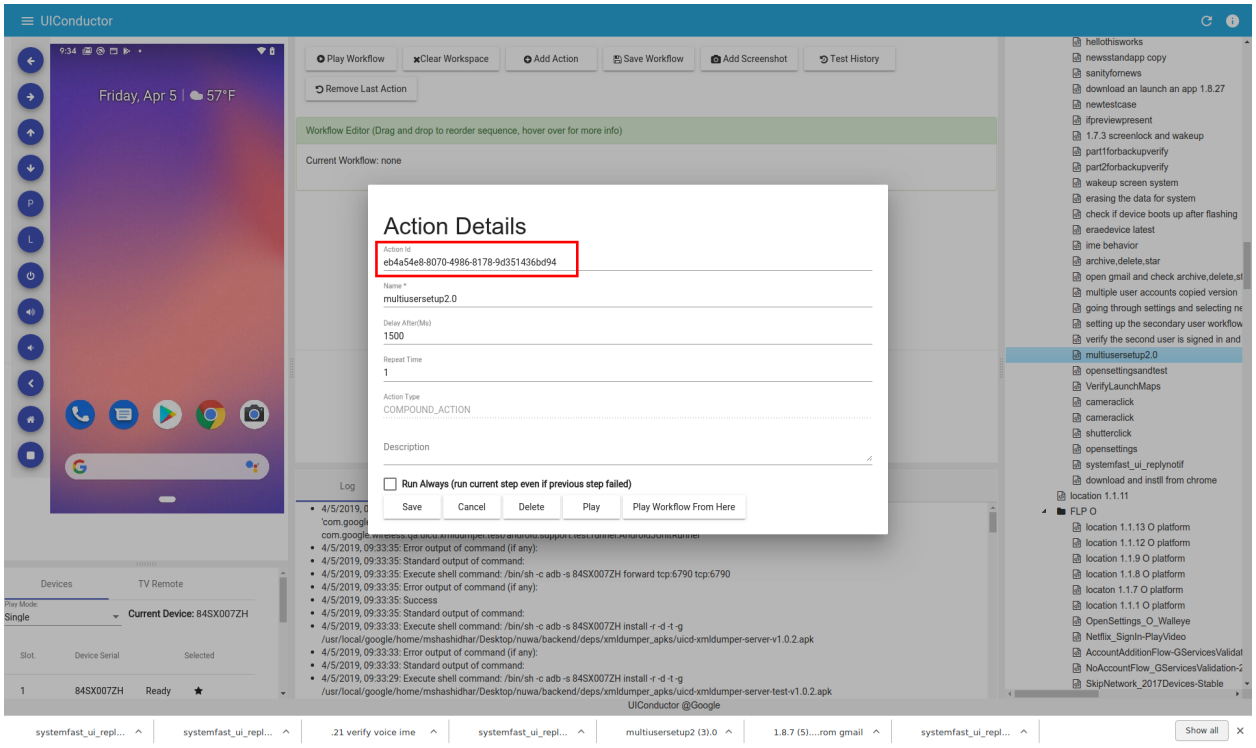


Using another user's compound action

UICD provides an option to share the test cases with another person using the import option. The imported test case may be used as a part of your work flow or you can add some steps to the imported test case depending on the requirement.

1. Know the Action ID of the test case to be imported. For instance, say there are two individuals, say A and B. A is trying to import the test case from B so B needs to provide his Action ID to A. How can B get to know the Action ID? It's very simple, B goes to the test case in his workspace he wants to share, right clicks on it then he chooses the edit option and clicks on it. He will be able to see a popup as shown in the image below, the

highlighted area will be the action ID for that test case.



2. Right click on the folder you want the test case to be imported.
3. Select the option import and input the Action ID, the required test case will be imported to your work space. You can even import the test case by knowing the ldap of the other person.