Contents

[SETTING UP TEST ENVIRONMENT 1](#_Toc528251025)

[OS REQUIREMENTS: 1](#_Toc528251026)

[SOFTWARES TO INSTALL 1](#_Toc528251027)

[COPYING TEST FILES 6](#_Toc528251028)

[**AUTOMATION SETUP** 7](#_Toc528251029)

[TEST EXECUTION 12](#_Toc528251030)

[DEBUGGING TIPS: 17](#_Toc528251031)

[HID Keycode/INTENT Semi-Automated tests 17](#_Toc528251032)

[if it works, you must be presented with a prompt like this : 18](#_Toc528251033)

[**Intent validator tests** 18](#_Toc528251034)

[**Individual intent tests** 19](#_Toc528251035)

[**Keycode validator** 20](#_Toc528251036)

# SETTING UP TEST ENVIRONMENT

## OS REQUIREMENTS:

Windows 10 / Windows Server 2012 r2

64 bit architecture

**NOTE:**

* Wherever command prompt is used, please run it as administrator.

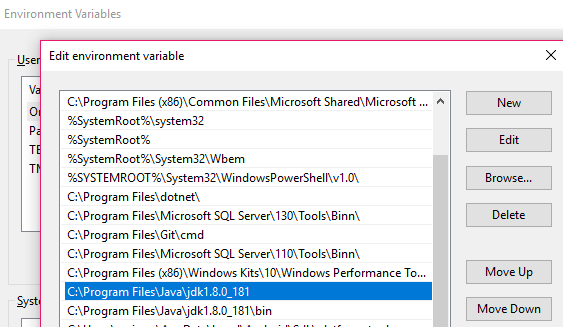
SOFTWARES TO INSTALL**:**

1. JRE/JDK Java SE Development Kit 8u181 [http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html%20) (64bit)

After installing, make sure the jdk bin is set in PATH in EnvironmentVariables.

C:\Program Files\Java\jdk1.8.0\_181

C:\Program Files\Java\jdk1.8.0\_181\bin



If everything is set, then from command prompt, you should be able to see the version of installed jdk without any errors. Eg.,

* **java -version**

*java version "1.8.0\_181"*

*Java(TM) SE Runtime Environment (build 1.8.0\_181-b13)*

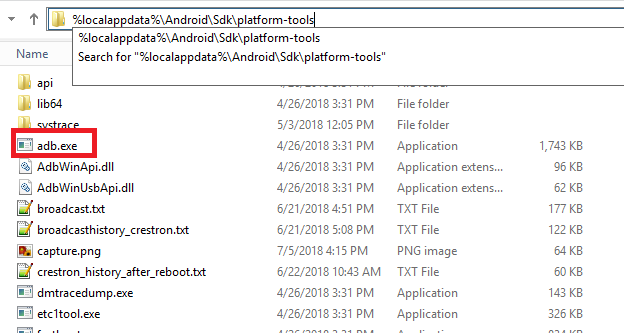
*Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)*

2. Android studio 2.3.3 (pre-req: JRE/JDK)

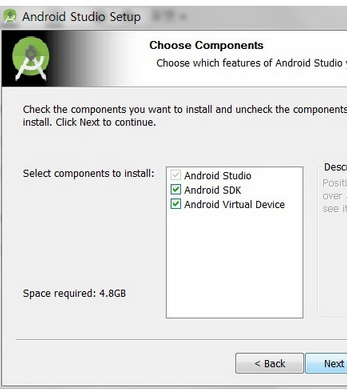
[https://developer.android.com/studio/archive.html](https://developer.android.com/studio/archive.html%20) (64 bit, select bundle with sdk)

SDK is very important. Make sure you see the following after install. Open file explorer,

%localappdata%/Android/Sdk/platform-tools/. You should see adb.exe (Fig. below)

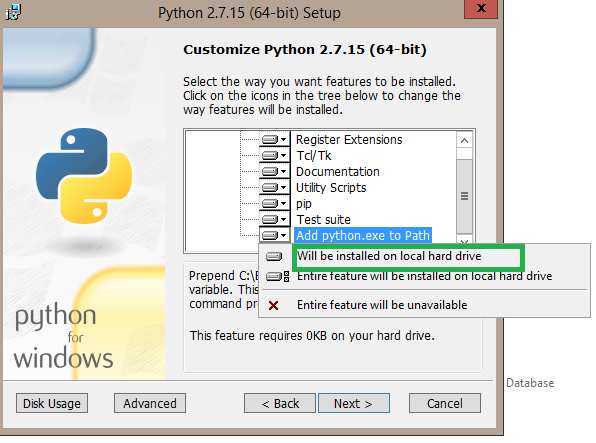


If you don’t see this, Restart your computer, try installing android studio again and make sure as you install, you see sdk components selected.



3. Python 2.7.15 <https://www.python.org/ftp/python/2.7.15/python-2.7.15.amd64.msi>

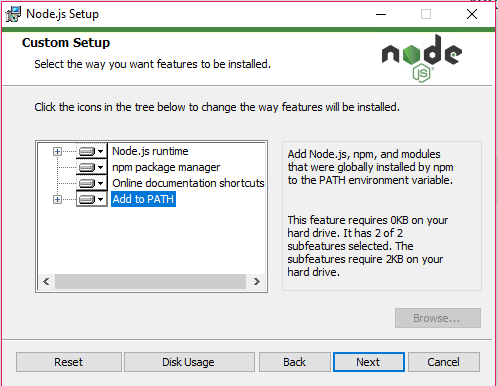
- As you install, make sure you select option "Add Python.exe to PATH" as “Will be installed on local hard drive”



4. Install node.js **(>10.1.0) (pre-req: jre/jdk)**

For windows 64 bit : <https://nodejs.org/dist/v10.8.0/node-v10.8.0-x64.msi>

As you install, make sure all components are selected (ie., there is no cross mark × against any of the modules). It should be like below image



Let it install in the default location.

After installation, open command prompt, type:

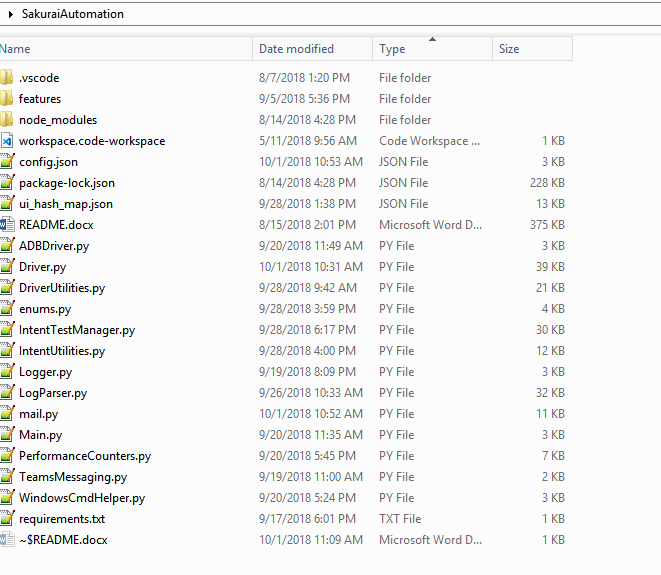
* node –v

This should show the nodejs version you’ve installed.

## COPYING TEST FILES

1. Copy thefiles to your local directory. Eg., under **c:\SakuraiAutomation**

**Directory should look like this:**



1. From **command prompt (admin)**, navigate to the project folder
   1. cd c:\SakuraiAutomation
2. **Install Appium :**

* npm install –g appium

After the command is run, to check if appium is installed, type

* appium –h

This should show all the available command options for appium0

1. ( Run the following command from folder where the project is placed and wait for completion. This will install all dependencies needed for the project )
   1. **cd c:\SakuraiAutomation pip install -r requirements.txt**

# **AUTOMATION SETUP**

1. Depending on the tests, you may need single or multiple(maximum 3) devices.

Eg., for sign-in tests – 1 device

For transfer tests – 3 device.

**Nice to have :** 3 dedicated devices for automation.

1. **ADB setup:**

Connect each device to your computer using adb.

From **command prompt,**

1. If connecting via IP,

* adb connect <IP\_ADDRESS>
* adb –s <IP\_ADDRESS> root

(If you root the device for the first time, adb conection will be reset so you have to connect again using following command. No harm connecting multiple times.)

* adb connect <IP\_ADDRESS>
* adb devices

(This should list your device’ IPaddress)

1. If connecting via USB,

* Plugin your device to your computer’s usb port.
* adb devices  
  (This should list your device’ MAC ID)

1. **TEST CONFIG:**

Devices configuration file is in c:\sakuraiautomation\config.json (pasted below)

**Following parameters are mandatory**. Please configure them correctly.

|  |  |
| --- | --- |
| **ATTRIBUTE** | **DEFINITION** |
| apk\_download\_path | Path where the Teams APk resides.  Note : This location should have only 1 teams apk which we want the devices to use. This will be installed via automation using **adb instal**l. CompanyPortal app needs to be installed manually. |
| primary\_test\_device\_model | We have one primary test device defined in a test suite. This will be the most active device in all the tests.   Device model of the primary test device should be given here. This should match the **model** tag under **devices** |
| users | List of all users involved in the tests.  **Ellie is the primary user** in all tests. Leo and Elizabeth will be used as required.  Make sure username, password and displayname are **AS-IS**. |
| devices | List of all devices.  **Serial** - For devices connection via usb, this should be the **MAC ID**.  For devices connected via IP address, this should be the **IP address**. |
| Teams\_webhooks: "schedule\_webhook" "success\_webhook" "failure\_webhook" "info\_webhook" | We will be using **4** webhooks in teams channel to publish test status. Please configure webhook with the below steps.  You’ll need to repeat this 4 times for each of the webhook.  Make sure you are admin in any of the team in Microsoft Teams.   1. Click on the ellipsis icon **…** on the channel you want to monitor test results. Choose **Connectors** 2. Search for **Webhook** and click **Configure** next to **Incoming Webhook** 3. Give the webhook a name (eg, schedule/success/failure/info) and, optionally, a custom image (optional) and click **Create** 4. On the next page you are given a URL. You need to use this URL as values for the webhook. |

**Following parameters are optional**. Please configure them correctly.

|  |  |
| --- | --- |
| **ATTRIBUTE** | **DEFINITION** |
| sendgrid\_api\_key | We use azure sendgrid to send emails after test completion. If you wish to receive email, please setup Sendgrid following this link : <https://docs.microsoft.com/en-us/azure/sendgrid-dotnet-how-to-send-email>  The value for this attribute would be the Generated API key. |
| mail\_to\_users | List of users whom you want the mail to be sent. (separated by comma) |
| common\_log\_share | Common share where you want the logs to be copied over after the test is run. |

**Sample json:**

{

"primary\_test\_device\_model" : "AB1",

"apk\_download\_path" : "C:\\Teams\_APK",

"common\_log\_share" :"\\\\minint-ba6e58r\\Scratch\\Sakurai",

"mail\_to\_users" :"v-sigop@microsoft.com,vimall@microsoft.com",

"sendgrid\_api\_key" :"…",

"users" : {

"Ellie" : {

"username" : "a1@a.onmicrosoft.com",

"password" : "dummy ",

"displayname" : "A 1",

"phonenumber" : "11234567894",

"pstndisplay" : "+1 123-456-7894"

},

"Leo" : {

"username" : "a2@a.onmicrosoft.com",

"password" : "dummy ",

"displayname" : "A 2",

"phonenumber" : "14789456321",

"pstndisplay" : "+1 478-945-6321"

},

"Elizabeth" : {

"username" : "a3@a.onmicrosoft.com",

"password" : "dummy ",

"displayname" : "A 3",

"phonenumber" : "13245678914 ",

"pstndisplay" : "+1 324-567-8914 "

},

"Peter" : {

"username" : "a4@a.onmicrosoft.com",

"password" : " dummy ",

"displayname" : "A 4",

"phonenumber" : "14875647892 ",

"pstndisplay" : "+1 487-564-7892 "

}

},

"devices" : {

"IPPhone1" : {

"serial" : "10.176.33.200",

"ip\_address" : "10.176.33.200",

"brand" : "Partner1",

"model" : "S12",

"firmware" : "1.0”

},

"IPPhone2" : {

"serial" : "10.176.33.134",

"ip\_address" : "10.176.33.134",

"brand" : "Partner2",

"model" : "AB1",

"firmware" : "1.0”

},

"IPPhone3" : {

"serial" : "10.176.33.201",

"ip\_address" : "10.176.33.201",

"brand" : " Partner1",

"model" : "S13",

"firmware" : "1.0"

},

"IPPhone4" : {

"serial" : "10.176.33.221",

"ip\_address" : "10.176.33.221",

"brand" : " Partner1",

"model" : "S14",

"firmware" : "1.0"

}

} ,

"teams\_webhooks":{

"schedule\_webhook" : "https://outlook.office.com/webhook/11",

"success\_webhook" : "https://outlook.office.com/webhook/22",

"failure\_webhook" : "https://outlook.office.com/webhook/33",

"info\_webhook" : "https://outlook.office.com/webhook/44”

}

}

**Note**: Peter is Fed account

TEST EXECUTION **:**

* **We don’t support conference/meeting creation through automation**. So, please manually create a every-day recurring meeting with the below titles and have the test users invited into it.

**Meeting names :**

1. **single\_person\_meeting**
2. **multi\_party\_meeting**

* Tests are written in Specflow (English-like) language. To execute them, we will be using “**Lettuce**” (This should have been installed as part of requirements.txt)
* Reference for Lettuce : <http://lettuce.it/reference/terrain.html>
* Available test features are under

**C:\sakuraiautomation\features**

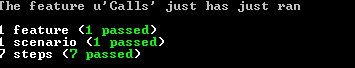
**List of available test features**

|  |  |
| --- | --- |
| **Feature name** | **Objective** |
| Signin.feature | Test signin using personal and share account |
| Calls.feataure | Test P2P call and conference join. |
| PerfTests.feature | Perf test suite (runs common user scenarios for 6 times) and gives cpu/memory usage for each scenario. |
| Sanity.feature | Sanity tests. Covers signin, p2pcall, meeting join, UI hold/resume, UI mute/unmute, conference join |

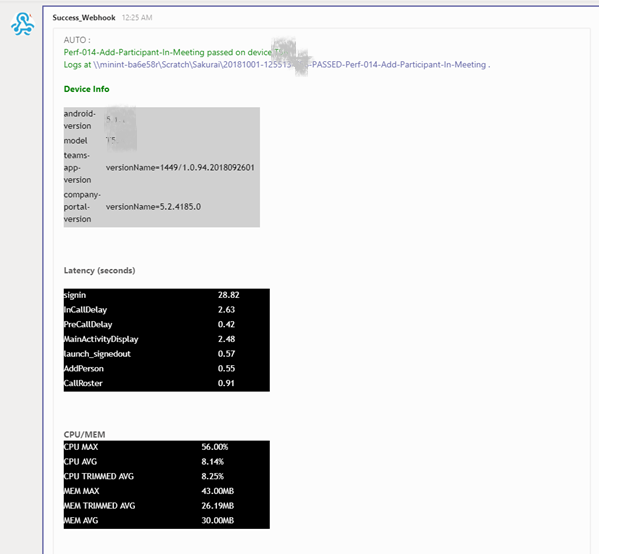
* For example, to execute P2P call scenario

From **command prompt** navigate to project directory and execute the below command:

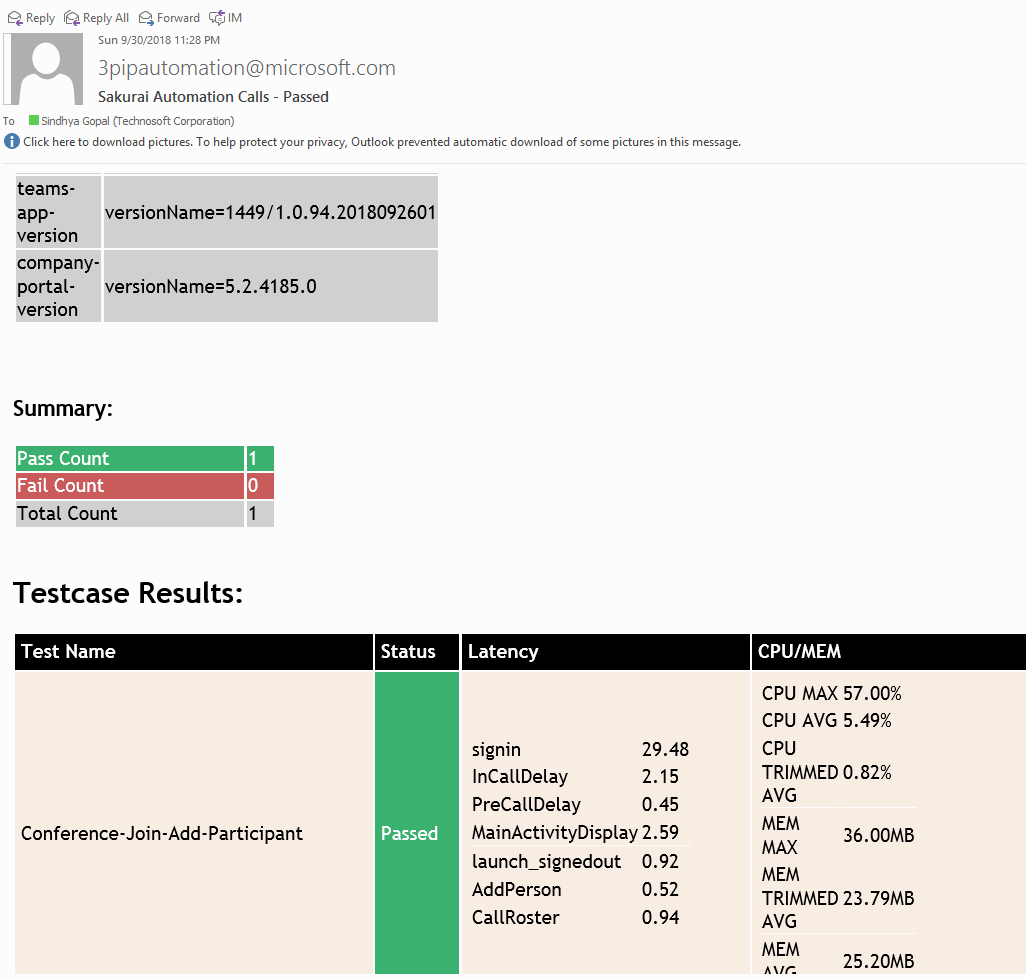
* Cd c:\SakuraiAutomation
* **lettuce features\Calls.feature** Once the execution is complete, you should see :



* webhook like this to your channel :

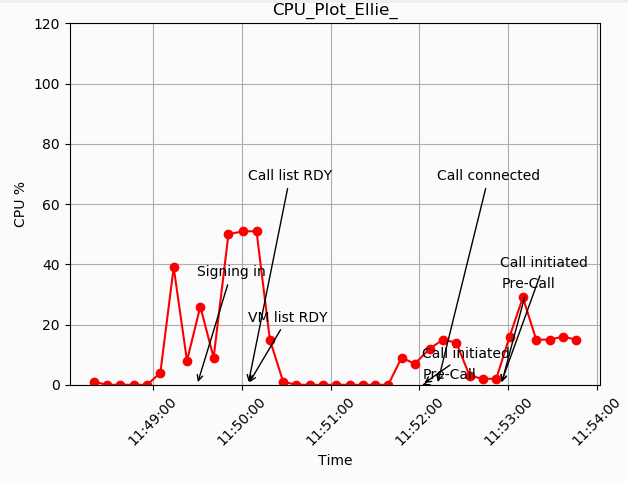


* **If you have sendgrid setup, you should receive email like this :**



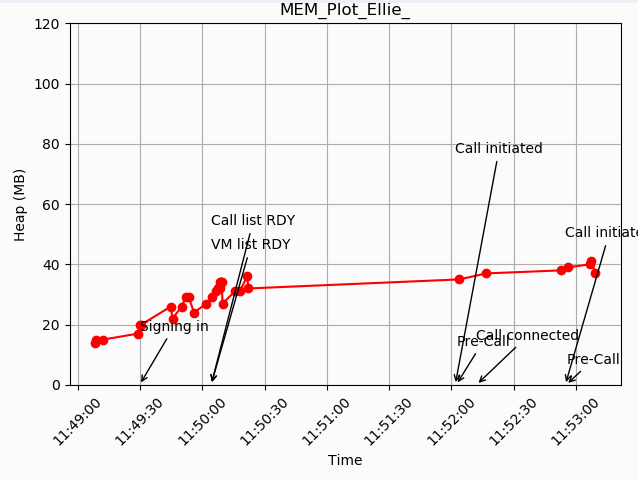
**LOG DESCRIPTION :**

Logs should be in Projectdirecory\<FeatureName\_datetime>\<ScenarioName> :

* CPU\_Plot\_**Ellie**<…>.png : This will show the teams app CPU % consumption for **Ellie** in the test.

Likewise, for each user involved in the tests, you should see CPU\_Plot**\_Username**..png

* MEM\_Plot\_**Ellie**<…>.png : This will show the teams app Heap MEM (MB) consumption for **Ellie** in the test.



Likewise, for each user involved in the tests, you should see MEM \_Plot**\_Username**..png

* **Testlog.log –** Towards the end of testlog, you ‘ll see the summary of all users involved in the tests. Eg.

############## BEGIN ANALYZED DATA . FOR :Ellie TP 100 ###################

LATENCY :

\*\*\*\*\*\*\*\*\*\*

signin :: TP 100 :: 34.432

MainActivityDisplay :: TP 100 :: 1.642

Device Info :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*

model :: AB1

android\_version :: 5.1.1

teams\_app\_version :: versionName=1449/1.0.94.2018072702

CPU/MEM info :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cpu\_max :: TP 100 :: 57.0

mem\_max :: TP 100 :: 35.0

cpu\_avg :: TP 100 :: 2.0

mem\_avg :: TP 100 :: 24.6

* Logcat**\_<device\_serial>.**log : This will have logcat logs for each of the devices involved in the tests. Logcat logs are split up into different files for easy viewing.

Logcat\_<device\_serial>.log

Logcat\_<device\_serial>\_1.log

Logcat\_<device\_serial>\_2.log

**LETTUCE COMMAND SHEET**

|  |  |
| --- | --- |
| **Command** | **Description** |
| Lettuce features/Calls.feature –s 1 | Runs the 1st scenario in the Calls.feature test |
| Lettuce features/Calls.feature –with-xunit –xunit-file=report.xml | **Gives a XML test report at the end of test.** |
| Lettuce features/PerfTests.feature | Runs all tests in PerfTests suite |

# DEBUGGING TIPS:

* For errors like “importerror: no module named x”
* Please install “x” by running the below command :
* pip install x

# HID Keycode/INTENT Semi-Automated tests

Pre-Req : These tests are run only on a single device. Make sure it is connected via adb.

All tests are to be verified manually.

Open command prompt as administrator and navigate to the project directory and run the below cmd

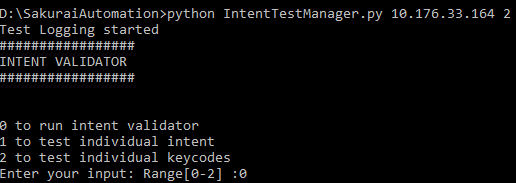
C:/SakuraiAutomation> python IntentTestManager.py <adb\_serial> <device\_type>

Eg., if I am connecting a conference device with IP 3.3.3.3

python IntentTestManager.py 3.3.3.3 1

|  |  |
| --- | --- |
| Device | Device\_type |
| Conference room (only mute and speaker led) | 1 |
| Personal device | 2 |

## if it works, you must be presented with a prompt like this :

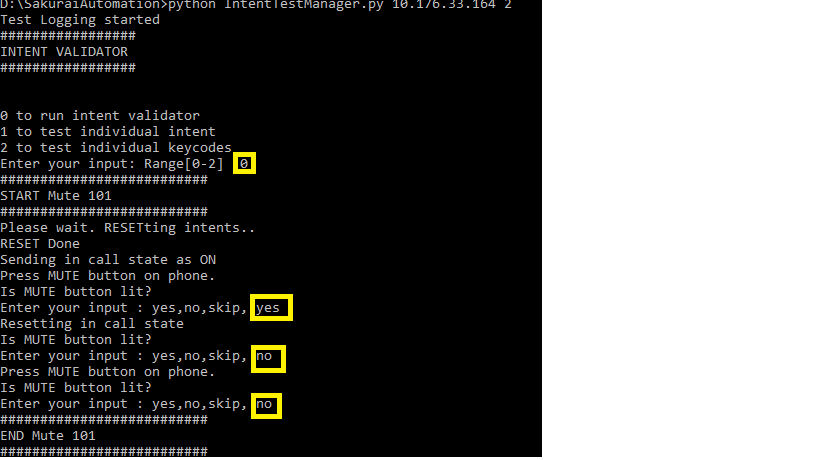


## **Intent validator tests**

Batch of tests to check Mute/Speaker/Handset functionality. Please verify manually for the behavior.

Follow the instructions displayed on prompt,

Eg:



At the end of test, you ll see statistics like this:

#####################

SUMMARY

#####################

Total tests : 7

Total passed : 7

Total failed : 0

Total N/A : 0

Passed test : Mute 101

Passed test : Dialtone 101 (SPEAKER)

Passed test : Dialtone 101 (HEADPHONE)

Passed test : Dialtone 101 (HANDSET)

Passed test : MissedCall 101

Passed test : VM 101

Passed test : INCOMING CALL 101

## **Individual intent tests**

Helps debugging intents

A screenshot of a social media post with text and a black background

Description automatically generated

## **Keycode validator**

Helps debugging keycodes sent from the device.

A screenshot of a social media post with text and a black background

Description automatically generated