Android GUI-Verification Tool

Version 1.3

Documentation and User Guide

Table of Contents

1) Setting Up Your Environment

1.1) Installing Java

To run the GUI-Verification Tool (GVT) you will need to install Java and set up the proper path variables on your machine. We built the GVT using Java to help facilitate the tool working across different Operating Systems and computing environments. Installing Java is a relatively simple procedure, just follow one of the guides below if you need to install it.

Installation Guide for Windows: http://docs.oracle.com/javase/7/docs/webnotes/install/windows/jdk-installation-windows.html

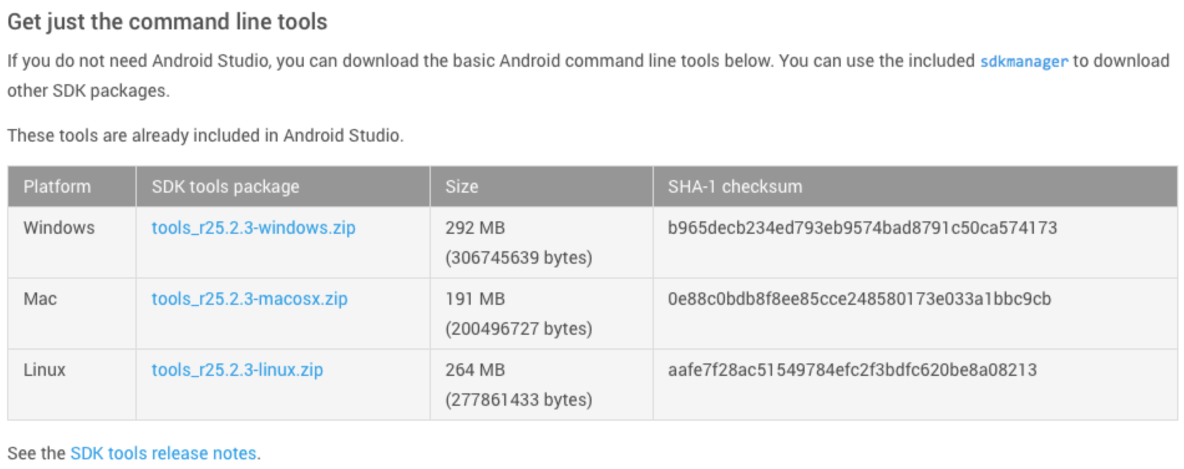
Installation Guide for Mac: http://docs.oracle.com/javase/7/docs/webnotes/install/mac/mac-jdk.html

Installation Guide for Linux: http://docs.oracle.com/javase/7/docs/webnotes/install/linux/linux-jdk.html

1.2) Installing the Android SDK

To run the Android-UI-Helper which extracts screenshots and UI-dump xml files from an app currently running on an Android device, you must first install the most recent version of the Android SDK. This will allow our program to interface with a connected Android device or emulator connected to a PC or Mac. It is not necessary to install the full-fledged version of the SDK that includes Android Studio, but rather, you can simply download the SDK tools, and use the built-in installer to download the platform-tools. This process involves the following steps:

1) Download the Android SDK command line tools (See Picture below) from the following link, you will need to scroll down to the bottom of the page for the correct download: https://developer.android.com/studio/index.html



Get just the command line tools

If you do not need Android Studio, you can download the basic Android command line tools below. You can use the included `sdkmanager` to download other SDK packages.

These tools are already included in Android Studio.

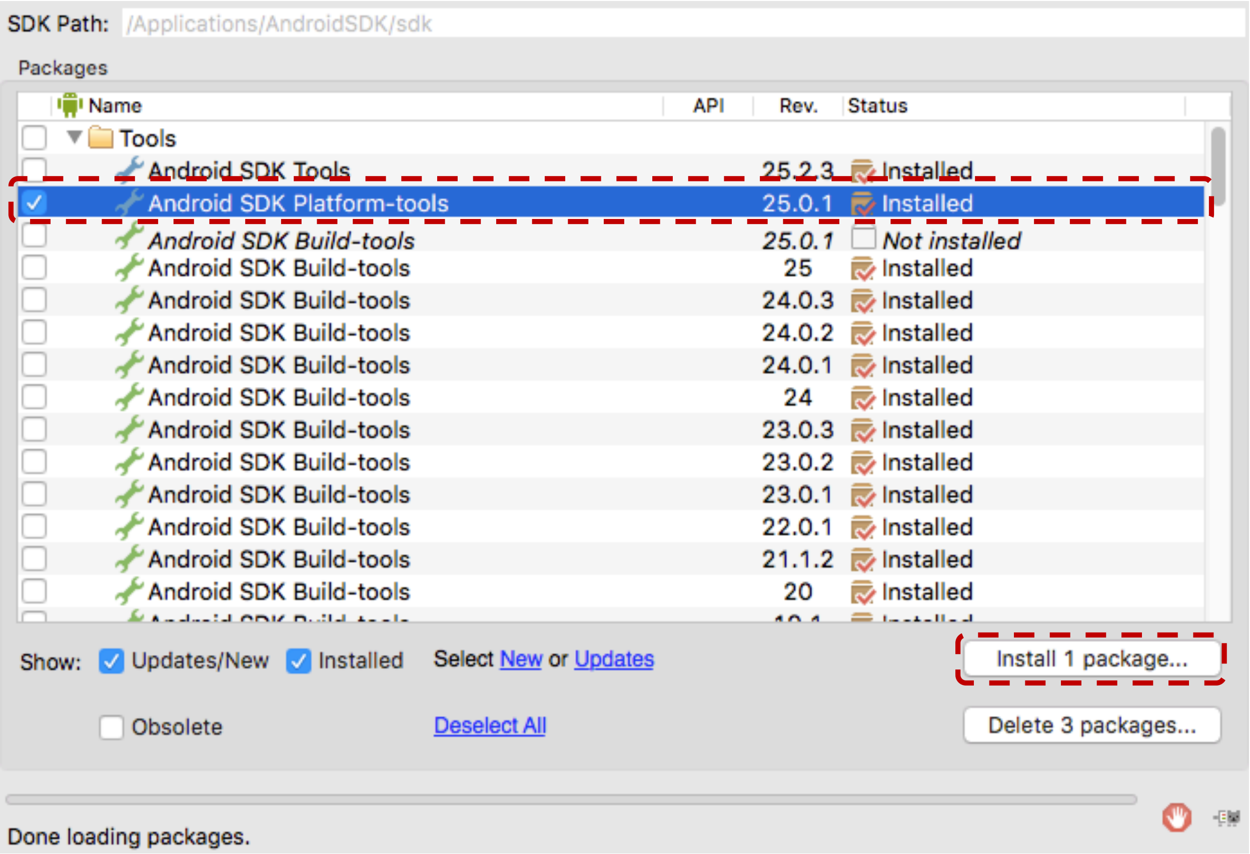
Platform	SDK tools package	Size	SHA-1 checksum
Windows	tools_r25.2.3-windows.zip	292 MB (306745639 bytes)	b965decb234ed793eb9574bad8791c50ca574173
Mac	tools_r25.2.3-macosx.zip	191 MB (200496727 bytes)	0e88c0bdb8f8ee85cce248580173e033a1bbc9cb
Linux	tools_r25.2.3-linux.zip	264 MB (277861433 bytes)	aafe7f28ac51549784efc2f3bdfc620be8a08213

See the SDK tools release notes.

2) Unzip the file in the location of your choice. For convenience, it is probably best to place this in your home or user directory.

3) After you unzip the file, navigate to the tools folder and run the android (or android.bat) executable file to bring up the SDK Manager.

4) After the Android SDK manager window opens, the box to download the Android SDK Platform tools, and click the “Install Packages” Button. See the red areas of the screenshot below for more detail:



SDK Path: /Applications/AndroidSDK/sdk

Packages

	Name	API	Rev.	Status
□	▼ Tools			
□	Android SDK Tools		25.2.3	Installed
☑	Android SDK Platform-tools		25.0.1	Installed
□	*Android SDK Build-tools*		*25.0.1*	*Not installed*
□	Android SDK Build-tools		25	Installed
□	Android SDK Build-tools		24.0.3	Installed
□	Android SDK Build-tools		24.0.2	Installed
□	Android SDK Build-tools		24.0.1	Installed
□	Android SDK Build-tools		24	Installed
□	Android SDK Build-tools		23.0.3	Installed
□	Android SDK Build-tools		23.0.2	Installed
□	Android SDK Build-tools		23.0.1	Installed
□	Android SDK Build-tools		22.0.1	Installed
□	Android SDK Build-tools		21.1.2	Installed
□	Android SDK Build-tools		20	Installed

Done loading packages.

2) Using the Android UI Helper

2.1) Tool Description & Setup Procedure

The Android UI Helper is a helper utility that allows for the capture of a UI-Dump consisting of an .xml file that contains information relating to the GUI-Hierarchy of a target Android application screen. These combinations of UI-dumps and screenshots are used as input for the GVT which compares them against a “ground truth” design mock-up (in the form of a Marketch file). The tool is self-contained within a jar file, but depends on the Android SDK to interface with a physical Android device or emulator. The executable jar file is called “`GVT.jar`”. To use this tool there are a few setup requirements:

1) If you will be using a physical Android phone as opposed to an emulator, you will need to enable the developer features and then enable USB debugging. This allows the phone to connect to a PC or Mac using the Android Debug Bridge (adb) which our program uses to interface with the device. Here is a link that describes the simple procedure to enable this: https://goo.gl/CXei8H

2) Next, if you are using a physical device, connect the device to your computer using the provided USB cable. If you are using an emulator, launch the emulator. Information about how to configure and launch an Android emulator can be found here: https://goo.gl/0eTeme
3) Then install the application for which you would like to capture UI-information. Then launch the app and navigate to the screen for which you would like to capture data. That's it! You are now ready to run the java program to capture the screen information.

2.2) Android UI-Helper Usage Information

The Android-UI-Helper is usage is as follows:

```
java -jar GVT.jar AndroidHelper [Path to Android SDK]
[Output File Path for UI-Dump] [Output File Path for Screenshot]
```

A description of the program arguments are as follows:

Path to Android SDK: This should be the full (not relative) path to root folder of the Android SDK that you previously installed. For example: `/Users/KevinMoran/dev/Android-SDK/`

Output File Path for UI-Dump: This is the full (not relative) file path for the output location of the ui-dump .xml file generated by the program. This file path should end in a file that uses the ".xml" file extension. For example: `/Users/KevinMoran/dev/ui-dump.xml`

Output File Path for Screenshot: This is the full (not relative) file path for the output location of the screenshot image file generated by the program. This file path should end in a file that uses the ".png" file extension. For example: `/Users/KevinMoran/dev/ui-screen.png`

*Examples of the output files generated by this tool are included in this download under the "Example-Output" folder.

3) Using the Android GUI-Verification Tool

3.1) Tool Description and Setup Procedure

The Android GUI-Verification Tool (GVT) is capable of detecting design violations between a design Mock-Up (taking the form of a Marketch web page which is generated from a Sketch image representation) and an actual app implementation. Currently, the tool can detect *component layout* and *component size* violations. The executable java file for the tool is called "`GVT-v1.0.jar`" To properly use the tool, the input files must be generated in the correct manner so that the tool can properly read the information, and perform GUI analysis. Please take the following steps to generate the proper input files:

3.1.1) Generating the Android UI-Dump and Screenshot Files

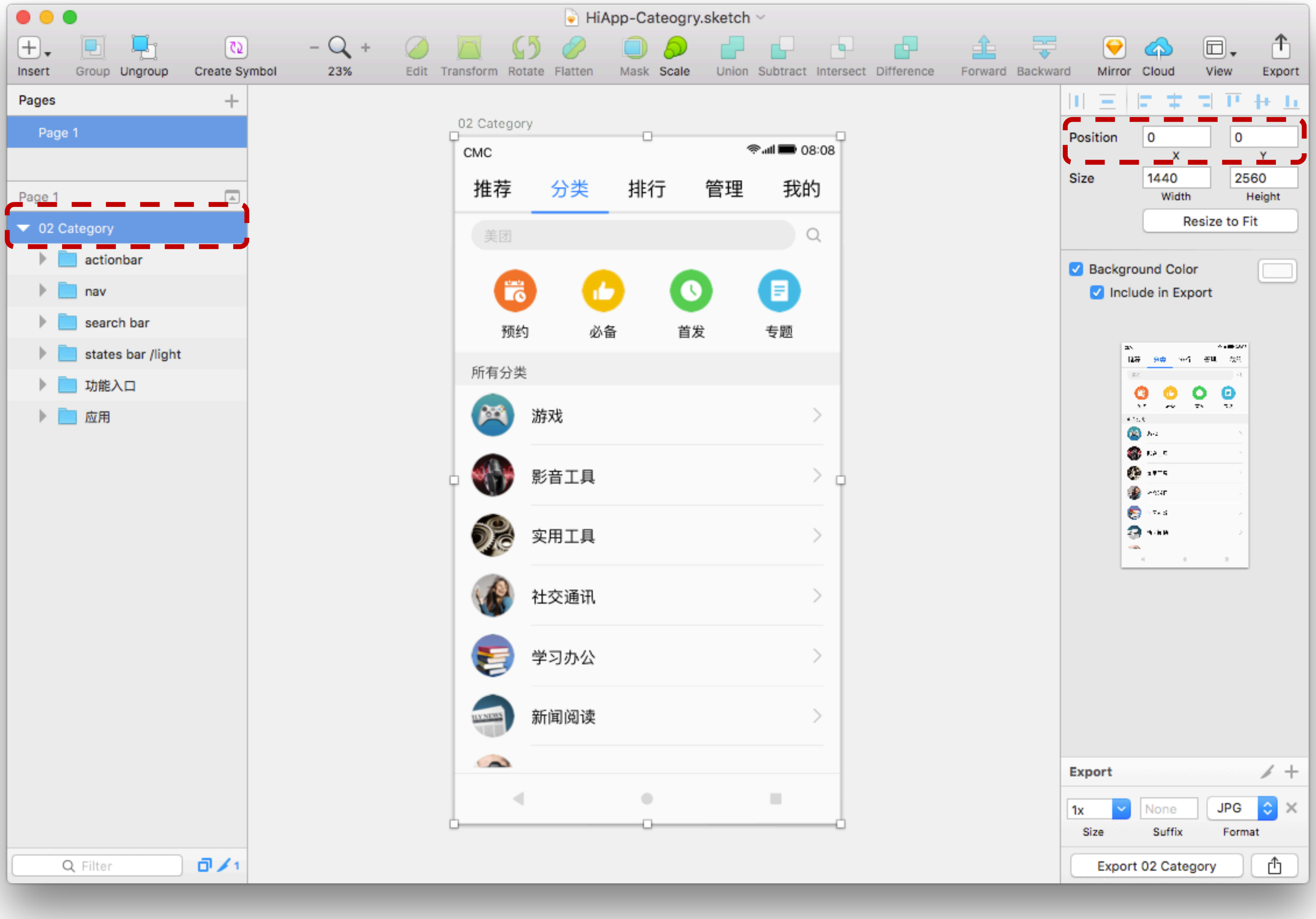
To generated the proper Android UI-Dump .xml and screenshot files, please follow the instructions in Section 2 of this documentation.

3.1.2) Generating the Proper Marketch File

Currently, the GVT only accepts a Marketch file with information corresponding to ***only one*** screen, or artboard. Therefore, the following steps should be taken to properly export the Marketch html from a corresponding Sketch File:

1) Open the Sketch file that contains the Design Mock-Up of a screen that you wish to analyze.

2) Select the artboard that you wish to analyze and ensure that the starting position for the Sketch file starts at position x=0 and y=0.



Position 0 X 0 Y

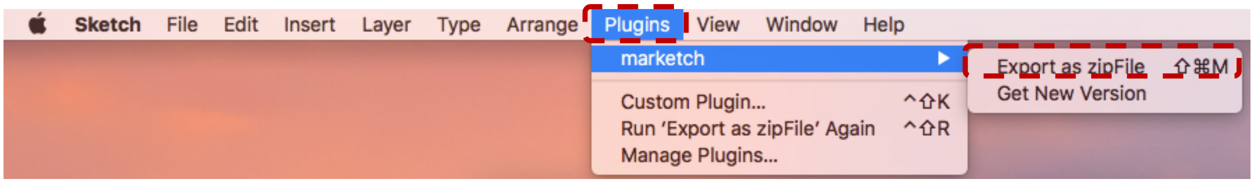
Size 1440 Width 2560 Height

☑ Background Color

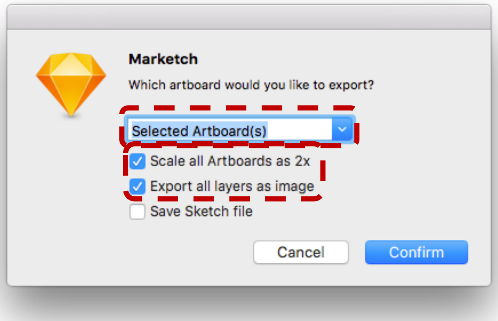
☑ Include in Export

Export: 1x Size, None Suffix, JPG Format

3) Next Navigate to the Plugins Dropdown Menu in the menu bar, and select “Marketch”→ “export as zip File”.



4) Next, the dialog box that appears, choose the “Selected Artboard(s)” option from the drop-down menu, and ensure that “Scale all Artboards as 2x” and “Export all layers as image”.



Marketch

Which artboard would you like to export?

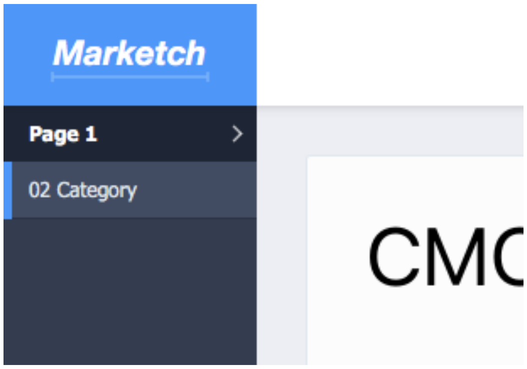
Selected Artboard(s)

☑ Scale all Artboards as 2x

☑ Export all layers as image

□ Save Sketch file

5) Finally, choose a file location for the zip file and complete the export process. After the export process is complete, extract the contents of the zip file and open the index.html file in a web-browser. The left-hand menu of the Marketch page should look like the following example, only containing one page and the screen in question:



2.2) GUI-Verification Tool Usage information

The usage for the GVT is the following:

```
java -jar GVT.jar GVT [Path to Properties json File] [Path to target Marketch
Folder] [Path to target Mock-Up Image] [Path to Android UI-dump xml file]
[Path to Android screenshot] [Path to source folder] [Path to output folder]
```

A description of the program arguments is as follows:

Path to Properties `json` File: This is a path to json properties file that contains some modifiable configuration information for the GVT. This file is included in the tool download, and is configured with the default values. The json structure and explanation of values can be found at the end of the documentation.

Path to Target Marketch Folder: This is the full file path to the unzipped Marketch folder for a design screen to be analyzed. For example: `/Users/KevinMoran/Marketch/HiApp-Category/`

<u>Path to target Mock-Up Image:</u> This is the path to a full image of the mock-up screen to be analyzed. Note that this image must have the same image dimensions as the Android screenshot listed below.

<u>Path to Android UI-dump xml File:</u> This is the full file path to the extracted Android UI-dump .xml file. For example: `/Users/KevinMoran/dev/ui-dump.xml`

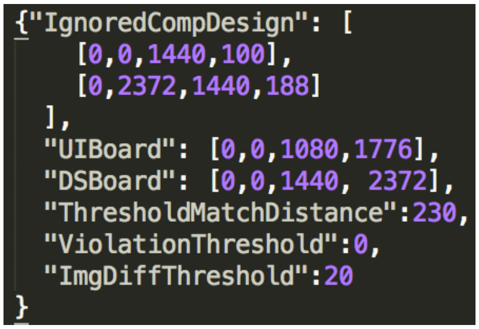
<u>Path to Android Screenshot:</u> This is the full file path to the extracted Android screenshot .png file. For example: `/Users/KevinMoran/dev/ui-screen.png`

<u>Path to Source Folder:</u> This is the full file path to the `/res` source code folder of the Android project's source code.

<u>Path to Output Folder:</u> This is the full file path to the folder where you would like the output of the GVT to be saved.

*Example Input and Output Files are included in the tool download in the “Example-Input” and “Example-Output” directories.

<u>Information and structure of the GVT properties.json file</u>:



```
{"IgnoredCompDesign": [
    [0,0,1440,100],
    [0,2372,1440,188]
  ],
  "UIBoard": [0,0,1080,1776],
  "DSBoard": [0,0,1440, 2372],
  "ThresholdMatchDistance":230,
  "ViolationThreshold":0,
  "ImgDiffThreshold":20
}
```

A description of the `json` fields are as follows:

<u>IgnoredCompDesign:</u> This field is an array of values that represent the location, height, and width of components that should be discarded from the GVT's analysis. For example, typically we do not include the status bar or the navigation buttons in our analysis. The values for each component correspond to the following structure: [x value of top-left corner, y value of top-left corner, component width, component height].

<u>UIBoard:</u> This field corresponds to the dimensions for the target Android device that the UI-Dump and screenshot were collected from. This field uses the same structure as above: [x value of top-left corner, y value of top-left corner, component width, component height].

<u>DSBoard:</u> This field corresponds to the dimensions for the Design Mock-Up (Marketch). This field uses the same structure as above: [x value of top-left corner, y value of top-left corner, component width, component height].

<u>ThresholdMatchDistance:</u> This is the value that should be used as the maximum distance between components that will be used to consider two components a match.

<u>ViolationThreshold:</u> This is the threshold that is used to detect design violations. If the difference between the layout position or size of two components is within the tolerance threshold, then a violation will not be reported.

ImgDiffThreshold: This is the threshold represented as a percentage that dictates when an Image Violation should be reported based on the percentage of difference pixels observed by the Perceptual Image Differencing CV technique.