**ADB(Android Debug Bridge)**

Android Debug Bridge (adb) is a versatile command-line tool that lets you communicate with a device. The adb command facilitates a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell that we can use to run a variety of commands on a device.

adb is mostly used by developers and also it can be used by people like us that are test engineers.

It is a client-server program that includes three components:

1. **A client:** which sends commands. The client runs on our development machine. We can invoke a client from a command-line terminal by issuing an adb command.
2. **A daemon:** which runs commands on a device. The daemon runs as a background process on each device.
3. **A server**: which manages communication between the client and the daemon. The server runs as a background process on your development machine.

**How adb works**

When you start an adb client, the client first checks whether there is an adb server process already running. If there isn't, it starts the server process. When the server starts, it binds to local TCP port 5037 and listens for commands sent from adb clients—all adb clients use port 5037 to communicate with the adb server.

The server then sets up connections to all running devices.

It locates emulators by scanning odd-numbered ports in the range 5555 to 5585, the range used by the first 16 emulators. Where the server finds an adb daemon (adbd), it sets up a connection to that port.

Note that each emulator uses a pair of sequential ports — an even-numbered port for console connections and an odd-numbered port for adb connections.

For example:

Emulator 1, console: 5554  
Emulator 1, adb: 5555  
Emulator 2, console: 5556  
Emulator 2, adb: 5557  
and so on...

As shown, the emulator connected to adb on port 5555 is the same as the emulator whose console listens on port 5554.

Once the server has set up connections to all devices, you can use adb commands to access those devices. Because the server manages connections to devices and handles commands from multiple adb clients, you can control any device from any client (or from a script).

so here what we all need for adb is a pc or and a usb cable your phone with usb debugging enabled also some commands to execute

**Configuration of ADB Client and server**

* Download and install Minimal ADB and Fastboot Tool

URL: https://androidmtk.com/download-minimal-adb-and-fastboot-tool

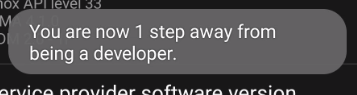
* Download and install Samsung USB Driver or Google USB Driver

URL: https://developer.samsung.com/mobile/android-usb-driver.html

URL: https://developer.android.com/studio/run/win-usb

**Configuration of Daemon**

* Enable Developer Option
* Go to settings
* Click on About phone
* Click on Software information
* Click on Build number(for multiple times)

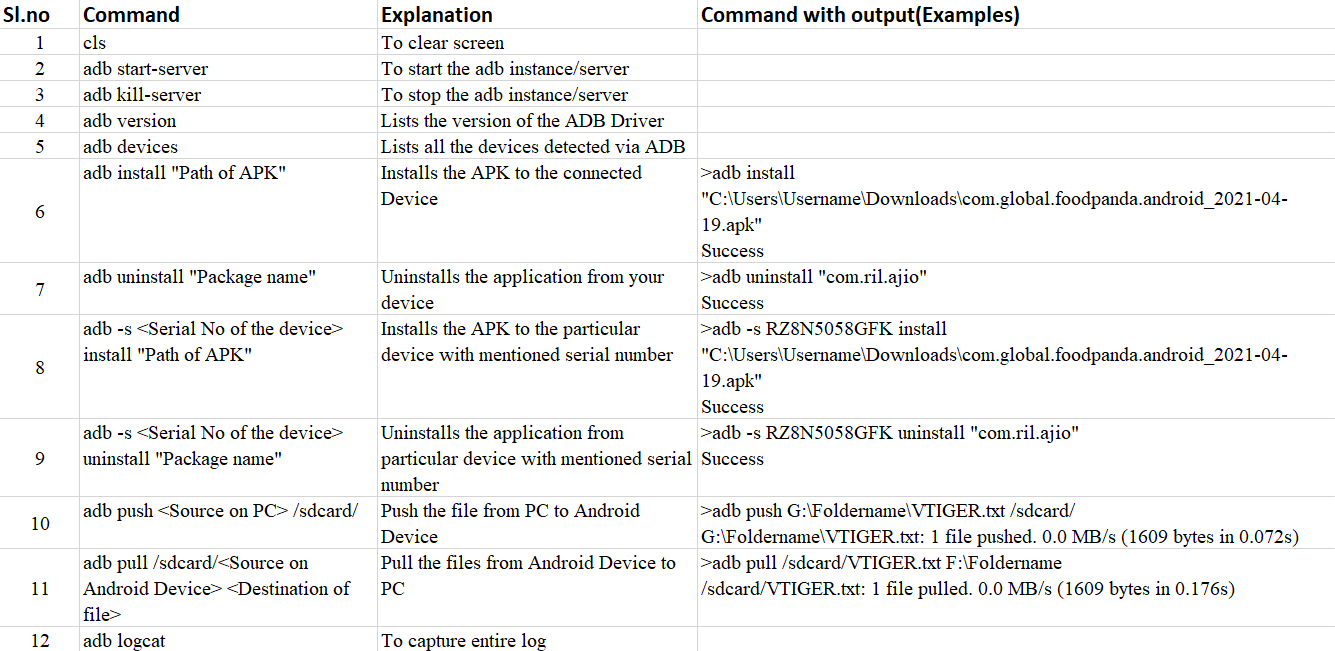


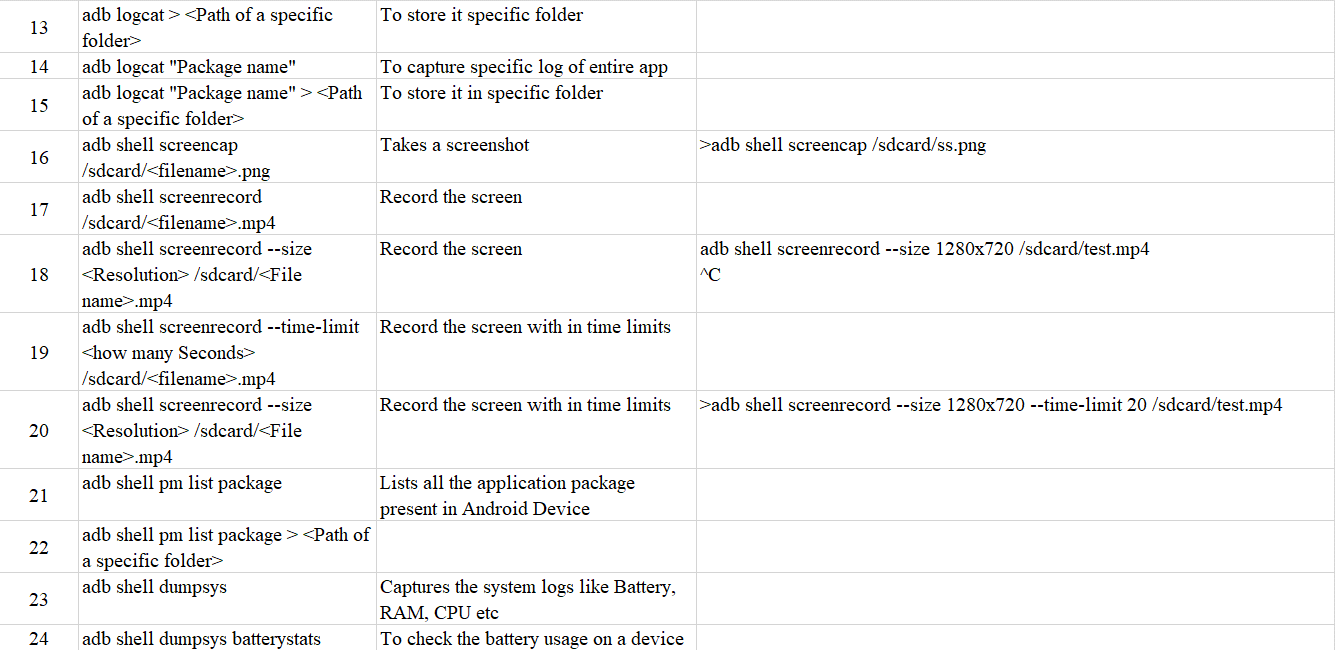
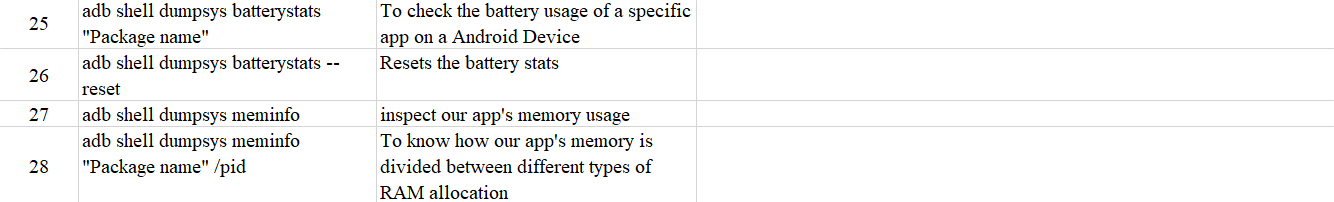
* Enable USB Debugging
* After Enabling Developer option
* Click on Developer option
* Enable USB Debugging

**ADB Commands**

**adb logcat -s <package name> → to filter the o/p**

**adb logcat -c → to clear the log**



### **The Top 25 Mobile Application Testing Scenarios :**

* Ensure the app has been launched by downloading and installing it for use.
* Verify that the mobile app display is adaptable to the device screen and also ensures all menus on the app are functioning.
* Verify that the text on the mobile app is readable and clear.
* Check that the app display is adaptable and amenable to the various display mode (i.e. landscape and portrait).
* Verify that the app does not stop the functioning of other apps on the mobile device.
* Verify that in the play screen, the back key allows the app to go back to the start-up screen.
* Check that the app still operates as intended, if the device resumes from inactive mode or the lock screen.
* Check whether the app reminds the user to save setting changes or changing of information before moving to other activities on the app.
* Verify that the on-screen keyboard appears immediately the user attempt to enter a text.
* Check if the app behaves as designed if the mobile device is shaken.
* Verify that the app still functions as designed when “battery low” notifications appear on the screen.
* Check that the app goes into the background when on call.
* Check that the app still operates as designed when a message or notification pop-up from another app such as Facebook messaged, Instagram, etc.
* If the app comes with a users’ settings features, check if the app changes when some form of change is affected by the user.
* Check the [**Performance**](https://www.testrigtechnologies.com/service/performance-testing/)of the app on the different internet networks such as 1G, 2G, 3G, or 4 G networks.
* Check that the app operates as intended when the device is connected to the internet through WiFi.
* Check that the app still operates normally when there is an incoming call or SMS.
* Check that the app is adaptable to different mobile platforms or OS such as Android, iOS, Microsoft, etc.
* Check that the font size and style of the app are compatible and readable to the users.
* Verify that that the loading time for the app is not too long.
* Check that the app is still working as intended after the successful update of the app.
* Check how the app function under different battery levels and temperatures.
* Verify that the app is not draining too much battery.
* Check that the app support image capturing.
* Check that the app does not log out the user before the end of a session.