ADB COMMANDS

---

adb

This is a nude command that can be used to get the ADB version and all possible commands associated with ADB.

adb -s

When many devices are connected to a computer, you can redirect ADB commands to a specific device using this command.

```
adb -s <deviceName> <command>
```

adb -d

This command directs the command to a device connected via USB.

```
adb -d <command>
```

adb -e

You can use this command to direct ADB commands to a connected emulator.

```
adb -e <command>
```

adb devices //show devices attached

This is similar to the ‘adb devices’ command that shows you the list of connected Android devices and emulators to your computer.

adb connect ip-address-of-device

If you want to find out and connect the IP address of your Android device to your PC. You can use this ADB command to **set up ADB over Wi-Fi**.

adb help

You can use the `adb help` command to display the help documentation on all ADB commands.

adb root

This command restarts `adbd` with root permissions.

adb version

Use this command to find out the version of the ADB driver installed on your computer.

adb reboot

As this very ADB command suggests, ‘adb reboot’ command can help reboot your Android phone or tablet into the bootloader, fastboot, or recovery mode in case the hardware keys of your device do not function properly.

adb reboot bootloader

You can use this command to reboot your Android device into the Fastboot or Bootloader Mode. The command is often used when you want to flash the factory images or a custom recovery.

adb reboot download

You can use this ADB command to boot Samsung devices into the Download Mode or Odin Mode.

adb reboot recovery

This is one of the most used ADB commands. You can execute it to reboot your device into Android Recovery mode.

adb install

Usually, we transfer an APK file to our Android device and install it via File Manager. Thanks to the ‘adb install’ command that it makes it easy to install APK files on Android devices directly from your computer. To be able to do so, copy the APK file to the SDK platform-tools folder first. There are some variations of this command for different conditions.

```
adb install com.facebook.katana.apk
```

While you can install an APK on your Android devices using the above command, you can use the command given below to update or reinstall an app without deleting its data.

```
adb install -r com.facebook.katana.apk
```

Some apps support installation on the SD card. By using the following ADB commands, move an app to the SD storage.

```
adb install -s com.facebook.katana.apk
```

```
adb install –k <add the path of the .APK file on your computer>
```

adb uninstall

Using this command in the ADB terminal windows, you can uninstall an app from your phone or tablet.

```
adb uninstall com.facebook.katana
```

If you want to uninstall an app but keep its data and cache files, you can use the ‘adb uninstall’ command with ‘**-k**‘ parameter as shown below.

```
adb uninstall -k com.facebook.katana
```

adb usb

If you want to find out the Android device or emulators currently connected to your Windows, Mac, or Linux computer, you can use the ‘adb usb’ command.

adb logcat

By executing the ‘adb logcat’ command, you can see the log data of your Android device on your computer. There are a few variations of this command with ‘**-c**‘ and ‘**-d**‘ parameters.

```
adb logcat -c // clear //
```

While using the above command you can clear all existing logs on your Android phone or tablet, you can save the logcat data on your PC with the following command.

```
adb logcat -d > [path_to_file] //
```

adb start-server

This is a useful command that lets you start the adb server in case it stops responding. It’s often used after killing the adb server as described below.

adb kill-server

If the ADB terminal is not functioning properly, you can try killing the ADB server. It’s similar to turning off our Android devices to fix small issues.

adb sideload

It’s one of the most popular ADB commands as it can be used to sideload software **update.zip** files using your computer. If you have downloaded a flashable update.zip, just copy it to the ‘**platform-tools**‘ folder and execute the following command. If the zip file you download has a different name, rename it to”update.zip” for the sake of convenience.

```
adb sideload update.zip
```

Alternatively, you can sideload or flash update zip packages using Android stock recovery or TWRP recovery as well. For that, you will have to transfer the update.zip file to your device, reboot your Android into recovery mode, and select the “**Apply update from sdcard**” option

adb pull

You can download or pull files stored on your Android device to your computer using the following ADB command.

This command can be used to pull any files from your device and save them to the ‘platform-tools’ folder on your computer.

```
adb pull /sdcard/video-01.mp4
```

In case you want to pull the file to a specific location or drive (D drive, for instance) on your PC, you issue the following command mentioning the path of the storage location as shown below.

```
adb pull /sdcard/video-01.mp4 d:\
```

adb push

Similarly, this command can be used to push a file from your computer to your device. Please note that you'll have to transfer the file you want to push tp the 'platform-tools' directory first. To push a file to your Android device's SD card, for example, use the following code.

```
adb push com.whatsapp_2.19.368-453132.apk /sdcard
```

In case you want to send a file to your device stored in a specific location, you can use the below command instead.

```
adb push d:\com.whatsapp_2.19.368-453132.apk /sdcard
```

adb backup //

To create or take a full backup of your Android to your computer, try this command.

adb restore //

You can also restore the backup you have already created using this command.

adb bugreport

This command is best suited if you want to diagnose any issue on Android devices. Its execution can show you log data, dumpstate, and dumpsys from your Android device on your computer.

adb jdwp

JDWP means Java Debug Wire Protocol. By using this ADB command, you can see the list of JDWP processes on your PC.

adb get-state

Execute this command to print the device state in the command window.

adb get-serialno

If you want to find out the ADB instance serial number, you can use this command.

adb get-state

Shows the ADB status of a connected device or emulator.

adb wait-for-device

This program tells ADB that it has to wait and keep the connection on hold until the next command is issued.

adb shell pm uninstall

This is a very useful ADB Shell command. Using this, you can easily uninstall unwanted system apps. To be able to execute it, you must issue `adb shell` command first. You can then use `pm uninstall -k --user 0` or `pm uninstall --user 0` followed by the Android app package name as shown below.

```
pm uninstall -k --user 0 com.facebook.appmanager
```

`-k`: Keep the app data and cache after package removal. If you want the app data to be cleared as well, use the following

```
pm uninstall --user 0 com.android.chrome
```

If you don't know the app package name for the apps you want to remove, you can use `adb shell pm list packages` to find it.

This command can help you if you want to remove all bloatware from your Android phone. Please note that most system apps don't have the '**Uninstall**' option on the device but this command works magically.

adb shell cmd package install-existing

Using the above command, you can re-install an uninstalled system app.

```
cmd package install-existing com.facebook.appmanager
```

adb shell pm disable-user <package-name>

If you want to disable a system app on your Android device, you can execute the above command followed by the app package name

```
pm disable-user --user 0 com.google.ar.core
```

adb shell pm clear <package-name>

Using this command, you can delete all data associated with an app.

```
adb shell pm clear --user 0 com.facebook.appmanager
```

adb shell pm hide <package-name>

In case you want to hide an installed app on your Android device, you can execute this command line followed by the app package name.

```
adb shell pm hide --user 0 com.whatsapp
```

adb shell pm list packages

Using the above ADB Shell command, you can print the list of the app package names for all apps installed on your Android device. You can use this command with different parameters to get a more specific list of app packages.

For instance, if you want to list the system apps only, use

```
adb shell pm list packages -s
```

The following command will print the list of all installed packages with the path of the APKs.

```
adb shell list packages -r
```

To list all third-party apps installed on your Android phone or tablet, you issue the following command.

```
adb shell pm list packages -3
```

Do you want ADB Shell to show the list of all enabled or disabled apps on your device, try the command with parameters like '**-d**' (for disabled apps), '**-e**' (for enabled apps), and '**-u**' (for uninstalled apps).

```
adb shell pm list packages -d
```

```
adb shell pm list packages -e
```

```
adb shell pm list packages -u
```

The following command will list the app packages with their installers.

```
adb shell pm list packages -i
```

To list app packages with specific keyword filters.

```
adb shell pm list packages <keywords>
```

To find the list of apps along with their associated packages, execute the following command

```
adb shell pm list packages -f
```

You can easily get a list of group packages by a certain manufacturer, or some common term. For instance, if you want to list all apps by Google, you can use the following command.

```
adb shell pm list packages | grep 'google'
```

You can replace 'google' with 'samsung', 'huawei', 'xiaomi', 'miui', 'evenwell', 'android', 'facebook', etc. to get the desired list of packages. You can refer to our detailed tutorial on removing bloatware from Android devices using ADB for more information.

adb shell pm path <package-name>

This command displays the APK path on the device's file system.

adb shell pm create-user

You can use this command to create a new user on your Android device.

```
adb shell pm create-user username
```

adb shell pm remove-user

Just in case you want to remove a user from your device, you can use the above command followed by the `user_id` as shown below.

```
adb shell pm remove-user user 1
```

adb shell pm get-max-users

By using this command, you can print the maximum number of users supported on an Android device.

adb shell pm list features

Use the above command to print all supported features of the system.

adb shell pm list permissions

This command prints the list of all known permissions, optionally only those in `group`. You can use it with the following parameters.

- **`-g`**: Organize permissions by group
- **`-f`**: Print all information
- **`-s`**: Short summary of permissions
- **`-d`**: List dangerous permissions only
- **`-u`**: List the permissions seen by users only

```
adb shell pm list permissions -d group
```

adb shell pm path

Get the path of a given app package.

```
adb shell pm path <package-name>
```

adb shell settings

You can use this command to print information about specific settings on your Android device. By adding different parameters, you can find out the Android settings provider, current system volume level, notification sound, device ID, Bluetooth MAC address, current mobile data status, current Wi-Fi status, etc.

- **`adb shell settings list system`**
- **`adb shell settings get system volume_system`**
- **`adb shell settings get system notification_sound`**
- **`adb shell settings list secure`**
- **`adb shell settings get secure android_id`**
- **`adb shell settings get secure bluetooth_address`**
- **`adb shell settings list global`**
- **`adb shell settings get global mobile_data`**
- **`adb shell settings get global wifi_on`**

adb shell dumpsys

It’s a very flexible command that can be used standalone or with various parameters to get data related to battery, display, CPU, RAM, storage, etc. The execution of this command will give you detailed information about the Android device’s software and hardware configuration.

Note: To use this tool don’t forget to add permission to your Android manifest automatically `android.permission.DUMP`

```
adb shell dumpsys
```

Other variations of the command are as follows:

- **adb shell dumpsys package packages (get info about all installed apps)**
- **adb shell dumpsys input**
- **adb shell dumpsys display (get details about the display)**
- **adb shell dumpsys battery** (get detailed info about your device's battery and status)
- **adb shell dumpsys batterystats** (battery usage statistics)
- **adb shell dumpsys activity** (get a complete list of all ongoing activities on your device)
- **adb shell dumpsys cpuinfo** (get detailed about CPU usage)

Executing the `adb shell dumpsys cpuinfo` command, for instance, will print a list of CPU usage by the running processes and apps on your Android device.adb shell wm density
The above command can be used to find out the pixel density of your Android device’s display.

adb shell dumpsys window displays

The above command will print very detailed info like pixel resolution, FPS, and DPI of your phone’s display.

adb shell wm size

You can find out the display resolution of your phone with this command.

```
PS C:\Users\Technastic\Desktop> adb shell wm size
Physical size: 1440x3040
Override size: 1080x2280
```

If you want to modify the screen resolution and the pixel density of your Android device’s display. If you’re not sure about your device’s display resolution, execute the command given below. Suppose your phone’s display resolution is QHD+, you can easily change it to Full HD+ or HD+.

- **FHD**

```
adb shell wm size 1080x2220

adb shell wm density 420
```

- **HD**

```
adb shell wm size 720x1560
```

```
adb shell wm density 360
```

adb shell screencap

By using this command, you can capture a screenshot and download it to your computer using the ‘adb pull’ command as described above.

```
adb shell screencap /sdcard/screenshot-01.png
```

adb shell screenrecord

ADB also lets you record your phone or tablet’s screen and download the recorded video to your computer. Besides, you can also set conditions like video duration, resolution in pixels, video bitrate, etc.

```
adb shell screenrecord /sdcard/screenrecord-01.mp4
```

```
adb pull screenrecord /sdcard/screenrecord.mp4
```

You can stop screen recording using **Ctrl+C**. In case you want to record the screen in a specific resolution, the following command lets you set custom width and height in pixels.

```
adb shell screenrecord --size 1920x1080 /sdcard/screenrecord-01.mp4
```

By default, Android’s screen recorder’s duration is set to 180 seconds (3 minutes). You can decrease this time limit according to your needs (180 seconds is the maximum limit).

```
adb shell screenrecord --time-limit 120 /sdcard/screenrecord-01.mp4
```

Similarly, you can also determine the bitrate of the video output. To set the bitrate to 4MBPS, for example, you can use the following value:

```
adb shell screenrecord --bit-rate 6000000 /sdcard/screenrecord-01.mp4
```

adb shell getprop & adb shell setprop

The ‘**getprop**‘ and ‘**setprop**‘ commands can be used to view and set or change the configuration of the ‘build.prop’ file on Android devices. The following command, for example, displays the Android system information.

```
adb shell getprop
```

Below are some more examples:

```
getprop ro.build.version.sdk
```

```
getprop ro.chipname
```

In case you want to change the value of an entry in the build.prop, you can use the `adb shell setprop` commands. See the examples below:

```
getprop net.dns1 1.2.3.4
```

```
setprop net.dns1 1.3.4.5

getprop net.dns2 1.1.2.3

setprop net.dns2 1.2.3.4
```

In the same way, if you want to change the configuration of the VMHeap size on your Android device, you can use the following command.

```
setprop dalvik.vm.heapsize 60m
```

There are some more variations of the `adb shell getprop` command that let you see information about Android system properties, SDK API level, Android security patch version, Soc, Android version, device model, device manufacturer, ADB serial number, OEM unlock status, Android device build fingerprint, WiFi MAC address, etc.

- **adb shell getprop**
- **adb shell getprop ro.build.version.sdk**
- **adb shell getprop ro.build.version.security_patch**
- **adb shell getprop ro.board.platform**
- **adb shell getprop ro.build.version.release**
- **adb shell getprop ro.vendor.product.model**
- **adb shell getprop ro.product.manufacturer**
- **adb shell getprop ro.serialno**
- **adb shell getprop ro.oem_unlock_supported**
- **adb shell getprop ro.bootimage.build.fingerprint**
- **adb shell getprop ro.boot.wifimacaddr**

adb -s shell getprop

If you want to check the full configuration, running services, and information about your Android phone or tablet, you can use the above command. First off, run the `adb devices` command and copy the alpha-numeric value of your device ID from the output.

```
PS C:\Users\Technastic\Desktop> adb devices
List of devices attached
RZ8M810BARJ device
```

Then execute the following command. Don’t forget to replace the device ID highlighted in blue with the ID of your device.

```
adb -s RZ8M810BARJ shell getprop
```

adb shell cat /proc/cpuinfo

Use the above command to get complete information about the CPU on your phone or tablet.

Manage App Permissions using Command

Reset Permissions

```
adb shell pm reset-permissions -p <app-package-name>
```

Grant Permissions

```
adb shell pm grant <app-package-name> <permission-name>
```

Revoke Permissions

```
adb shell pm revoke <app-package-name> <permission-name>
```

Get an Android device properties

By running the following command, you can see the system properties.

```
adb shell getprop | grep -e 'model' -e 'version.sdk' -e 'manufacturer' -e
'hardware' -e 'platform' -e 'revision' -e 'serialno' -e 'product.name' -e
'brand'
```

adb shell cd

Change the ADB shell directory using '*cd <directory>*'

```
adb shell
```

Then execute the following command:

```
cd /system
```

adb shell rm

This command lets you easily delete a file or folder from your Android device's storage. Launch the command window, execute the 'adb shell' command, and then try the following command with '**-f**' (to delete a file) and '**-d**' (to remove a directory) parameters.

```
rm -f /sdcard/com.whatsapp.apk
```

```
rm -d /sdcard/WhatsApp
```

Note: Instead of '**rm-d**', you can also use '**rmdir**'.

adb shell mkdir

Besides deleting an existing directory or folder, ADB Shell also lets you create a new directory or sub-directory. Not only that, you can set permissions for the newly created folder.

```
mkdir /sdcard/NewFolder
```

```
mkdir -p /sdcard/NewFolder/NewFolder1
```

```
mkdir -m 644 /sdcard/NewFolder
```

adb shell cp

‘**cp**‘ stands for ‘copy’. You can use this command to copy files and directories located on your Android device. Again, you need to start with the `adb shell` command first.

To copy files and then paste them, mention the source and destination locations as shown below:

```
cp /sdcard/OPWallpaperResources.apk /sdcard/DCIM/Camera
```

adb shell mv

‘**mv**‘ stands for ‘move’. This command can be used to **move a file** stored on your device from a source location to a destination location.

```
mv /sdcard/livewallpapers.apk /system/app
```

The following command will allow you to move a file with a new name.

```
mv /sdcard/livewallpapers.apk /sdcard/Wallpapers
```

adb shell top

To display the list of top CPU processes on an Android phone or tablet, you can use the above command. CPU processes monitor can be stopped using **Ctrl + C**.

adb shell ip

Find out the WiFi IP address of an Android phone or tablet.

```
ip -f inet addr show wlan0
```

adb shell netstat

Displays the network statistics of Android phones.

```
adb shell netstat
```

Get a List of an Android Device’s Features

```
adb shell pm list features
```

ADB Shell Command to Make a Call

To make a call using an ADB Shell command from your Android phone, you can use the following command.

```
adb shell am start -a android.intent.action.CALL -d tel:+19797220011
```

ADB Shell command to Send SMS screen

If you want to send a text message using a command, try the following code.

```
adb shell am start -a android.intent.action.SENDTO -d sms:+19797220011 --es sms_body "Test Message" --ez exit_on_sent false
```

You can use the following command to

Input Text in a Text Field

You can input or print text on your phone using the command. I tested this command in apps like Messages, WhatsApp, Facebook, etc. If such apps are not open, the text will open in Google Search.

```
adb shell input text 'I love this adb command'
```

ADB Command to Open a URL in Web Browser

You can open a URL in your Android phone's default web browser using the following command.

```
adb shell am start -a android.intent.action.VIEW -d https://technastic.com
```

Command to Open the Gallery App

Executing the following ADB Shell command will open the Gallery app on your Android device.

```
adb shell am start -t image/* -a android.intent.action.VIEW
```

ADB Shell Key Event Commands

Key Event Commands to Toggle and Trigger Functions

Android devices support KeyEvent commands that can let you perform certain actions that require you to press a hardware button or tap an app or UI option. You can control your Android phone or tablet device and even launch apps by using these KeyEvent commands. These commands might come in handy if the hardware keys on your device are not functioning due to some damage.

- **Turn Android device ON or OFF:** `adb shell input keyevent 2`
- **Press Home button:** `adb shell input keyevent 3`
- **Press Back button:** `adb shell input keyevent 4`
- **Press Call button:** `adb shell input keyevent 5`
- **End a call:** `adb shell input keyevent 6`
- **Press Power Button to wake up screen:** `adb shell input keyevent 26`
- **Turn ON the camera:** `adb shell input keyevent 27`
- **Open web browser:** `adb shell input keyevent 64`
- **Press the Enter key:** `adb shell input keyevent 66`
- **Press Backspace button:** `adb shell input keyevent 67`
- **Open Contacts app:** `adb shell input keyevent 207`

- **Decrease display brightness:** adb shell input keyevent 220
- **Increase Display brightness:** adb shell input keyevent 221
- **Cut text:** adb shell input keyevent 277
- **Copy text:** adb shell input keyevent 278
- **Paste text:** adb shell input keyevent 279
- **Make the device sleep**: adb shell input keyevent KEYCODE_SLEEP
- **Make device wakeup**: adb shell input keyevent KEYCODE_WAKEUP
- **Toggle Power menu**: adb shell input keyevent KEYCODE_POWER
- **Trigger Volume up**: adb shell input keyevent 24
- **Trigger Volume down**: "adb shell input keyevent 25
- **Trigger Zoom in**: adb shell input keyevent 168
- **Trigger Zoom out**: adb shell input keyevent 169
- **Menu**: adb shell input keyevent 82
- **Open Notifications**: adb shell input keyevent 83
- **Launch Search**: adb shell input keyevent 84
- **Play or pause media**: adb shell input keyevent 85
- **Mute audio**: adb shell input keyevent 91
- **Page up**: adb shell input keyevent 92
- **Page down**: adb shell input keyevent 93
- **Open Calendar**: adb shell input keyevent 208
- **Launch Music**: adb shell input keyevent 209
- **Open Calculator**: adb shell input keyevent 210
- **Volume up**: adb shell input keyevent 24
- **Make the lock screen sleep**: adb shell input keyevent 223
- **Wakeup the lock screen**: adb shell input keyevent 224
- **Make device wakeup**: adb shell input keyevent KEYCODE_WAKEUP
- **Toggle Power menu**: adb shell input keyevent KEYCODE_POWER

Android Keyboard-related KeyEvent Commands

You can use the following key events commands with ADB Shell to print letters, numbers, and symbols on your Android device right from your computer.

- **Clear**: `adb shell input keyevent 28`
- **Caps lock**: `adb shell input keyevent 115`
- **Number '0'**: `adb shell input keyevent 7`
- **Number '1'**: `adb shell input keyevent 8`
- **Number '2'**: `adb shell input keyevent 9`
- **Number '3'**: `adb shell input keyevent 10`
- **Number '4'**: `adb shell input keyevent 11`
- **Number '5'**: `adb shell input keyevent 12`
- **Number '6'**: `adb shell input keyevent 13`
- **Number '7'**: `adb shell input keyevent 14`
- **Number '8'**: `adb shell input keyevent 15`
- **Number '9'**: `adb shell input keyevent 16`
- **Letter 'a'**: `adb shell input keyevent 29`
- **Letter 'b'**: `adb shell input keyevent 30`
- **Letter 'c'**: `adb shell input keyevent 31`
- **Letter 'd'**: `adb shell input keyevent 32`
- **Letter 'e'**: `adb shell input keyevent 33`
- **Letter 'f'**: `adb shell input keyevent 34`
- **Letter 'g'**: `adb shell input keyevent 35`
- **Letter 'h'**: `adb shell input keyevent 36`
- **Letter 'i'**: `adb shell input keyevent 37`
- **Letter 'j'**: `adb shell input keyevent 38`
- **Letter 'k'**: `adb shell input keyevent 39`
- **Letter 'l'**: `adb shell input keyevent 40`
- **Letter 'm'**: `adb shell input keyevent 41`
- **Letter 'n'**: `adb shell input keyevent 42`

- **Letter 'o'**: adb shell input keyevent 43
- **Letter 'p'**: adb shell input keyevent 44
- **Letter 'q'**: adb shell input keyevent 45
- **Letter 'r'**: adb shell input keyevent 46
- **Letter 's'**: adb shell input keyevent 47
- **Key 't'**: adb shell input keyevent 48
- **Letter 'u'**: adb shell input keyevent 49
- **Letter 'v'**: adb shell input keyevent 50
- **Letter 'w'**: adb shell input keyevent 51
- **Letter 'x'**: adb shell input keyevent 52
- **Letter 'y'**: adb shell input keyevent 53
- **Letter 'z'**: adb shell input keyevent 54
- **Key 'comma'**: adb shell input keyevent 55
- **Key 'period'**: adb shell input keyevent 56
- **Key 'alt_left'**: adb shell input keyevent 57
- **Key 'alt_right'**: adb shell input keyevent 58
- **Key 'shift_left'**: adb shell input keyevent 59
- **Key 'shift_right'**: adb shell input keyevent 60
- **Key 'tab'**: adb shell input keyevent 61
- **Key 'space'**: adb shell input keyevent 62
- **Key 'symbols'**: adb shell input keyevent 63
- **Key 'minus'**: adb shell input keyevent 69
- **Key 'equals'**: adb shell input keyevent 70
- **Key 'left bracket'**: adb shell input keyevent 71
- **Key 'right bracket'**: adb shell input keyevent 72
- **Key 'backslash'**: adb shell input keyevent 73
- **Key 'semicolon'**: adb shell input keyevent 74
- **Key 'apostrophe'**: adb shell input keyevent 75
- **Key 'slash'**: adb shell input keyevent 76

- **Key '@'**: adb shell input keyevent 77

The following key events work on the numerical keypad of Android devices.

- **Number '0'**: adb shell input keyevent 144
- **Number '1'**: adb shell input keyevent 145
- **Number '2'**: adb shell input keyevent 146
- **Number '3'**: adb shell input keyevent 147
- **Number '4'**: adb shell input keyevent 148
- **Number '5'**: adb shell input keyevent 149
- **Number '6'**: adb shell input keyevent 150
- **Number '7'**: adb shell input keyevent 151
- **Number '8'**: adb shell input keyevent 152
- **Number '9'**: adb shell input keyevent 153
- **Symbol 'divide'**: adb shell input keyevent 154
- **Symbol 'multiply'**: adb shell input keyevent 155
- **Symbol 'subtract'**: adb shell input keyevent 156
- **Symbol 'add'**: adb shell input keyevent 157
- **Symbol 'dot'**: adb shell input keyevent 158
- **Symbol 'comma'**: adb shell input keyevent 159
- **Symbol 'enter'**: adb shell input keyevent 160
- **Symbol 'equals'**: adb shell input keyevent 16

Android Debug Bridge (adb) commands are very useful while debugging any Android device but at the same time it's difficult to find all the commands at one place.

In this article, you will find 200+ ADB, shell and fastboot commands which are commonly used for debugging Android devices. Using these commands can reduce the required efforts and can help achieve the required result in less time.

Below is the list of commands which can be executed from command prompt or terminal on Android devices while the device is connected with the computer with USB debugging turned on.

ADB commands to connect and list devices:

If anyone has used ADB in the past then they must have used 'adb devices' to list all the connected devices. However, it's not just limited to listing the devices, we can connect devices using wires or wirelessly and even control the ADB server. Here are the few commands which can help to connect and list the devices.

Use	Command
To List all connected devices via adb	adb devices
To List connected devices (-l for long output)	adb devices -l
To list all devices connected via USB or To restart adb device in USB mode	adb usb
To list all forward socket connections	adb forward –list
To set up port forwarding (sets up forwarding of computer port 6123 to Android device port 7123)	adb forward tcp:6123 tcp:7123
To start the ADB server process	adb start-server
To terminates the adb server process	adb kill-server
To Set the target device to listen for a TCP/IP connection on port 5555Connect a device using Wi-Fi network.Confirm that your host computer is also connected to the Android device over Wi-Fi Help Text: disconnect the USB cable from the Android devicefind your Android device IP address at Settings > About phone > Status > IP addressList of devices attached192.xxx.xxx.xx:5555 device	adb tcpip 5555adb connect 192.xxx.xxx.xxadb devices

ADB and shell commands to manage applications and get the application details:

The basic operations which everyone perform using ADB commands are to install and uninstall applications. ADB commands are also used to get the information of the applications installed on the device. Below are some ADB and shell commands to manage and get information about applications on the device.

Use	Command
To push a single application to the device and install.	adb install <apk_name.apk>
To push multiple applications to the device and install.	adb install-multiple <apkname.apk> <apkname2.apk>
To push multiple applications to the device and install it automatically.	adb install-multi-package <apkname.apk> <apkname2.apk>
To replace existing application by keeping its data.	adb install -r <apkname.apk>
To install apk in external storage	adb install -s <apkname.apk>
To allow test packages	adb install -t <apkname.apk>
To uninstall apk by package name	adb uninstall <package_name_of_apk>
To uninstall apk by package name while keeping cache and app data	adb uninstall -k <package_name_of_apk>
To clear application data by package name	adb shell pm clear <package_name_of_apk>
To uninstall system application by package name	adb shell pm uninstall -k –user 0 <package_name_of_apk>
To disable application by package name	adb shell pm disable <package_name_of_apk>
To disable application by package name for system user	adb shell pm disable-user –user 0 <package_name_of_apk>
To disable re-enable disabled application by package name	adb shell pm enable <package_name_of_apk>
To hide application by package name	adb shell pm hide <package_name_of_apk>
To unhide application by package name	adb shell pm unhide <package_name_of_apk>
To suspend any application by package name	adb shell pm suspend

	<package_name_of_apk>
To re-enable suspended application by package name	adb shell pm unsuspend <package_name_of_apk>
To list package name of all installed apks	adb shell pm list packages
To list package name of all system apps	adb shell pm list packages -s
To list package name of third party apps installed on the device	adb shell pm list packages -3
To list package name of disabled apps on the device	adb shell pm list packages -d
To list package name of only enabled apps on the device	adb shell pm list packages -e
To list package name of uninstalled apps from the device	adb shell pm list packages -u

ADB commands to reboot device:

While debugging an android device, rebooting the device or rebooting in the recovery mode or bootloader mode is required for operations like sideloading any image or application. Below are some commands to achieve the same.

Use	Command
To reboot the device	adb reboot
To reboot device in bootloader or fastboot mode	adb reboot bootloader
To reboot device in recovery mode	adb reboot recovery
To restart ADB device with root permission	adb root
To restart ADB device without root permission	adb unroot
To manually install OTA package using recovery mode	adb sideload ota-updatefile.zip

ADB commands for collecting logs and bug report:

Logs and Bug Reports contain useful information related to application and are important for developers while fixing bugs or making changes in the application. Here are some commands which are useful for taking bug reports and collecting logs.

Use	Command
To display full logcat data on the computer screen	adb logcat
To clear current logcat data	adb logcat -c
To save current logcat data in local file	adb logcat -d <path>
To generate bug report of connected device	adb bugreport <path>
To get the serial number of connected device	adb get-serialno
To wait for android device until the current process is completed	adb wait-for-device
To get the device state in the terminal/command prompt	adb get-state
To list JDWP (Java Debug Wire Protocol) processes of android device	adb jdwp

ADB commands to take backup and restore:

It’s always a good idea to take a backup of your device’s data so that it can be restored later whenever it’s required but if you are unlocking the bootloader or debugging the device then it’s highly recommended to take backup. Here are some commands which can help you with the same.

Use	Command

To take full backup of android device to computer	adb backup //
To restore backup from computer to device	adb restore //
To take backup of apps and settings	adb backup -apk -all -f backup.ab
To take backup of applications and application settings and shared storage	adb backup -apk -shared -all -f backup.ab
To take backup of only third party applications.	adb backup -apk -nosystem -all -f backup.ab
To connect to device using its ip address	adb connect ip_address_of_device
To restore previously created backup	adb restore backup.ab

ADB and shell commands to perform file operations:

Performing copy-paste using explorer requires a lot of time and also compared to ADB it's a little slow. You must have used ADB push and ADB pull for file operations but here are some more commands which can help you more with file operations.

Use	Command
To copy or transfer a file from a device's storage to a computer.	adb pull /system/app/<apkname.apk>
To transmit or push a file from a computer to a device	adb push /local/path/<apkname.apk> /sdcard/apps/
To copy files inside device storage from one folder to another folder.	adb shell<enter>cp /sdcard/<filename.extension> /sdcard/foldername
To move files inside device storage from one folder to another folder.	adb shell<enter>mv /sdcard/<filename.extension> /sdcard/foldername
To move file inside the device storage and rename the file in destination folder	adb shell<enter>mv /sdcard/<filename.extension> /sdcard/apps/<newfilename.extension>

Shell commands to get and change display resolution:

Information like screen resolution, pixel density and refresh rate are usually not getting displayed in the device and to get such information the user has to struggle a lot in Android settings and it's even difficult to change it. Below are some commands to get the same information and change it using shell.

Use	Command
To start remote shell console	adb shell
To get pixel density, screen resolution, refresh rate and DPI information of android device	adb shell wm density
To change screen resolution of android deviceNote: enter values based on the supported resolution of device	adb shell wm size 1080×2220
To change pixel density of android deviceNote: enter values based on the supported density of device	adb shell wm density 480

Shell commands to change directory to some important file location:

Using ADB, users can only perform 1 operation at a time on device storage but shell gives an advantage here to move inside any directory and then perform any operations on the same folder. Here are some important directories which are really useful.

Use	Command
To start remote shell console	adb shell

To change directory to /system	adb shell<enter>cd /system
To delete file from device	adb shell<enter>rm -f /sdcard/<apkname.apk>
To delete folder from device	adb shell<enter>rm -d /sdcard/test
To create folder in device	adb shell<enter>mkdir /sdcard/test
To check network statistic of android device	adb shell<enter>netstat
To get IP address information of device's WIFI	adb shell<enter>ip -f inet addr show wlan0
To get the list and monitor all process running on android device	adb shell<enter>top
To get the properties of android build.prop configuration	adb shell<enter>getprop ro.build.version.sdk
To Set/replace the properties of android build.prop configuration	adb shell<enter>setprop net.dns1 4.4.4.0

Shell commands to get the device stats:

Using shell commands users can get the information related to software and hardware for battery, display and battery stat which are useful for building an application. Below are the commands to get them.

Use	Command
To get display information of device relating to software and hardware configuration	adb shell dumpsys display
To get battery information of device relating to software and hardware configuration	adb shell dumpsys battery
To get battery stats information of device relating to software and hardware configuration	adb shell dumpsys batterystats

Shell commands to send SMS & take screenshot / to record screen:

Screen Record functionality is not available with old android devices but shell scripts are there to overcome that limitation. Also, some users don't want to even touch the devices while the device is connected for debugging. So, the user needs commands to perform all those operations from the computer. Here are a few commands to perform those operations remotely from the computer.

Use	Command
To send SMS from android device using ADB	adb shell am start -a android.intent.action.SENDTO -d sms:<phonenumber> –es sms_body "<SMS Text>" –ez exit_on_sent true adb shell input keyevent 22adb shell input keyevent 66
To take screenshot using ADB command	adb shell screencap /sdcard/screenshot.png
To start screen recording using ADB command Note: To stop the screen recording press CTRL+C or COMMAND+C	adb shell screenrecord /sdcard/movie.mp4
To start screen recording by defining output resolution using ADB command. Note:1. To stop the screen recording press	adb shell screenrecord –size 1440 x 2960 /sdcard/movie.mp4

CTRL+C or COMMAND+C2. Enter resolution values based on the supported resolution of the device.	
To start screen recording using ADB command and set recording duration. Note: Replace ‘36’ with desired value in seconds.	adb shell screenrecord –time-limit 36 /sdcard/movie.mp4
To start screen recording using ADB command and set the bitrate of the video output file.	adb shell screenrecord –bit-rate 1500000 /sdcard/movie.mp4

Shell Commands to perform key operations:

Most Android developers and testers want to automate actions on Android devices but it needs a separate setup to achieve it. While ADB provides some inbuilt commands to perform those actions. Below are the key event commands of shell to perform action remotely from computer:

Use	**Command**
To press HOME button using ADB	adb shell input keyevent 3oradb shell am start -W -c android.intent.category.HOME -a android.intent.action.MAIN
To press BACK button using ADB	adb shell input keyevent 4
To press CALL button using ADB	adb shell input keyevent 5
To press END CALL button using ADB	adb shell input keyevent 6
To to toggle device to ON/OFF	adb shell input keyevent 26
To launch camera using ADB	adb shell input keyevent 27
To launch default browser using ADB	adb shell input keyevent 64
To press enter using ADB	adb shell input keyevent 66
To press delete or backspace using ADB	adb shell input keyevent 67
To launch contacts list using ADB	adb shell input keyevent 207
To turn down brightness level using ADB	adb shell input keyevent 220
To turn up brightness level using ADB	adb shell input keyevent 221
To perform Cut (ctrl+x) operation on selected item using ADB command	adb shell input keyevent 277
To perform Copy (ctrl+c) operation on selected item using ADB command	adb shell input keyevent 278
To perform Paste operation (ctrl+v) on selected item using ADB command	adb shell input keyevent 279
To pass keycode(keyboard value) unknown/0 using ADB	adb shell input keyevent 0
To pass keycode(keyboard value) MENU/SOFT_LEFT using ADB	adb shell input keyevent 1

To pass keycode(keyboard value) SOFT_RIGHT using ADB	adb shell input keyevent 2
To pass keycode(keyboard value) HOME using ADB	adb shell input keyevent 3
To pass keycode(keyboard value) BACK using ADB	adb shell input keyevent 4
To pass keycode(keyboard value) CALL using ADB	adb shell input keyevent 5
To pass keycode(keyboard value) END CALL using ADB	adb shell input keyevent 6
To pass keycode(keyboard value) 0 using ADB	adb shell input keyevent 7
To pass keycode(keyboard value) 1 using ADB	adb shell input keyevent 8
To pass keycode(keyboard value) 2 using ADB	adb shell input keyevent 9
To pass keycode(keyboard value) 3 using ADB	adb shell input keyevent 10
To pass keycode(keyboard value) 4 using ADB	adb shell input keyevent 11
To pass keycode(keyboard value) 5 using ADB	adb shell input keyevent 12
To pass keycode(keyboard value) 6 using ADB	adb shell input keyevent 13
To pass keycode(keyboard value) 7 using ADB	adb shell input keyevent 14
To pass keycode(keyboard value) 8 using ADB	adb shell input keyevent 15
To pass keycode(keyboard value) 9 using ADB	adb shell input keyevent 16
To pass keycode(keyboard value) STAR using ADB	adb shell input keyevent 17
To pass keycode(keyboard value) POUND using ADB	adb shell input keyevent 18
To pass keycode(keyboard value) DPAD_UP using ADB	adb shell input keyevent 19
To pass keycode(keyboard value) DPAD_DOWN using ADB	adb shell input keyevent 20
To pass keycode(keyboard value) DPAD_LEFT using ADB	adb shell input keyevent 21
To pass keycode(keyboard value) DPAD_RIGHT using ADB	adb shell input keyevent 22
To pass keycode(keyboard value) DPAD_CENTER using ADB	adb shell input keyevent 23
To pass keycode(keyboard value) VOLUME_UP using ADB	adb shell input keyevent 24
To pass keycode(keyboard value)	adb shell input keyevent 25

VOLUME_DOWN using ADB	
To pass keycode(keyboard value) POWER using ADB	adb shell input keyevent 26
To pass keycode(keyboard value) CAMERA using ADB	adb shell input keyevent 27
To pass keycode(keyboard value) CLEAR using ADB	adb shell input keyevent 28
To pass keycode(keyboard value) A using ADB	adb shell input keyevent 29
To pass keycode(keyboard value) B using ADB	adb shell input keyevent 30
To pass keycode(keyboard value) C using ADB	adb shell input keyevent 31
To pass keycode(keyboard value) D using ADB	adb shell input keyevent 32
To pass keycode(keyboard value) E using ADB	adb shell input keyevent 33
To pass keycode(keyboard value) F using ADB	adb shell input keyevent 34
To pass keycode(keyboard value) G using ADB	adb shell input keyevent 35
To pass keycode(keyboard value) H using ADB	adb shell input keyevent 36
To pass keycode(keyboard value) I using ADB	adb shell input keyevent 37
To pass keycode(keyboard value) J using ADB	adb shell input keyevent 38
To pass keycode(keyboard value) K using ADB	adb shell input keyevent 39
To pass keycode(keyboard value) L using ADB	adb shell input keyevent 40
To pass keycode(keyboard value) M using ADB	adb shell input keyevent 41
To pass keycode(keyboard value) N using ADB	adb shell input keyevent 42
To pass keycode(keyboard value) O using ADB	adb shell input keyevent 43
To pass keycode(keyboard value) P using ADB	adb shell input keyevent 44
To pass keycode(keyboard value) Q using ADB	adb shell input keyevent 45
To pass keycode(keyboard value) R using ADB	adb shell input keyevent 46
To pass keycode(keyboard value) S using ADB	adb shell input keyevent 47
To pass keycode(keyboard value) T using ADB	adb shell input keyevent 48

To pass keycode(keyboard value) U using ADB	adb shell input keyevent 49
To pass keycode(keyboard value) V using ADB	adb shell input keyevent 50
To pass keycode(keyboard value) W using ADB	adb shell input keyevent 51
To pass keycode(keyboard value) X using ADB	adb shell input keyevent 52
To pass keycode(keyboard value) Y using ADB	adb shell input keyevent 53
To pass keycode(keyboard value) Z using ADB	adb shell input keyevent 54
To pass keycode(keyboard value) COMMA using ADB	adb shell input keyevent 55
To pass keycode(keyboard value) PERIOD using ADB	adb shell input keyevent 56
To pass keycode(keyboard value) ALT_LEFT using ADB	adb shell input keyevent 57
To pass keycode(keyboard value) ALT_RIGHT using ADB	adb shell input keyevent 58
To pass keycode(keyboard value) SHIFT_LEFT using ADB	adb shell input keyevent 59
To pass keycode(keyboard value) SHIFT_RIGHT using ADB	adb shell input keyevent 60
To pass keycode(keyboard value) TAB using ADB	adb shell input keyevent 61
To pass keycode(keyboard value) SPACE using ADB	adb shell input keyevent 62
To pass keycode(keyboard value) SYM using ADB	adb shell input keyevent 63
To pass keycode(keyboard value) EXPLORER using ADB	adb shell input keyevent 64
To pass keycode(keyboard value) ENVELOPE using ADB	adb shell input keyevent 65
To pass keycode(keyboard value) ENTER using ADB	adb shell input keyevent 66
To pass keycode(keyboard value) DEL using ADB	adb shell input keyevent 67
To pass keycode(keyboard value) GRAVE using ADB	adb shell input keyevent 68
To pass keycode(keyboard value) MINUS using ADB	adb shell input keyevent 69
To pass keycode(keyboard value) EQUALS using ADB	adb shell input keyevent 70
To pass keycode(keyboard value) LEFT_BRACKET using ADB	adb shell input keyevent 71
To pass keycode(keyboard value)	adb shell input keyevent 72

RIGHT_BRACKET using ADB	
To pass keycode(keyboard value) BACKSLASH using ADB	adb shell input keyevent 73
To pass keycode(keyboard value) SEMICOLON using ADB	adb shell input keyevent 74
To pass keycode(keyboard value) APOSTROPHE using ADB	adb shell input keyevent 75
To pass keycode(keyboard value) SLASH using ADB	adb shell input keyevent 76
To pass keycode(keyboard value) AT using ADB	adb shell input keyevent 77
To pass keycode(keyboard value) NUM using ADB	adb shell input keyevent 78
To pass keycode(keyboard value) HEADSETHOOK using ADB	adb shell input keyevent 79
To pass keycode(keyboard value) FOCUS using ADB	adb shell input keyevent 80
To pass keycode(keyboard value) PLUS using ADB	adb shell input keyevent 81
To pass keycode(keyboard value) MENU using ADB	adb shell input keyevent 82
To pass keycode(keyboard value) NOTIFICATION using ADB	adb shell input keyevent 83
To pass keycode(keyboard value) SEARCH using ADB	adb shell input keyevent 84
To pass keycode(keyboard value) MEDIA_PLAY/PAUSE using ADB	adb shell input keyevent 85
To pass keycode(keyboard value) MEDIA_STOP using ADB	adb shell input keyevent 86
To pass keycode(keyboard value) NEXT using ADB	adb shell input keyevent 87
To pass keycode(keyboard value) PREVIOUS using ADB	adb shell input keyevent 88
To pass keycode(keyboard value) MEDIA_REWIND using ADB	adb shell input keyevent 89
To pass keycode(keyboard value) MEDIA_FAST_FORWARD using ADB	adb shell input keyevent 90
To pass keycode(keyboard value) MUTE using ADB	adb shell input keyevent 91
To pass keycode(keyboard value) PAGE UP using ADB	adb shell input keyevent 92
To pass keycode(keyboard value) PAGE DOWN using ADB	adb shell input keyevent 93
To pass keycode(keyboard value) PICT SYMBOLS using ADB	adb shell input keyevent 94
To pass keycode(keyboard value)	adb shell input keyevent 122

MOVE_HOME using ADB	
To pass keycode(keyboard value) MOVE_END using ADB	adb shell input keyevent 123

Fastboot commands for flashing:

Fastboot enables options for users to flash custom image and recovery files using fastboot mode. Some useful operations and it's commands are listed below:

Use	Command
To list all fastboot devices	fastboot devices
To unlock bootloader of connected fastboot device	fastboot oem unlock
To re-lock bootloader of connected fastboot device	fastboot oem lock
To reboot device in fastboot or bootloader mode from fastboot mode	fastboot reboot bootloader
To flash boot image using fastboot mode	fastboot flash boot boot.img
To flash recovery image using fastboot mode	fastboot flash recovery recovery.im
To boot custom Image in device without flashing the image file	fastboot boot filename.img

Shell commands to get the device information:

Device information such as IMEI, Serial Number and also files available in device storage are important for us but at the same time it's not easy to find some of this information as it's hidden for users like system storage where firmware files are available. Some of this information are required while debugging the device. Some commands to get this information are given below:

Use	Command
To list directory content	adb shell ls
To print size of each file in selected directory	adb shell ls -s
To print list of subdirectories recursively	adb shell ls -R
To print state of device	adb get-state
To print IMEI of connected device	adb shell dumpsys iphonesybinfo
To print information related to connectivity like TCP Ip and connections	adb shell netstat
To print information of current working directory	adb shell pwd
To print list of phone features	adb shell pm list features
To print list of all services of device	adb shell service list
To print information of application activity by package and activity name	adb shell dumpsys activity <package>/<activity>
To print process status	adb shell ps
To print current display resolution information	adb shell wm size
To print activity of current opened application	dumpsys window windows | grep -E 'mCurrentFocus|mFocusedApp'
To list the information of all opened apps	adb shell dumpsys package packages
To get the information of particular application	adb shell dump <name>
To get the path of application by package name	adb shell path <package>
To change battery level of device (emulator)Note: Value can	adb shell dumpsys battery set level

be 0 to 100	<n>
To change battery status of device (emulator)Note: Value can be 0 to 3 which relates to state unknown, charging, discharging, not charging or full	adb shell dumpsys battery set status<n>
To reset battery status of device (emulator)	adb shell dumpsys battery reset
To change USB status of device (emulator)Note: Value can be 0 or 1	adb shell dumpsys battery set usb <n>