ADB Shell Commands List and Detailed Cheat Sheet

by Rakesh / March 29, 2023

ADB or Android Debug Bridge is a command-line tool developed to facilitate communication between a computer and a connected emulator or Android device. Using ADB and ADB Shell commands, we can perform various actions on a device. To be able to execute ADB and **Fastboot commands**, the Android SDK Platform-tools package must be installed on your Windows, Linux, or macOS computer. In this article, we'll explore a huge list of ADB Shell commands list with a cheat sheet.

Don't forget to check out the detailed list of **ADB commands** explaining the function of each of them.

What is ADB Shell?

ADB commands can be used to debug Android devices, install or uninstall apps, and get information about a connected device. ADB works with the aid of three components called Client, Daemon, and Server. If you are curious about how these 3 components work together to make ADB and ADB shell commands functions, see below:

- **Client**: It's is very computer on which you use a command-line terminal to issue an ADB command. which sends commands.
- **Daemon**: Or, ADBD is a background process that runs on both connected devices. It's responsible for running commands on a connected emulator or Android device.
- **Server**: It runs in the background and works as a bridge between the Client and the Daemon and manages the communication. which manages communication between the client and the daemon.

ADB Shell commands provide access to a Unix Shell that runs a command directly on your Android device. As soon as you execute an 'adb shell' command on the command terminal, it sends a signal to your Android device and triggers the remote shell command console. Thus ADB shell commands let you control your Android device.

Using ADB commands, you can reboot your device, push and pull files, create a backup and restore it, and sideload an 'update.zip' package, or an APK. ADB Shell commands, however, work on a much deeper level. They can be used to change the resolution of your device display, uninstall bloatware or system apps, enable and disable features, modify the system files, and change their configuration directly using commands from your computer.

There are more tasks you can perform using these commands, and below we'll check them all with examples. Please note that there are three prerequisites before you can make use of ADB, Fastboot, and ADB shell commands.

- **Android SDK Platform-tools**
- **USB Drivers for your Android device**
- **Enable USB Debugging**

Now you can use **Web ADB** in a web browser window to **run ADB commands on an Android device** or computer without installing ADB and Fastboot tools and USB drivers.

Finally, without any further ado, let's proceed with our list of ADB Shell commands.

Warning: Don't use the commands mentioned on this page unless you know how to use them and have some prior knowledge or experience.

ADB Shell Commands List & Cheat Sheet

In this ADB shell commands cheat sheet, I'll try to explain the function of all commands in simple language.

adb shell

This command activates the remote shell command console on the connected Android smartphone or tablet.

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 —user 0

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 –user 0

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 –user 0

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
```

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
```

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 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 settings

You can use this command to **get information about certain 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 WiFi 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 input`
- `adb shell dumpsys display`
- `adb shell dumpsys battery`
- `adb shell dumpsys batterystats`
- `adb shell dumpsys activity`
- `adb shell dumpsys cpuinfo`

- **adb shell dumpsys battery**

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 as shown below:

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

PS C:\Users\Technastic\Desktop> adb shell dumpsys cpuinfo
Load: 12.48 / 12.76 / 12.82
CPU usage from 138400ms to 89027ms ago:
8.1% 5954/system_server: 5.5% user + 2.5% kernel / faults:
9802 minor 5 major
3.1% 6485/com.android.phone: 2.2% user + 0.9% kernel /
faults: 6575 minor 1 major
2.7% 6596/com.android.systemui: 2.1% user + 0.6% kernel /
faults: 3178 minor 1 major
2.6% 26484/com.netflix.mediaclient: 1.3% user + 1.3% kernel
/ faults: 109 minor
2% 2231/sugov:0: 0% user + 2% kernel
1% 24100/kworker/u18:2: 0% user + 1% kernel
1% 5706/statsd: 0.9% user + 0.1% kernel
0.5% 3752/ueventd: 0.4% user + 0.1% kernel / faults: 25
```

```
minor
0.5% 5721/rild: 0.3% user + 0.2% kernel / faults: 20 minor
0.5% 5169/logd: 0.3% user + 0.2% kernel / faults: 43 minor
0.5% 5558/surfaceflinger: 0.3% user + 0.2% kernel / faults:
1 minor
0.4% 5170/servicemanager: 0.2% user + 0.2% kernel
0.4% 1/init: 0.3% user + 0% kernel
0.4% 19725/kworker/u17:3: 0% user + 0.4% kernel
0.3% 5546/lmkd: 0% user + 0.3% kernel
0.3% 5456/kworker/u17:1: 0% user + 0.3% kernel
0.3% 5715/argosd: 0.1% user + 0.1% kernel
0.3% 2233/sugov:4: 0% user + 0.3% kernel
0.2% 23487/kworker/u18:0: 0% user + 0.2% kernel
0.1% 23896/kworker/u16:3: 0% user + 0.1% kernel / faults: 6
minor
0.1% 8/rcu_preempt: 0% user + 0.1% kernel
0.1% 5718/lhd: 0% user + 0.1% kernel
0.1% 23489/kworker/0:2: 0% user + 0.1% kernel
```

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

You'll get a very detailed output on the command window with info like pixel resolution, **FPS, and DPI of your phone's display**.

```
Display: mDisplayId=0
    init=1440x3040 560dpi base=1080x2280 420dpi
cur=1080x2280 app=1080x2069 rng=1080x1017-2069x2069
    deferred=false mLayoutNeeded=false
mTouchExcludeRegion=SkRegion((0,0,1080,2280))

mDisplayInfo=DisplayInfo{"Built-in Screen, displayId 0",
uniqueId "local:0", app 1080 x 2069, real 1080 x 2280,
largest app 2069 x 2069, smallest app 1080 x 1017, mode 1,
defaultMode 1, modes [{id=1, width=1440, height=3040,
fps=60.000004}
```

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 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:+918052000222 --es sms_body "Test --ez exit_on_sent
false
```

adb shell screencap

By using this command, you can c**apture 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

On Android devices running Android 4.4 KitKat and above, you can even **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 and 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.

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 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 just 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, by mentioning 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
```

ADB Shell KeyEvent commands

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 simply by using these KeyEvent commands. These commands might come in handy if the hardware keys on your device are not functioning properly 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 wen 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

Download: ADB Shell Commands List PDF

Read Next: Unlock Android Lock Screen PIN and Pattern using ADB

Tags: ADB AND FASTBOOT

Related Posts

How to Use ADB Commands on Wear OS Watch over Wi-Fi

How to Install ADB and Fastboot on Linux

Fix ADB Device Unauthorized Error on Android Devices [How to]

How to Use ADB Commands to Install APK on Android Devices

Show comments