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. In order that you can execute ADB and Fastboot commands, 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, installing or uninstalling apps, and getting 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 the 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, 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.

Actually, 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**

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

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 -k –user 0

This is really a very useful ADB Shell command. Using this, you can easily uninstall the 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' followed by the Android app package name as shown below.

```
pm uninstall -k --user 0 com.whatsapp_2.20.apk
```

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 out.

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

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
```

In order 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 '**-d**' (disabled apps) and '**-e**' (for enabled apps) parameter.

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

To list app packages with specific keywords filter.

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

adb shell pm path <package-name>

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

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: In order to use this tool don't forget to add permission into your Android manifest automatically `android.permission.DUMP`

```
adb shell dumpsys
```

Other variations of the command are as follows:

- `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 (2020-03-14 17:10:34.508 to 2020-03-14
17:11:23.881):
8.2%
23098/com.google.android.webview:sandboxed_process0:org.chromium.content.app.
SandboxedProcessService0:0: 6% user + 2.2% kernel
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
0.8% 5522/android.hardware.sensors@1.0-service: 0.2% user + 0.5% kernel
0.7% 8408/com.sec.android.sdhms: 0.3% user + 0.4% kernel / faults: 897 minor
0.7% 6980/com.sec.imsservice: 0.5% user + 0.1% kernel / faults: 1177 minor
0.6% 5144/com.samsung.android.messaging: 0.5% user + 0.1% kernel / faults:
2645 minor
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% 8824/com.teslacoilsw.launcher: 0.1% user + 0.1% kernel / faults: 4 minor
0.2% 23219/com.sec.android.app.shealth:remote: 0.2% user + 0% kernel /
faults: 32 minor
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
0.1% 5518/android.hardware.graphics.composer@2.2-service: 0% user + 0% kernel
0.1% 7155/com.google.android.gms.persistent: 0.1% user + 0% kernel / faults:
28 minor
0.1% 1072/spi17: 0% user + 0.1% kernel
0.1% 2235/sugov:6: 0% user + 0.1% kernel
0.1% 5463/android.system.suspend@1.0-service: 0% user + 0% kernel
0.1% 21205/kworker/u17:2: 0% user + 0.1% kernel
0.1% 22203/kworker/u16:2: 0% user + 0.1% kernel / faults: 7 minor
0.1% 7616/adbd: 0% user + 0.1% kernel / faults: 162 minor
0.1% 7783/com.google.android.gms: 0% user + 0% kernel / faults: 27 minor
0.1% 10822/com.skype.raider: 0% user + 0% kernel / faults: 99 minor
0.1% 24174/kworker/u16:6: 0% user + 0.1% kernel
```

```
0% 5711/android.hardware.health@2.0-service.samsung: 0% user + 0% kernel
0% 6736/com.sec.location.nsflp2: 0% user + 0% kernel / faults: 183 minor
0% 7594/iod: 0% user + 0% 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'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 480
```

- **HD**

```
adb shell wm size 720x1560
adb shell wm density 350
```

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 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

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. You need to press **Ctrl+C** to stop recording manually.

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

Use the following command to set the width x height of the video:

```
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 4000000 /sdcard/screenrecord-01.mp4
```

adb shell getprop & adb shell setprop

Using the above commands, you can not only get the properties of your Android's build.prop configuration but can also set a value of property tag on the build.prop. See the examples below:

Type '**adb shell**' in the cmd window, hit the Enter key and then issue the following command:

```
getprop
```

Below are some more examples:

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

Now, to set the value of a specific build.prop property, you can use the '**adb shell setprop**' commands. See the examples below:

```
setprop net.dns1 1.2.3.4
setprop net.dns2 1.2.3.5
```

Similarly, you can also set a custom VMHeap size:

```
setprop dalvik.vm.heapsize 40m
```

adb shell cd

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

```
adb shell
(Hit Enter then the following command)
cd /system
```

adb shell rm

By using this ADB shell command, you can remove any file or directory from your Android device.

To do that, you have to type '**adb shell**' command first and hit the Enter key. After that, you can use one of the following commands followed by the file or directory name as shown below.

Delete a file:

```
rm -f /sdcard/OPWallpaperResources.apk
```

Delete a directory or folder:

```
rm -d /sdcard/ZooperWidget
```

You can also use '**rmdir**' in place of '**rm -d**' to remove a directory.

adb shell mkdir

This ADB shell command is used to create a new directory or directories under an existing directory. You can also set permission for the directory too. Execute 'adb shell' and then the following commands:

```
mkdir /sdcard/NewFolder
mkdir -m 644 /sdcard/NewFolder
mkdir -p /sdcard/NewFolder/Folder1
```

adb shell cp & adb shell mv

You can use these commands to copy, move and rename files and directories. 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/LiveWallpapers
```

To move a file from one location to another, type the following command mentioning the source and destination locations:

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

If you want to move a file to a different location with a new name,

```
mv /sdcard/OPWallpaperResources.apk /sdcard/com.whatsapp.apk
```

adb shell netstat

To check the network statistics of your Android device, execute '**adb shell**' command and type:

```
netstat
```

adb shell ip

Using this command, you can see, your phone's Wi-Fi IP address. Execute '**adb shell**' in the command window and then issue the following command:

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

adb shell top

If you want to know about the top CPU processes running on your Android device, you can use the following command after executing '**adb shell**':

```
top
```

If you want to stop CPU processes monitor, press Ctrl+C on your keyboard.

ADB Shell KeyEvent commands

By using the following ADB Shell key event commands, you can trigger certain actions performed by certain hardware buttons or UI options on Android devices.

```
adb shell input keyevent 3 // Home btn
adb shell input keyevent 4 // Back btn
adb shell input keyevent 5 // Call
```

```
adb shell input keyevent 6 // End call
adb shell input keyevent 26 // Turn Android device ON and OFF. It will toggle
device to on/off status.
adb shell input keyevent 27 // Camera
adb shell input keyevent 64 // Open browser
adb shell input keyevent 66 // Enter
adb shell input keyevent 67 // Delete (backspace)
adb shell input keyevent 207 // Contacts
adb shell input keyevent 220 / 221 // Brightness down/up
adb shell input keyevent 277 / 278 /279 // Cut/Copy/Paste
```

Finally, it's time to wrap up the ADB shell commands cheat sheet.

For more info, visit **https://technastic.com/**