Adb useful commands list

By Pulimet

Archived: 2026-04-05 12:53:01 UTC

Hi All! I've recently launched a tool that wraps many of the commands here with a user interface. This desktop application is currently available for macOS. There's a roadmap outlining planned features for the near future. Feel free to request any features you'd like to see, and I'll prioritize them accordingly. One of the most important aspects of this application is that every command executed behind the scenes is displayed in a special log section. This allows you to see exactly what’s happening and learn from it. Here's the link to the repository: https://github.com/Pulimet/ADBugger App Description: ADBugger is a desktop tool designed for debugging and QA of Android devices and emulators. It simplifies testing, debugging, and performance analysis by offering device management, automated testing, log analysis, and remote control capabilities. This ensures smooth app performance across various setups. ============================== adb help // List all comands == Adb Server adb kill-server adb start-server == Adb Reboot adb reboot adb reboot recovery adb reboot-bootloader adb root //restarts adb with root permissions == Shell adb shell // Open or run commands in a terminal on the host Android device. == Devices adb usb adb devices //show devices attached adb devices -l //devices (product/model) adb connect ip_address_of_device == Get device android version adb shell getprop ro.build.version.release == LogCat adb logcat adb logcat -c // clear // The parameter -c will clear the current logs on the device. adb logcat -d > [path_to_file] // Save the logcat output to a file on the local system. adb bugreport > [path_to_file] // Will dump the whole device information like dumpstate, dumpsys and logcat output. == Files adb push [source] [destination] // Copy files from your computer to your phone. adb pull [device file location] [local file location] // Copy files from your phone to your computer. == App install adb -e install path/to/app.apk -d - directs command to the only connected USB device... -e - directs command to the only running emulator... -s <serial number> ... -p <product name or path> ... The flag you decide to use has to come before the actual adb command: adb devices | tail -n +2 | cut -sf 1 | xargs -IX adb -s X install -r com.myAppPackage // Install the given app on all connected devices. == Uninstalling app from device adb uninstall com.myAppPackage adb uninstall <app .apk name> adb uninstall -k <app .apk name> -> "Uninstall .apk withour deleting data" adb shell pm uninstall com.example.MyApp adb shell pm clear [package] // Deletes all data associated with a package. adb devices | tail -n +2 | cut -sf 1 | xargs -IX adb -s X uninstall com.myAppPackage //Uninstall the given app from all connected devices == Update app adb install -r yourApp.apk // -r means re-install the app and keep its data on the device. adb install –k <.apk file path on computer> == Home button adb shell am start -W -c android.intent.category.HOME -a android.intent.action.MAIN == Activity Manager adb shell am start -a android.intent.action.VIEW adb shell am broadcast -a 'my_action' adb shell am start -a android.intent.action.CALL -d tel:+972527300294 // Make a call // Open send sms screen with phone number and the message: adb shell am start -a android.intent.action.SENDTO -d sms:+972527300294 --es sms_body "Test --ez exit_on_sent false // Reset permissions adb shell pm reset-permissions -p your.app.package adb shell pm grant [packageName] [Permission] // Grant a permission to an app. adb shell pm revoke [packageName] [Permission] // Revoke a permission from an app. // Emulate device adb shell wm size 2048x1536 adb shell wm density 288 // And reset to default adb shell wm size reset adb shell wm density reset == Print text adb shell input text 'Wow, it so cool feature' == Screenshot adb shell screencap -p

/sdcard/screenshot.png $ adb shell shell@ $ screencap /sdcard/screen.png shell@ $ exit $ adb pull /sdcard/screen.png --- adb shell screenrecord /sdcard/NotAbleToLogin.mp4 $ adb shell shell@ $ screenrecord --verbose /sdcard/demo.mp4 (press Control + C to stop) shell@ $ exit $ adb pull /sdcard/demo.mp4 == Key event 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 0 --> "KEYCODE_0" 1 --> "KEYCODE_SOFT_LEFT" 2 --> "KEYCODE_SOFT_RIGHT" 3 --> "KEYCODE_HOME" 4 --> "KEYCODE_BACK" 5 --> "KEYCODE_CALL" 6 --> "KEYCODE_ENDCALL" 7 --> "KEYCODE_0" 8 --> "KEYCODE_1" 9 --> "KEYCODE_2" 10 --> "KEYCODE_3" 11 --> "KEYCODE_4" 12 --> "KEYCODE_5" 13 --> "KEYCODE_6" 14 --> "KEYCODE_7" 15 --> "KEYCODE_8" 16 --> "KEYCODE_9" 17 --> "KEYCODE_STAR" 18 --> "KEYCODE_POUND" 19 --> "KEYCODE_DPAD_UP" 20 --> "KEYCODE_DPAD_DOWN" 21 --> "KEYCODE_DPAD_LEFT" 22 --> "KEYCODE_DPAD_RIGHT" 23 --> "KEYCODE_DPAD_CENTER" 24 --> "KEYCODE_VOLUME_UP" 25 --> "KEYCODE_VOLUME_DOWN" 26 --> "KEYCODE_POWER" 27 --> "KEYCODE_CAMERA" 28 --> "KEYCODE_CLEAR" 29 --> "KEYCODE_A" 30 --> "KEYCODE_B" 31 --> "KEYCODE_C" 32 --> "KEYCODE_D" 33 --> "KEYCODE_E" 34 --> "KEYCODE_F" 35 --> "KEYCODE_G" 36 --> "KEYCODE_H" 37 --> "KEYCODE_I" 38 --> "KEYCODE_J" 39 --> "KEYCODE_K" 40 --> "KEYCODE_L" 41 --> "KEYCODE_M" 42 --> "KEYCODE_N" 43 --> "KEYCODE_O" 44 --> "KEYCODE_P" 45 --> "KEYCODE_Q" 46 --> "KEYCODE_R" 47 --> "KEYCODE_S" 48 --> "KEYCODE_T" 49 --> "KEYCODE_U" 50 --> "KEYCODE_V" 51 --> "KEYCODE_W" 52 --> "KEYCODE_X" 53 --> "KEYCODE_Y" 54 --> "KEYCODE_Z" 55 --> "KEYCODE_COMMA" 56 --> "KEYCODE_PERIOD" 57 --> "KEYCODE_ALT_LEFT" 58 --> "KEYCODE_ALT_RIGHT" 59 --> "KEYCODE_SHIFT_LEFT" 60 --> "KEYCODE_SHIFT_RIGHT" 61 --> "KEYCODE_TAB" 62 --> "KEYCODE_SPACE" 63 --> "KEYCODE_SYM" 64 --> "KEYCODE_EXPLORER" 65 --> "KEYCODE_ENVELOPE" 66 --> "KEYCODE_ENTER" 67 --> "KEYCODE_DEL" 68 --> "KEYCODE_GRAVE" 69 --> "KEYCODE_MINUS" 70 --> "KEYCODE_EQUALS" 71 --> "KEYCODE_LEFT_BRACKET" 72 --> "KEYCODE_RIGHT_BRACKET" 73 --> "KEYCODE_BACKSLASH" 74 --> "KEYCODE_SEMICOLON" 75 --> "KEYCODE_APOSTROPHE" 76 --> "KEYCODE_SLASH" 77 --> "KEYCODE_AT" 78 --> "KEYCODE_NUM" 79 --> "KEYCODE_HEADSETHOOK" 80 --> "KEYCODE_FOCUS" 81 --> "KEYCODE_PLUS" 82 --> "KEYCODE_MENU" 83 --> "KEYCODE_NOTIFICATION" 84 --> "KEYCODE_SEARCH" 85 --> "KEYCODE_MEDIA_PLAY_PAUSE" 86 --> "KEYCODE_MEDIA_STOP" 87 --> "KEYCODE_MEDIA_NEXT" 88 --> "KEYCODE_MEDIA_PREVIOUS" 89 --> "KEYCODE_MEDIA_REWIND" 90 --> "KEYCODE_MEDIA_FAST_FORWARD" 91 --> "KEYCODE_MUTE" 92 --> "KEYCODE_PAGE_UP" 93 --> "KEYCODE_PAGE_DOWN" 94 --> "KEYCODE_PICTSYMBOLS" ... 122 --> "KEYCODE_MOVE_HOME" 123 --> "KEYCODE_MOVE_END" // https://developer.android.com/reference/android/view/KeyEvent.html == ShPref # replace org.example.app with your application id # Add a value to default shared preferences. adb shell 'am broadcast -a org.example.app.sp.PUT --es key key_name --es value "hello world!"' # Remove a value to default shared preferences. adb shell 'am broadcast -a org.example.app.sp.REMOVE --es key key_name' # Clear all default shared preferences. adb shell 'am broadcast -a org.example.app.sp.CLEAR --es key key_name' # It's also possible

to specify shared preferences file. adb shell 'am broadcast -a org.example.app.sp.PUT --es name Game --es key level --ei value 10' # Data types adb shell 'am broadcast -a org.example.app.sp.PUT --es key string --es value "hello world!"' adb shell 'am broadcast -a org.example.app.sp.PUT --es key boolean --ez value true' adb shell 'am broadcast -a org.example.app.sp.PUT --es key float --ef value 3.14159' adb shell 'am broadcast -a org.example.app.sp.PUT --es key int --ei value 2015' adb shell 'am broadcast -a org.example.app.sp.PUT --es key long --el value 9223372036854775807' # Restart application process after making changes adb shell 'am broadcast -a org.example.app.sp.CLEAR --ez restart true' == Monkey adb shell monkey -p com.myAppPackage -v 10000 -s 100 // monkey tool is generating 10.000 random events on the real device == Paths /data/data/<package>/databases (app databases) /data/data/<package>/shared_prefs/ (shared preferences) /data/app (apk installed by user) /system/app (pre-installed APK files) /mmt/asec (encrypted apps) (App2SD) /mmt/emmc (internal SD Card) /mmt/adcard (external/Internal SD Card) /mmt/adcard/external_sd (external SD Card) adb shell ls (list directory contents) adb shell ls -s (print size of each file) adb shell ls -R (list subdirectories recursively) == Device onformation adb get-state (print device state) adb get-serialno (get the serial number) adb shell dumpsys iphonesybinfo (get the IMEI) adb shell netstat (list TCP connectivity) adb shell pwd (print current working directory) adb shell dumpsys battery (battery status) adb shell pm list features (list phone features) adb shell service list (list all services) adb shell dumpsys activity <package>/<activity> (activity info) adb shell ps (print process status) adb shell wm size (displays the current screen resolution) dumpsys window windows | grep -E 'mCurrentFocus|mFocusedApp' (print current app's opened activity) == Package info adb shell list packages (list package names) adb shell list packages -r (list package name + path to apks) adb shell list packages -3 (list third party package names) adb shell list packages -s (list only system packages) adb shell list packages -u (list package names + uninstalled) adb shell dumpsys package packages (list info on all apps) adb shell dump <name> (list info on one package) adb shell path <package> (path to the apk file) ==Configure Settings Commands adb shell dumpsys battery set level <n> (change the level from 0 to 100) adb shell dumpsys battery set status<n> (change the level to unknown, charging, discharging, not charging or full) adb shell dumpsys battery reset (reset the battery) adb shell dumpsys battery set usb <n> (change the status of USB connection. ON or OFF) adb shell wm size WxH (sets the resolution to WxH) == Device Related Commands adb reboot-recovery (reboot device into recovery mode) adb reboot fastboot (reboot device into recovery mode) adb shell screencap -p "/path/to/screenshot.png" (capture screenshot) adb shell screenrecord "/path/to/record.mp4" (record device screen) adb backup -apk -all -f backup.ab (backup settings and apps) adb backup -apk -shared -all -f backup.ab (backup settings, apps and shared storage) adb backup -apk -nosystem -all -f backup.ab (backup only non-system apps) adb restore backup.ab (restore a previous backup) adb shell am start|startservice|broadcast <INTENT> [<COMPONENT>] -a <ACTION> e.g. android.intent.action.VIEW -c <CATEGORY> e.g. android.intent.category.LAUNCHER (start activity intent) adb shell am start -a android.intent.action.VIEW -d URL (open URL) adb shell am start -t image/* -a android.intent.action.VIEW (opens gallery) == Logs adb logcat [options] [filter] [filter] (view device log) adb bugreport (print bug reports) == Other adb backup // Create a full backup of your phone and save to the computer. adb restore // Restore a backup to your phone. adb sideload // Push and flash custom ROMs and zips from your computer. fastboot devices // Check connection and get basic information about devices connected to the computer. // This is essentially the same command as adb devices from earlier. //However, it works in the bootloader, which ADB does not. Handy for ensuring that you have properly established a connection. ---------- Shared Preferences # replace org.example.app with your application id # Add a value to default shared preferences. adb shell 'am broadcast -a org.example.app.sp.PUT --es key key_name --es value "hello world!"' # Remove a value to default

shared preferences. adb shell 'am broadcast -a org.example.app.sp.REMOVE --es key key_name' # Clear all default shared preferences. adb shell 'am broadcast -a org.example.app.sp.CLEAR --es key key_name' # It's also possible to specify shared preferences file. adb shell 'am broadcast -a org.example.app.sp.PUT --es name Game --es key level --ei value 10' # Data types adb shell 'am broadcast -a org.example.app.sp.PUT --es key string --es value "hello world!"' adb shell 'am broadcast -a org.example.app.sp.PUT --es key boolean --ez value true' adb shell 'am broadcast -a org.example.app.sp.PUT --es key float --ef value 3.14159' adb shell 'am broadcast -a org.example.app.sp.PUT --es key int --ei value 2015' adb shell 'am broadcast -a org.example.app.sp.PUT --es key long --el value 9223372036854775807' # Restart application process after making changes adb shell 'am broadcast -a org.example.app.sp.CLEAR --ez restart true' ---------- === Few bash snippets === @Source (https://jonfhancock.com/bash-your-way-to-better-android-development-1169bc3e0424) === Using tail -n //Use tail to remove the first line. Actually two lines. The first one is just a newline. The second is “List of devices attached.” $ adb devices | tail -n +2 === Using cut -sf // Cut the last word and any white space off the end of each line. $ adb devices | tail -n +2 | cut -sf -1 === Using xargs -I // Given the -I option, xargs will perform an action for each line of text that we feed into it. // We can give the line a variable name to use in commands that xargs can execute. $ adb devices | tail -n +2 | cut -sf -1 | xargs -I X echo X aw yiss === Three options below together // Will print android version of all connected devices adb devices | tail -n +2 | cut -sf -1 | xargs -I X adb -s X shell getprop ro.build.version.release === Using alias -- Example 1 alias tellMeMore=echo tellMeMore "hi there" Output => hi there -- Example 2 // Define alias alias apkinstall="adb devices | tail -n +2 | cut -sf 1 | xargs -I X adb -s X install -r $1" // And you can use it later apkinstall ~/Downloads/MyAppRelease.apk // Install an apk on all devices -- Example 3 alias rmapp="adb devices | tail -n +2 | cut -sf 1 | xargs -I X adb -s X uninstall $1" rmapp com.example.myapp // Uninstall a package from all devices -- Example 4 alias clearapp="adb devices | tail -n +2 | cut -sf 1 | xargs -I X adb -s X shell pm clear $1" clearapp com.example.myapp // Clear data on all devices (leave installed) -- Example 5 alias startintent="adb devices | tail -n +2 | cut -sf 1 | xargs -I X adb -s X shell am start $1" startintent https://twitter.com/JonFHancock // Launch a deep link on all devices Setting up your .bash_profile Finally, to make this all reusable even after rebooting your computer (aliases only last through the current session), we have to add these to your .bash_profile. You might or might not already have a .bash_profile, so let’s make sure we append to it rather than overwriting it. Just open a terminal, and run the following command touch .bash_profile && open .bash_profile This will create it if it doesn’t already exist, and open it in a text editor either way. Now just copy and paste all of the aliases into it, save, and close. alias startintent="adb devices | tail -n +2 | cut -sf 1 | xargs -I X adb -s X shell am start $1" alias apkinstall="adb devices | tail -n +2 | cut -sf 1 | xargs -I X adb -s X install -r $1" alias rmapp="adb devices | tail -n +2 | cut -sf 1 | xargs -I X adb -s X uninstall $1" alias clearapp="adb devices | tail -n +2 | cut -sf 1 | xargs -I X adb -s X shell pm clear $1" ========== Sources: - Internet - https://www.automatetheplanet.com/adb-cheat-sheet/
