adb (Android Debug Bridge) Android device control

START HERE

`adb devices -l`
List connected devices

`adb pair <ip>:<port>`
Pair for wireless debugging

`adb install <apk>`
Install an app

`adb kill-server`
Restart a stuck adb

`adb shell`
Open a device shell

`adb logcat`
Stream the device log

`adb push <local> <remote>`
Copy a file to the device

`adb pull <remote> <local>`
Copy a file from the device

adb client your command → TCP 5037 → **adb server** on the host → USB or TCP → **adbd** on the device

CONCEPTS

adb is a client talking to a server on host port 5037, which talks to adbd on each device. If a device acts up, restart the server.

With several devices attached every command needs -s <serial>, -d or -e, or set ANDROID_SERIAL.

DEVICES & SERVER

`adb devices -l`
List devices with model and product

`adb -s <serial> <command>`
Target one device

`adb -d <command>`
Target the only USB device

`adb kill-server`
Stop the adb server

`adb start-server`
Start the adb server

`adb version`
Print the adb version

`adb wait-for-device`
Block until a device is connected

WIRELESS

`adb pair <ip>:<port>`
Pair with a code (Android 11+)
Use the address and code from Developer options, Wireless debugging, Pair device. The pairing port differs from the connect port.

`adb connect <ip>:<port>`
Connect over Wi-Fi

`adb disconnect <ip>:<port>`
Drop a Wi-Fi connection

`adb tcpip 5555`
Switch a USB device to TCP/IP
Needed on Android 10 and lower: connect by USB first, then run adb connect <ip>:5555.

APPS

`adb install <apk>`
Install an APK

`adb install -r <apk>`
Reinstall, keep data

`adb install -g <apk>`
Grant all runtime permissions

`adb uninstall <package>`
Remove an app

`adb shell pm list packages`
List installed packages

APPS (cont.)

`adb shell pm list packages -3`
Only third-party apps

`adb shell pm clear <package>`
Wipe app data

`adb shell pm grant <package> <permission>`
Grant a permission

ACTIVITY MANAGER

`adb shell am start -n <package>/<activity>`
Launch an activity

`adb shell am start -a android.intent.action.VIEW -d <uri>`
Open a URL or deep link

`adb shell am force-stop <package>`
Stop an app

FILES & SHELL

`adb shell`
Interactive device shell

`adb shell <command>`
Run one command on the device

FILES & SHELL (cont.)

`adb push <local> <remote>`
Copy a file to the device

`adb pull <remote> [<local>]`
Copy a file from the device

`adb exec-out <command> > <file>`
Binary-safe output to a local file

LOGS & DEBUGGING

`adb logcat`
Stream the log

`adb logcat -c`
Clear the log buffers

`adb logcat -d`
Dump the log and exit

`adb logcat -s <tag>`
Show only one tag

`adb logcat <tag>:<level> *:S`
Filter by tag and priority
*Priorities low to high: V D I W E F S. *:S silences everything not named.*

`adb logcat -e <regex>`
Only lines matching a regex

LOGS & DEBUGGING (cont.)

`adb logcat -v time`
Show timestamps

`adb bugreport [<path>]`
Collect a full bug report zip

`adb shell dumpsys <service>`
Dump a system service state

SCREEN & INPUT

`adb exec-out screencap -p > <file>.png`
Take a screenshot

`adb shell screenrecord /sdcard/<file>.mp4`
Record the screen (Ctrl+C stops)
Maximum 180 seconds, no audio. Pull the file afterwards.

`adb shell input text <text>`
Type text

`adb shell input keyevent <code>`
Send a key press (3 home, 4 back)

PORTS & POWER

`adb forward tcp:<host> tcp:<device>`
Host port to device port

`adb reverse tcp:<device> tcp:<host>`
Device port to host port

`adb reboot`
Reboot the device

`adb reboot bootloader`
Reboot to the bootloader

RECIPES

Debug an app over Wi-Fi

```
# phone: Wireless debugging on
adb pair 192.168.1.20:37099
adb connect 192.168.1.20:41235
adb devices -l
```

Clean reinstall and watch it start

```
adb uninstall com.example.app
adb install -g app-debug.apk
adb logcat -c
adb shell am start -n
com.example.app/.MainActivity
adb logcat --pid=$(adb shell
pidof com.example.app)
```

Reach a dev server from the phone

```
adb reverse tcp:8080 tcp:8080
# on the device, http://
localhost:8080 now hits your
host
```