FREE eBook

LEARNING adb

Free unaffiliated eBook created from
Stack Overflow contributors.

#adb

Table of Contents

About 1

Chapter 1: Getting started with adb 2

Remarks 2

Examples 2

Installation or Setup 2

Introduction 3

Chapter 2: Collecting adb commands log 4

Remarks 4

Examples 4

in Windows 4

Chapter 3: Connecting to device 5

Examples 5

Finding devices connected to your PC 5

Chapter 4: Showing Logs on ADB 6

Examples 6

Displaying and filtering with Logcat 6

Chapter 5: Starting an app in debug mode 8

Examples 8

How to wait for debugger before starting the app? 8

Chapter 6: Transferring files using push and pull 9

Syntax 9

Parameters 9

Remarks 9

Examples 9

Push a file to the SD card 9

Pull a file from the SD card to the current working directory 9

Credits 10

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: adb

It is an unofficial and free adb ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official adb.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with adb

Remarks

This section provides an overview of what adb is, and why a developer might want to use it.

It should also mention any large subjects within adb, and link out to the related topics. Since the Documentation for adb is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Specific to Windows System and android Phone:

Requirements:

1. USB Cable
2. Android Device
3. Android Driver Software

Basically after connecting USB cable PC detects the Android Device and it will automatically search for the required Drivers for that Android Device. If that drivers are not found then you have to install manually.

Manual Installation:

1. First install Android SDK in your PC(Windows)
2. After installing in Android SDK tools Right click on the SDK Manager and select "Run as Administrator"
3. In the SDK Manager select "Extras->Google USB Driver". Enable the checkbox and click "Install 1 Package"
4. When the Google USB driver is installed, plug in your device. Warning: The driver won't install automatically. We will do it manually in the next steps.
5. Open the System Properties dialog (press Win+Break on the keyboard or locate "Computer" in Start Menu, right-click on it and select "Properties".
6. Click on the "Device Manager" link.
7. In the Device Manager locate your Android device. Then right-click on it and select "Update Driver Software".
8. Select "Browse my computer for driver software".

9. Select "Let me pick from a list of device drivers on my computer".
10. Select "Show All Devices".
11. Press the "Have Disk" button.
12. Enter the path to the Google USB driver. Normally it is located in the following directory:C:\Program Files (x86)\Android\android-sdk\extras\google\usb_driver
13. Select "Android ADB Interface" from the list of device types.
14. Confirm the installation of the driver by pressing "Yes".
15. Confirm the installation again by pressing "Install".
16. When the installation is done, press "Close".
17.

Introduction

`adb` is a command line tool for communicating with an emulator instance or connected device. It allows for installing and debugging apps, transferring files, as well as a variety of other interactions with the connected emulator or device. The ADB system consists of a *client*, which sends commands from the host computer, a *daemon*, which runs on the connected device and executes commands received from the client, and a *server*, which runs on the host computer and manages communications between the client and daemon.

Official Documentation

https://developer.android.com/studio/command-line/adb.html

Read Getting started with adb online: https://riptutorial.com/adb/topic/2633/getting-started-with-adb

Chapter 2: Collecting adb commands log

Remarks

Make sure that your automation does not use `adb kill-server` command.

Examples

in Windows

Open a *Command Prompt* window and run the following commands:

```
adb kill-server
set ADB_TRACE=sockets
adb nodaemon server 2>&1 | for /f "usebackq tokens=7*" %a in (`findstr /c:"): '"`) do @echo %a
%b >> %USERPROFILE%\Desktop\adb_host_log.txt
```

Now you can run your Android automation. When done run `adb kill-server` in another *Command Prompt* window. Now the *adb_host_log.txt* file on your *Desktop* contains the log of all commands all *adb clients* have sent to the *adb host*.

Read Collecting adb commands log online: https://riptutorial.com/adb/topic/5631/collecting-adb-commands-log

Chapter 3: Connecting to device

Examples

Finding devices connected to your PC

Enable USB Debugging on your device and from command line type `adb devices`. If everything is OK, the response should be:

```
List of devices attached
1234567890 device
```

Where `1234567890` is the device's id.
If multiple devices are connected, you should see all of them:

```
List of devices attached
1234567890 device
2222222222 device
...
```

When connecting a device for the first time, you'll get a pop-up window on your device, asking you to approve the connection.

Read Connecting to device online: https://riptutorial.com/adb/topic/3174/connecting-to-device

Chapter 4: Showing Logs on ADB

Examples

Displaying and filtering with Logcat

Displaying all the logs from the default buffer on the Command Line can be accomplished by:

```
adb logcat
```

This command will show you all the logs from the device's main buffer. Notice that if you use it for the first time, you'll get a lot of information, an enormous stream of data. So you may want to clear the logs first...

Cleaning the logs:

```
adb logcat -c
```

This will clean clear the logs, and start fresh.

Displaying Alternate Buffers

There are two other buffers besides the main buffer that may be displayed as follows:

`adb logcat -b` *buffer_name*,

where *buffer_name* is one of the following:

- `radio` - view the buffer that contains radio/telephony related messages.
- `events` - view the buffer containing events-related messages.
- `main` - view the main log buffer (default)

Filtering Log Output

Logcat logs got so called levels:

> **V** — Verbose, **D** — Debug, **I** — Info, **W** — Warning, **E** — Error, **F** — Fatal, **S** — Silent

Those levels are specified when application uses those Log function:

```
Log.v(); // Verbose
Log.d(); // Debug
Log.i(); // Info
Log.w(); // Warning
Log.e(); // Error
```

if your code Log call is:

```
Log.i("MainActivtyTag", "Showing the very first fragment");
```

in logcat you'll see this output:

```
07-27 11:34:21.027 I MainActivtyTag 66 : Showing the very first fragment
```

So, this is the log convention:

```
<timestamp> <logLevel> <tag> <line> : <messge>
```

For instance, if you want to show all the logs that have Fatal (F) level:

```
adb logcat *:F
```

* *is a what called a wild card - stands for all package names*

Filtering by application package name

Since package names are guaranteed to be unique , you can filter logcat by your package name, of course you can combine it with the Level filter:

```
adb logcat <package name>:<log level>
```

For exiting/interrupting process - press `Ctrl + X`

Read Showing Logs on ADB online: https://riptutorial.com/adb/topic/4252/showing-logs-on-adb

Chapter 5: Starting an app in debug mode

Examples

How to wait for debugger before starting the app?

Let's say your launch activity is called `MainActivity`, in your app com.example.myapp. In the manifest:

```
<activity
        android:name=".MainActivity"
        >
        <intent-filter>
          <action android:name="android.intent.action.MAIN"/>
          <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
</activity>
```

Now let's say you want to launch the app, so that it waits for the debugger to connect before the app really starts.

You can use `adb shell` to achieve that.
In our case, simply run:

```
adb shell am start -D -n com.example.myapp/com.example.myapp.MainActivity
```

Now, all that's left is to attach your favorite debugger. For example, if you use Intellij or Android Studio go to Run->Attach debugger to Android process-> select your app package name

Read Starting an app in debug mode online: https://riptutorial.com/adb/topic/4009/starting-an-app-in-debug-mode

Chapter 6: Transferring files using push and pull

Syntax

- adb push [-p] LOCAL REMOTE
- adb pull [-a] [-p] REMOTE [LOCAL]

Parameters

Parameters	Details
LOCAL	A file or directory that is located on the user's computer
REMOTE	A file or directory that is located on the user's Android device
-a	Also copy the file the remote file's timestamp and file mode data
-p	Display transfer progress while the file or directory is copying

Remarks

If LOCAL is omitted in the adb pull command, the filename from REMOTE is used

LOCAL can be a relative path or an absolute path, but REMOTE must be an absolute path

Examples

Push a file to the SD card

```
adb push file.txt /sdcard/
```

Pull a file from the SD card to the current working directory

```
adb pull /sdcard/file.txt
```

Read Transferring files using push and pull online:
https://riptutorial.com/adb/topic/5844/transferring-files-using-push-and-pull

Credits

S. No	Chapters	Contributors
1	Getting started with adb	Community, Paul Ratazzi, Vesrak r
2	Collecting adb commands log	Alex P.
3	Connecting to device	TDG
4	Showing Logs on ADB	Paul Ratazzi, Pavel Durov
5	Starting an app in debug mode	Ginandi
6	Transferring files using push and pull	Carlo B.