Category:Android debugging tools

Contents

Stable: 17.06.2020 - 15:26 / Revision: 17.01.2020 - 08:26

Stable: 17.06.2020 - 10:20 | Revision: 17.01.2020 - 08:20

A quality version of this page, approved on *17 June 2020*, was based off this revision.

This category groups together all articles related to the **Android™ debugging** tools that allow to get dynamic logs about the Android framework.

Pages in category "Android debugging tools"

This category contains only the following page.

- ADB

Stable: 15.02.2021 - 12:33 / Revision: 12.02.2021 - 08:25

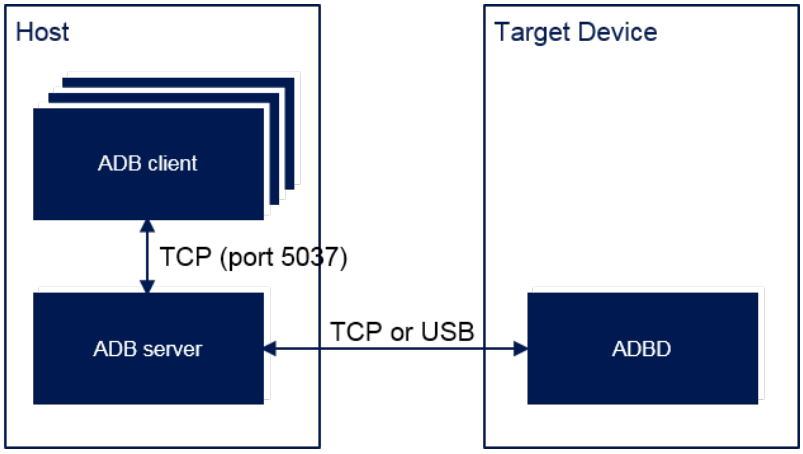
A quality version of this page, approved on *15 February 2021*, was based off this revision.

The Android Debug Bridge (ADB) is a versatile command-line tool that lets you communicate with a device (an emulator or a connected Android device).

This article is intended for Developer Package or Distribution Package users (see Which Package better suits your needs for more information).

Contents

1 Overview



- The **ADB client** is an executable that can be launched with a subcommand such as `adb shell` or `adb logcat`.
- The **ADB server** acts as a proxy between adb clients and adbd.
- The **ADBD daemon** is started by executing an init command on the target device.

The target device can be connected using different solutions:

- How to connect over USB (default)
- How to connect over Ethernet
- How to connect over Wi-Fi

As soon as it is connected, you can start testing your application with `Android Studio`.

It is also possible to execute all ADB commands (see ADB commands for more details).

For that purpose, search for the ADB tool in your environment: Host computer installation.

Information

If you are using a Distribution Package, please refer to ADB for Distribution Package

2 Host computer installation

ADB is part of the Android SDK (loaded using `Android Studio`) for Linux®, Windows® and OS X®.

At this stage, ADB commands can be executed directly from the `Android Studio` terminal:

```
cd <SDK path>
cd platform-tools
```

The *<SDK path>* can be determined by opening the SDK Manager from `Android Studio`.

3 Commands

The ADB commands enable the execution of a variety of actions on the target device, such as installing and debugging apps. They also provide access to a Unix shell that can be used to run a variety of commands.

See ADB commands summary for more information.

Several useful commands:

- `adb devices`: check connected devices
- `adb logcat`: trace
- `adb root`: give root privileges
- `adb shell`: open a console on the device
- `adb remount`: remount read-only partition in read-write to allow updating their containing (use overlayfs mechanism)
- `adb push <file>`: push a file on the device
- `adb pull <file>`: pull a file from the device
- `adb install <package>`: install an application package (APK) on the device

4 Device connection

4.1 How to connect over USB

Follow the steps below to connect your device:

- Configure your host computer
- Configure your target device
- Check the device connection

4.1.1 Host computer configuration

First check if you can already see the device connected: Template:PC adb devices

The installation and configuration depend on the host computer OS used:

- Linux Host (Ubuntu) case
- Windows Host case
- OS X Host case

4.1.1.1 Linux® Host (Ubuntu)

In Linux environments, the Android USB drivers are built-in. The only action required is to set the target device information.

Information

To peform this action, you need administrator rights. In addition, you may need to add *sudo* in front of each executed command

To do this, open a terminal and

- Create (or update if it already exists) the *51-android.rules* file in */etc/udev/rules.d/* with the following information
  - *idVendor* = 0483 (STMicroelectronics vendor)
  - *idProduct* = 0adb (ADB on STMicroelectronics device)
  - *Mode* = 0666 (read/write permissions)
  - *Group* = plugdev (Unix group which owns the device node)

Information

Check if you belong to the *plugdev* group by executing the following command:

```
$ groups
```

Example:

```
# ADB on STMicroelectronics devices
SUBSYSTEM=="usb", ATTR{idVendor}=="0483", ATTR{idProduct}=="0adb", MODE="0660", GROUP="
plugdev"
```

- Make sure that the access rights to the created file are the correct ones:

```
chmod a+r /etc/udev/rules.d/51-android.rules
```

At this stage, your USB driver is correctly installed and configured.

4.1.1.2 *Windows Host*

In Windows environments, it is required to install the USB driver and set the target device information.

Get back a compatible driver here after (Windows 10): st-android-winusb.zip

Install it:

- Connect the device to your computer's USB port.
- Open Windows **Settings**, then select **Devices**, then '"Devices and Printers"'
- Right-click the name of the device you connected, and then select **Properties**
- In the **Hardware** tab, select again **Properties**
- In the **Driver** tab, you can select **Update Driver** and then select **Browse my computer for driver software** and click **Next**
- Locate the USB driver folder you just loaded
- Click **Next** to install the driver.

At this stage, the USB driver is correctly installed and configured.

4.1.1.3 *OS X Host*

In OS X environments, the Android USB drivers are built-in and no manual configuration is required to install and configure the USB driver.

4.1.2 Target device configuration

Prerequisite: Use ST Android distribution to start the target device and access the Android user interface.

On the target device:

- Disconnect the USB cable between the target device and the host computer (if it is connected).
- Find the *Settings > Developer options* configuration screen on your target device (if it is not visible, select *Settings > About device* and click the *Build number* menu entry seven times).
- Enable the USB Debugging option from the *Settings > Developer options* menu.
- Reconnect the USB cable between the target device and the host computer.
- If an alert is displayed on your target device requesting permission to "Allow USB debugging," click *OK. It is recommended to check the "Always allow from this computer" option to avoid this from happening each time the target device is connected to the same host computer.*

4.1.3 Checking the device connection

First, ensure that the device is connected on the host computer through an USB cable, and that it has been started.

With `Android Studio`, it's easy to check if the target device is available. For example, click on `Logcat` at the bottom and check the devices seen in the list, `No connected devices` is seen otherwise.

Information

The name of the STM32MPU devices start with *STMicroelectronics*

4.2 How to connect over Ethernet

To use ADB over an Ethernet connection, connect the host computer and the target device on the same network.

First, follow the instructions provided in How to connect over USB.

Then:

- Connect the USB cable between the target device and the host computer.
- Open a terminal on your host computer.
  - Go to the *<SDK path>/platform-tools/* directory (see Host computer installation to know how to get the <SDK path>).
  - Execute *adb shell ifconfig* to get the target IP address <device_ip_address>.
  - Execute *adb tcpip 5555* to set the TCP/IP connection over port 5555.
- Disconnect the USB cable between the target device and the host computer.
- Execute *adb connect <device_ip_address>* to reconnect on the device through Ethernet

4.3 How to connect over Wi-Fi

Follow the instructions provided in the *Android Studio* website: Connect to a device over Wi-Fi

5 ADB for Distribution Package

5.1 Installing the Host computer

By default, *ADB* is built when you compile the ST Android distribution. It is automatically added to your environment's $PATH.

Another possibility is to install ADB but executing *apt-get install android-tools-adb*, although this may be a hindrance to the development in case of ST Android distribution and OS ADB version mismatch (the server is restarted every time it is used with a mismatched version of an ADB client).

The device is not necessarily visible in the terminal when executing `adb devices`. If this is not the case:

- Create a file within ~/.android/ directory named adb_usb.ini.
- Add one line containing the STMicroelectronics idVendor:

```
0x0483
```

5.2 Implementing ADB

ADB source code is stored in *system/core/adb* (common files between host and target device are differentiated by the ADB _HOST tag).

Three properties can be configured for ADB:

- `persist.adb.trace_mask` used to enable or disable ADBD traces on target device.

  Possible values (*listed in* `system/core/adb/adb_trace.cpp`): 0, 1 (or all), adb, sockets, packets, rwx, usb, sync, sysdeps, transport, jdwp, services, auth, fdevent or shell

```
Board $> stop adbd
Board $> setptop persist.adb.trace_mask = <value>
Board $> start adbd
```

Information

Traces are available in `/data/adb/adb-<date_at_start>`

- `ro.adb.secure` used to enable or disable the authentication mechanism (1: authentication enabled, 0: authentication disabled). This property can not be modified dynamically. It is considered only in case of `eng` or `userdebug` image builds. It is ignored otherwise (the authentication mechanism is enabled by default for user build).
- `service.adb.tcp.port` used to enable the TCP/IP port if a wireless connection is used (this property is not set by default as it is not necessary for USB).

ADB is an Android specific gadget device available in user space. functionfs must be mounted since it is used to create this gadget device and register it to the USB Device Controller (UDC). Three endpoints are used (one for control, one for data input, one for data output).

It is also possible to create a composite device between ADB and other protocols, using the configfs mechanism.

ADB over USB is the default configuration within the Android baseline. By default, it is configured as follows:

- `persist.sys.usb.config`: [adb] USB ADB gadget device
- `sys.usb.config`: [adb] USB ADB gadget device
- `service.adb.root`: [0] no root privileges

The ADB over wireless connection can be set within your distribution (Ethernet or Wi-Fi) by adding the following property in */device/stm/<STM32Series>/<BoardId>/device.mk*:

```
PRODUCT_PROPERTY_OVERRIDES += service.adb.tcp.port=5555
```

Android debug bridge (Android specific)

Android debug bridge daemon (Android specific)

Software development kit (A programming package that enables a programmer to develop applications for a specific platform.)

Linux® is a registered trademark of Linus Torvalds.

Operating System

technology for wireless local area networking with devices based on the IEEE 802.11 standards

Read Only

USB Device Controller

Configuration File System (See https://en.wikipedia.org/wiki/Configfs for more details)

stm32mp1

eval,disco (Generic term used, to complete configuration modules paths depending on used board)