



AVD

Run Apps on a Hardware Device



Enabling On-device Developer Options

- When building an Android app, it's important that you always test your application on a real device before releasing it to users.
- Android-powered devices have a host of developer options that you can access on the phone, which let you:
 - Enable debugging over USB.
 - Quickly capture bug reports onto the device.
 - Show CPU usage on screen.
 - Draw debugging information on screen such as layout bounds, updates on GPU views and hardware layers, and other information.
 - Plus many more options to simulate app stresses or enable debugging options.



Cont ...

- To access these settings, open the Developer options in the system Settings.
- On Android 4.2 and higher, the Developer options screen is hidden by default.
- To make it visible, go to Settings > About phone and tap Build number seven times.
- Return to the previous screen to find Developer options at the bottom.



Setting up a Device for Development

- With an Android-powered device, you can develop and debug your Android applications just as you would on the emulator.
- Before you can start, there are just a few things to do:
 - 1) Verify that your application is "debuggable" in your manifest or build.gradle file.

In the build file, make sure the debuggable property in the debug build type is set to true.

 - The build type property overrides the manifest setting.



Cont ...

```
android {  
    buildTypes {  
        debug {  
            debuggable true  
        }  
    }  
}
```

- In the AndroidManifest.xml file, add **android:debuggable="true"** to the **<application>** element.
- **NB:** If you manually enable debugging in the manifest file, be sure to disable it in your release build (your published application should usually not be debuggable).



Cont ...

2) Enable USB debugging in the device system settings, under Settings > Developer options.

3) Set up your system to detect your device.

- If you're developing on Windows, you need to install a USB driver for adb. <https://developer.android.com/tools/extras/oem-usb.html>
- If you're developing on Mac OS X, it just works. Skip this step.
- If you're developing on Ubuntu Linux, you need to add a udev rules file that contains a USB configuration for each type of device you want to use for development. In the rules file, each device manufacturer is identified by a unique vendor ID, as specified by the ATTR{idVendor} property.



Cont ...

- To set up device detection on Ubuntu Linux:
 - Log in as root and create this file: **/etc/udev/rules.d/51-android.rules**.
 - Use this format to add each vendor to the file:
 - **SUBSYSTEM=="usb", ATTR{idVendor}=="0bb4", MODE="0666", GROUP="plugdev"**
- Now execute:
 - **chmod a+r /etc/udev/rules.d/51-android.rules**

- When plugged in over USB, you can verify that your device is connected by executing `adb devices` from your SDK platform-tools/ directory. If connected, you'll see the device name listed as a "device."



Cont ...

- If using Android Studio, run or debug your application as usual.
- You will be presented with a Device Chooser dialog that lists the available emulator(s) and connected device(s).
- Select the device upon which you want to install and run the application.
- If using the Android Debug Bridge (adb), you can issue commands with the **-d** flag to target your connected device.



USB Vendor IDs

- Company USB Vendor ID
- Acer 0502
- ASUS 0b05
- Dell 413c
- Foxconn 0489
- Fujitsu 04c5
- Fujitsu Toshiba 04c5
- ZTE 19d2
-



Cont ...

- Garmin-Asus 091e
- Google 18d1
- Haier 201E
- Hisense 109b
- HP 03f0
- HTC 0bb4
- Huawei 12d1

Cont ...

- 
- Intel 8087
 - K-Touch 24e3
 - KT Tech 2116
 - Kyocera 0482
 - Lenovo 17ef
 - LG 1004
 - Motorola 22b8
 - MTK 0e8d



Cont ...

- NEC 0409
- Nook 2080
- Nvidia 0955
- OTGV 2257
- Pantech 10a9
- Pegatron 1d4d
- Philips 0471



Cont ...

- PMC-Sierra 04da
- Qualcomm 05c6
- SK Telesys 1f53
- Samsung 04e8
- Sharp 04dd
- Sony 054c



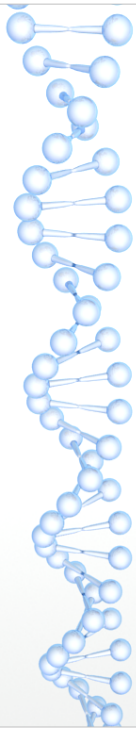
Cont ...

- Sony Ericsson 0fce
- Sony Mobile Communications 0fce
- Teleepoch 2340
- Toshiba 0930



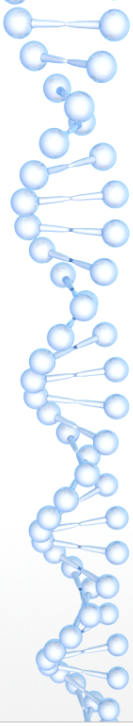
AVD

Run Apps on a Hardware Device



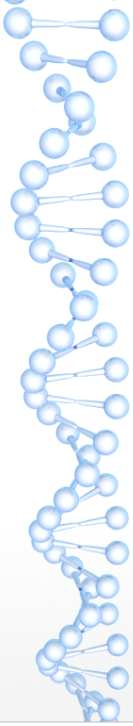
Enabling On-device Developer Options

- When building an Android app, it's important that you always test your application on a real device before releasing it to users.
- Android-powered devices have a host of developer options that you can access on the phone, which let you:
 - Enable debugging over USB.
 - Quickly capture bug reports onto the device.
 - Show CPU usage on screen.
 - Draw debugging information on screen such as layout bounds, updates on GPU views and hardware layers, and other information.
 - Plus many more options to simulate app stresses or enable debugging options.



Cont ...

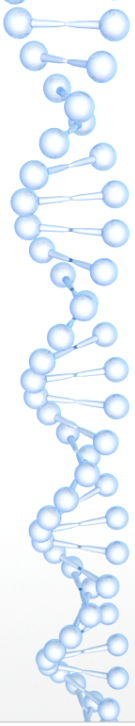
- To access these settings, open the Developer options in the system Settings.
- On Android 4.2 and higher, the Developer options screen is hidden by default.
- To make it visible, go to Settings > About phone and tap Build number seven times.
- Return to the previous screen to find Developer options at the bottom.



Setting up a Device for Development

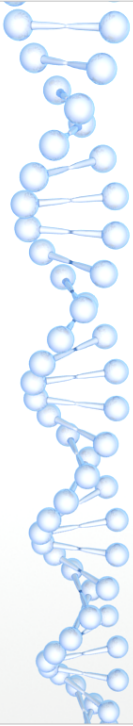
- With an Android-powered device, you can develop and debug your Android applications just as you would on the emulator.
- Before you can start, there are just a few things to do:
 - 1) Verify that your application is "debuggable" in your manifest or build.gradle file.
 - In the build file, make sure the debuggable property in the debug build type is set to true.
 - The build type property overrides the manifest setting.

Cont ...



```
android {  
  buildTypes {  
    debug {  
      debuggable true  
    }  
  }  
}
```

- In the AndroidManifest.xml file, add **android:debuggable="true"** to the **<application>** element.
- **NB:** If you manually enable debugging in the manifest file, be sure to disable it in your release build (your published application should usually not be debuggable).



Cont ...

2) Enable USB debugging in the device system settings, under Settings > Developer options.

3) Set up your system to detect your device.

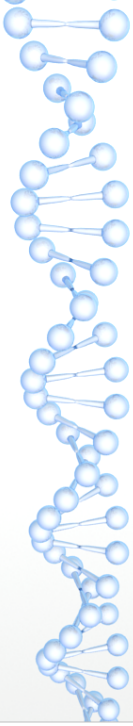
- If you're developing on Windows, you need to install a USB driver for adb. <https://developer.android.com/tools/extras/oem-usb.html>
- If you're developing on Mac OS X, it just works. Skip this step.
- If you're developing on Ubuntu Linux, you need to add a udev rules file that contains a USB configuration for each type of device you want to use for development. In the rules file, each device manufacturer is identified by a unique vendor ID, as specified by the ATTR{idVendor} property.



Cont ...

- To set up device detection on Ubuntu Linux:
 - Log in as root and create this file: **/etc/udev/rules.d/51-android.rules**.
 - Use this format to add each vendor to the file:
 - **SUBSYSTEM=="usb", ATTR{idVendor}=="0bb4", MODE="0666", GROUP="plugdev"**
- Now execute:
 - **chmod a+r /etc/udev/rules.d/51-android.rules**

When plugged in over USB, you can verify that your device is connected by executing `adb devices` from your SDK platform-tools/ directory. If connected, you'll see the device name listed as a "device."



Cont ...

- If using Android Studio, run or debug your application as usual.
- You will be presented with a Device Chooser dialog that lists the available emulator(s) and connected device(s).
- Select the device upon which you want to install and run the application.
- If using the Android Debug Bridge (adb), you can issue commands with the **-d** flag to target your connected device.



USB Vendor IDs

·	Company	USB Vendor ID
·	Acer	0502
·	ASUS	0b05
·	Dell	413c
·	Foxconn	0489
·	Fujitsu	04c5
·	Fujitsu Toshiba	04c5
·	ZTE	19d2
·		

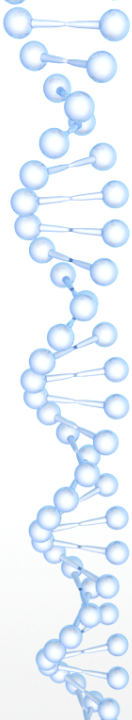
Cont ...

- Garmin-Asus 091e
- Google 18d1
- Haier 201E
- Hisense 109b
- HP 03f0
- HTC 0bb4
- Huawei 12d1

Cont ...

- 
- Intel 8087
 - K-Touch 24e3
 - KT Tech 2116
 - Kyocera 0482
 - Lenovo 17ef
 - LG 1004
 - Motorola 22b8
 - MTK 0e8d

Cont ...

- 
- NEC 0409
 - Nook 2080
 - Nvidia 0955
 - OTGV 2257
 - Pantech 10a9
 - Pegatron 1d4d
 - Philips 0471

Cont ...

- 
- PMC-Sierra 04da
 - Qualcomm 05c6
 - SK Telesys 1f53
 - Samsung 04e8
 - Sharp 04dd
 - Sony 054c



Cont ...

- Sony Ericsson Ofce
- Sony Mobile Communications Ofce
- Teleepoch 2340
- Toshiba 0930