**QPST:**

 QPST stands for **QUALCOMM Product Support Tools**. This tool is compatible with multiple Qualcomm chipset devices ranging from smart phones to tablets. The main purpose of this tool is to allow flashing of firmware files onto the required device.

It is used for build loading,taking QCN backup and installing QCN into device.

**QCN(Qualcomm Calibration Network):**

The **qcn file extension** is used for some sort of drive backup file created using the Qualcomm QPST utility.

**QFIL:**

Qfil stands for **Qualcomm flash image loader**.Qfil erases all partitions and it is used for flashing.

**.MBN file:**

Multi boot binary [firmware](https://techterms.com/definition/firmware" \t "https://fileinfo.com/extension/techterms) image used by a variety of Android devices, mainly Samsung mobile phones; contains binary data for a device's memory partitions, such as the resources and power manager, secondary bootloader,AP bootloader, and trust zone; used to update firmware on an Android device.

**RAMDUMP:**

A memory dump is a process in which the contents of memory are displayed and stored in case of an application or system crash.Sahara folder contains all RAMDUMP.

**QXDM:**

The **QUALCOMM extensible Diagnostic Monitor** (QXDM Professional) is a real-time data collection and diagnostic logging tool for measuring mobile-based RF performance.Designed to operate using all commercial handsets that contain QUALCOMM ASIC’s (application specific integrated circuits) and with QUALCOMM’s test/trial phones,\*QXDM Professional displays statistics and diagnostic information, and enables users to read and write non-volatile memory. Whether conducting tests in the lab or the field, QXDM Professional is a powerful platform for evaluating handset and network performance.

**Android Architecture:**

**Linux Kernel:**

The foundation of the Android platform is the Linux kernel. For example, [the Android Runtime (ART)](https://developer.android.com/guide/platform/index.html" \l "art) relies on the Linux kernel for underlying functionalities such as threading and low-level memory management.

Using a Linux kernel allows Android to take advantage of [key security features](https://source.android.com/security/overview/kernel-security.html) and allows device manufacturers to develop hardware drivers for a well-known kernel.

**Hardware Abstraction Layer(HAL):**

The [hardware abstraction layer (HAL)](https://source.android.com/devices/index.html" \l "Hardware Abstraction Layer) provides standard interfaces that expose device hardware capabilities to the higher-level [Java API framework](https://developer.android.com/guide/platform/index.html" \l "api-framework). The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the [camera](https://source.android.com/devices/camera/index.html) or [bluetooth](https://source.android.com/devices/bluetooth.html) module. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.

**Android Runtime:**

For devices running Android version 5.0 (API level 21) or higher, each app runs in its own process and with its own instance of the [Android Runtime (ART)](http://source.android.com/devices/tech/dalvik/index.html). ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a bytecode format designed specially for Android that's optimized for minimal memory footprint.

Prior to Android version 5.0 (API level 21), Dalvik was the Android runtime. If your app runs well on ART, then it should work on Dalvik as well, but [the reverse may not be true](https://developer.android.com/guide/practices/verifying-apps-art.html).

**Native C/C++ Libraries:**

Many core Android system components and services, such as ART and HAL, are built from native code that require native libraries written in C and C++. The Android platform provides Java framework APIs to expose the functionality of some of these native libraries to apps. For example, you can access[OpenGL ES](https://developer.android.com/guide/topics/graphics/opengl.html) through the Android framework’s [Java OpenGL API](https://developer.android.com/reference/android/opengl/package-summary.html) to add support for drawing and manipulating 2D and 3D graphics in your app.

**Java API Framework:**

The entire feature-set of the Android OS is available to you through APIs written in the Java language. These APIs form the building blocks you need to create Android apps by simplifying the reuse of core, modular system components and services,which includes resource manager,activity manager,notification manager etc.

**System Apps:**

Android comes with a set of core apps for email, SMS messaging, calendars, internet browsing, contacts, and more. Apps included with the platform have no special status among the apps the user chooses to install. So a third-party app can become the user's default web browser, SMS messenger, or even the default keyboard.

**Three versions in Android OS:**

**Compiled Version:**The compiled Version is the version of the API the app is compiled against. This means you can use Android API features included in that version of the API (as well as all previous versions).

**Target Version:**The target Version has nothing to do with how your app is compiled or what APIs you can utilize. The target Version is supposed to indicate that you have tested your app on the version you specify. This is more like a certification or sign off you are giving the Android OS as a hint to how it should handle your app in terms of OS features.

**Minimum Version:**This means you can use Android API features included in that version of the API but not the previous versions.

**Type of Application Development:**

**Native Apps:**It live on the device and are accessed through [icons](https://www.nngroup.com/articles/classifying-icons/) on the device home screen. Native apps are installed through an application store (such as Google Play or Apple’s App Store). They are developed specifically for one platform, and can take full advantage of all the device features.

**Hybrid Apps:**They are part native apps, part web apps. Like native apps, they live in an app store and can take advantage of the many device features available. Like web apps, they rely on HTML being rendered in a browser, with the caveat that the browser is embedded within the app.

**Web Apps:**They are not real applications; they are really ****websites**** that, in many ways,*look and feel* like native applications, but are not *implemented*as such. They are run by a browser and typically written in HTML5.

**Activity Life cycle:**

|  |  |
| --- | --- |
| **Method** | **Description** |
| **onCreate** | called when activity is first created. |
| **onStart** | called when activity is becoming visible to the user. |
| **onResume** | called when activity will start interacting with the user. |
| **onPause** | called when activity is not visible to the user. |
| **onStop** | called when activity is no longer visible to the user. |
| **onRestart** | called after your activity is stopped, prior to start. |
| **onDestroy** | called before the activity is destroyed. |

**Android Testing Approach:**

**Junit:** JUnit is a unit testing framework for Java programming language. JUnit promotes the idea of "first testing then coding", which emphasizes on setting up the test data for a piece of code that can be tested first and then implemented. This approach is like "test a little, code a little”.

**UI Testing:** User interface testing, a testing technique used to identify the presence of defects is a product/software under test by using Graphical user interface [GUI].

**Espresso:** Espresso is a testing framework contained in the Android Testing Support Library. It provides APIs to simulate user interactions and write functional UI tests.Espresso tests are written based on what user might do while interacting with your app.

**UI Automator:** UI Automator is a Java library used to create customized-UI test cases for an android application and it provides an execution engine to automate and run test cases.

The UI Automator testing framework provides a set of APIs to build user interface tests that perform interactions on user and system apps for Android. The UI Automator APIs allow you to perform operations such as opening the **Settings** menu or the app launcher in a test device.

**Integration:** Tests more than one individual module working together that implements a complete functional part of an app.

**DVM(Dalvik Virtual Machine):**

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimised for minimal memory footprint.

The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included “dx” tool.

**ADB(Android Debug Bridge):**

The [Android Debug Bridge](https://en.wikipedia.org/wiki/Android_Debug_Bridge" \t "https://wiki.lineageos.org/_blank) is a development tool that facilitates communication between an Android device and a

personal computer. This communication is most often done over a USB cable, but Wi-Fi connections are also supported. The adb command facilitates a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell that you can use to run a variety of commands on a device. It is a client-server program that includes three components:

* **A client**, which sends commands. The client runs on your development machine. You can invoke a client from a command-line terminal by issuing an adb command.
* **A daemon (adbd)**, which runs commands on a device. The daemon runs as a background process on each device.
* **A server**, which manages communication between the client and the daemon. The server runs as a background process on your development machine.

adb is included in the Android SDK Platform-Tools package.

Device will be in 3 states:

1. ADB Mode:Device up(Device should be active)
2. Fastboot Mode: Fastboot is a protocol that can be used to re-flash partitions on your device (update the flash file system in Android devices). It is this small tool that comes with the Android SDK (Software Developer Kit), which is an alternative to the Recovery Mode for doing installations and updates.While in fastboot, you can modify the file system images from a computer over a USB connection.
3. Download Mode:Download mode is a special state on Android devices. It is mainly used for ROM flashing or system update.

**ADB Commands:**

* adb shell  - launches a [shell](https://en.wikipedia.org/wiki/Shell_(computing)" \t "https://wiki.lineageos.org/_blank) on the device
* adb push <local> <remote> - pushes the file <local> to <remote>
* adb pull <remote> [<local>] - pulls the file <remote> to <local>. If <local> isn’t specified, it will pull to the current folder.
* adb logcat  - allows you to view the device log in real-time. You can use adb logcat -b radio to view radio logs,adb logcat -C to view logs in colour
* adb install <file> - installs the given .apk file to your device.
* adb start-server - To start the adb server
* adb kill-server -To stop the adb server
* adb devices -Will list all connected devices.
* adb uninstall com.myAppPackage - Remove the given app from the device.
* adb reboot-bootloader -To change from adb mode to fastboot mode
* Fastboot reboot -To come back from fastboot to reboot
* adb remount -Conversion from unrooted to rooted device
* adb logcat > logcat.txt -Redirects all the logs into file.

**LOGS:**

* The Log.v() method is used to log verbose messages.
* The Log.d() method is used to log debug messages.
* The Log.i() method is used to log informational messages.
* The Log.w() method is used to log warnings.
* The Log.e() method is used to log errors.
* radio: View the buffer that contains radio/telephony related messages.

adb logcat -b radio

* events: View the interpreted binary system event buffer messages.

adb logcat -b events

* main: View the main log buffer (default) does not contain system and crash log messages.
* system: View the system log buffer (default).
* crash: View the crash log buffer (default).
* all: View all buffers.
* Kernel:View the Kernel related information.

adb shell kernel

* Procrank:View the memory related information.

adb shell procrank

* Top:View the process related information

adb shell top

**User build:** Limited access; suited for production.

**User debug build: userdebug**, like "**user**" but with root access and debuggability; preferred for **debugging**.

**ANR(Application Not Responding):**ANRs are a problem because the app’s main thread, which is responsible for updating the UI, can’t process user input events or draw, causing frustration to the user. 

**Tombstone:**The tombstone is a file with extra data about the crashed process. In particular, it contains stack traces for all the threads in the crashing process (not just the thread that caught the signal), a full memory map, and a list of all open file descriptors.

**Android Features:**

* Open Source:[Open-source](https://www.androidcentral.com/tags/open-source) software is software that makes the source code freely available, for anyone to see and use.
* DVM:It is optimisation of jvm.Works for low battery/power management and low memory
* SQlite: SQLite Database has methods to create, delete, execute SQL commands, and perform other common database management tasks.SQLite is a open source SQL database that stores data to a text file on a device.

Drawbacks:SQLite is used to handle low to medium traffic HTTP requests.

Database size is restricted to 2GB in most cases.

* Notifications:A notification is a message that Android displays outside your app's UI to provide the user with reminders, communication from other people, or other timely information from your app. Users can tap the notification to open your app or take an action directly from the notification.
* OpenGL: OpenGL is a cross-platform graphics API that specifies a standard software interface for 3D graphics processing hardware.
* IOT:The **Internet of things** (**IoT**) is the network of physical devices, vehicles, home appliances and other items embedded with electronics,software,sensors,and connectivity which enables these objects to connect and exchange data.

**Android Components:**

* **Activities:** They dictate the UI and handle the user interaction to the smart phone screen.
* **Services:**They handle background processing associated with an application.
* **Broadcast Receivers:** They handle communication between Android OS and applications.
* **Content Providers:** They handle data and database management issues.

**Android App Structure:**

**main/**

Contains the "main" sourceset files: the Android code and resources shared by all build variants

**src/**

Contains all code and resource files for the module in the following subdirectories:

**java/**

Contains Java code sources.

**res/**

Contains application resources, such as drawable files(mdpi,hdpi,xhdpi), layout files(To design UI,using XML code) and values(strings,dimensions,style).

**libs/**

Contains private libraries.

**AndroidManifest.xml**

Describes the nature of the application and each of its components.

**Environment setup:**

**For Development:**

* **Android studio:** This is the official IDE (Integrated Development Environment) for the Android platform, developed by Google and used to make the majority of the apps that you probably use on a daily basis . Android Studio provides the fastest tools for building apps on every type of Android device.
* **SDK(Software Development Kit):** SDK is a set of programs used by a computer programmer to write [application](http://searchsoftwarequality.techtarget.com/definition/application) programs. Typically, an SDK includes a visual screen builder, an editor, a [compiler](http://whatis.techtarget.com/definition/compiler), a linker, and sometimes other facilities.
* **NDK(Native Development Kit):**The Android NDK is a toolset that lets you implement parts of your app in native code, using languages such as C and C++
* **JDK(Java Development Kit): JDK** is a software development environment used for developing Java applications . It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development.

**For Testing:**

* SDK
* JDK

**Two ways to test apk(application package Kit):**

* **Device:**

**Developer Option:**Settings > About phone > Build number

Tap *Build number* seven times. After the first few taps, you should see the steps counting down until you unlock the developer options. Once activated, you will see a message that reads, “You are now a developer!”

Go back to *Settings*, whereyou’ll find a *Developer options* entry in the menu.

**Unknown Resource:**The “unknown sources” term means anything other than the official Play Store. Installing apps from unknown sources means that you are now responsible for your phone’s safety.

* **Emulator/Simulator:**The Android SDK includes a virtual mobile device emulator that runs on your computer. The emulator lets you prototype, develop and test Android applications without using a physical device.