NXP Semiconductors

Document Number: AFAQ
Rev. O8.1.0_2.0.0-AUTO-beta, 01/2019

Android™ Frequently Asked Questions

1 How do I configure the build information?

For every build, define a BUILD ID and BUILD NUMBER. In the release package, the BUILD_ID is defined as an internal release build number and the BUILD_NUMBER is defined as an internal release date. Customize them in build_id.mk under ${MY_ANDROID}/device/fsl/xxxx/build_id.mk, where "xxxx" is the board name of the product.

The following is an example to update the BUILD_ID for the i.MX 8QuadXPlus/8QuadMax MEK board:

```
diff --git a/mek_8q/build_id_car.mk b/mek_8q/
build_id_car.mk
index 0dffaab..15b6e95 100644
--- a/mek_8q/build_id_car.mk
+++ b/mek_8q/build_id_car.mk
@@ -18,5 +18,5 @@
 # (like "CRB01").  It must be a single word,
and is
 # capitalized by convention.

-export BUILD_ID=1.1.0-auto-beta-rc2
+export BUILD_ID=1.1.0-auto-beta-rc3
 export BUILD_NUMBER=20180430
```

2 How do I download the Android source code behind a firewall?

If you have an HTTPS proxy and your firewall supports socks, perform the following steps:

1. Install Dante, which is a socks client.

   ```
   $ sudo apt-get install dante-client
   ```

2. Configure Dante by adding below lines into `/etc/dante.conf`.

   ```
   route {
   from: 0.0.0.0/0 to: .  via: DNS_OR_IP_OF_YOUR_SOCKS_SERVER port =
   PORT_OF_YOUR_SOCKS_SERVER
   proxyprotocol: socks_v5
   }       resolveprotocol: fake
   ```

3. Set the environment variable for HTTP proxy and socks.

   ```
   $ export https_proxy=...
   $ export SOCKS_USER=...
   $ export SOCKS_PASSWD=...
   ```

4. Download Android code from Google.

   ```
   $ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
   $ chmod a+x ~/bin/repo
   $ ~/bin/repo init -u https://android.googlesource.com/platform/manifest -b
   android-9.0.0_r8
   $ socksify ~/bin/repo sync
   ```

3 How do I use ADB over Ethernet?

Since Jelly Bean, security ADB is enabled by default (ro.adb.security is set to 1). In security ADB mode, the ADB over ethernet is not allowed. To use the ADB over ethernet, there are two steps to follow:

1. Disable the security ADB by changing ADB security setting in init.rc.
   Delete or comment the following line, and then rebuilt the boot.img.

   ```
   setprop ro.adb.security 0
   ```
2. Keep the board connecting with USB to the PC, and enable the ‘USB debugging’ from Setting->Developer Options and then follow the steps below to set up ADB over ethernet.

On the Linux® OS PC, assuming you had built Android code or had installed Android SDK, complete the following actions to use ADB over Ethernet:

```
 $ ping IP_OF_YOUR_BOARD (run "netcfg" on board to get IP address)
 $ export ADBHOST=IP_OF_YOUR_BOARD
 "adb" is a host tool created during Android build.It's under out/host/linux-x86/bin/. Make
sure you set path properly.
 $ adb kill-server (Not sure why this step is needed. Just re-start adb daemon on board.)
 $ adb shell
```

Set the ADB port properly on the device:

```
$ setprop service.adb.tcp.port 5555
```

After setting up the ADB listener port, re-enable the USB debug function in the Settings application.

4 How do I set up a computer to support ADB?

In this release, Google vendor ID and product ID are used for all Android gadget functions.

The user can download the latest Android SDK package and use ADB tool to test ADB function.

On a computer ruuning the Windows® OS, install Google extra win USB driver, contained in the SDK package, when Windows OS finds your device.

On a computer ruuning the Linux OS, add the following rules for udev rule file: /etc/udev/rules.d/51-android.rules

```
SUBSYSTEM=="usb", SYSFS{idVendor}=="18d1", MODE="0777"
SUBSYSTEM=="usb|usb_device", ATTR{idVendor}=="18d1", MODE="0666", GROUP="plugdev"
```

5 How do I set up a computer to support ADB in recovery mode?

Linux OS supports this feature by default if SDK is updated to a version later than Jelly Bean.

For Windows OS, follow the steps below:

1. Install the driver.
2. Apply the patch below to the USB driver from Google.
3. Connect the USB cable to the board and install the driver according to the instructions provided.

```
--- android_winusb.inf    2013-06-04 13:39:40.344756457 +0800
+++ android_winusb.inf    2013-06-04 13:43:46.634756423 +0800
@@ -23,6 +23,8 @@


[Google.NTx86]
+;adb sideload support
+%SingleAdbInterface%        = USB_Install, USB\VID_18D1amp;PID_D001

;Google Nexus One
%SingleAdbInterface%        = USB_Install, USB\VID_18D1amp;PID_0D02
@@ -59,7 +61,8 @@


[Google.NTamd64]
-
+;adb sideload support
+%SingleAdbInterface%        = USB_Install, USB\VID_18D1amp;PID_D001
;Google Nexus One
%SingleAdbInterface%        = USB_Install, USB\VID_18D1amp;PID_0D02
%CompositeAdbInterface%     = USB_Install, USB\VID_18D1amp;PID_0D02amp;MI_01
```

6 How do I enable USB tethering?

The USB tethering feature is supported in this release.

The upstream device can be Wi-Fi or Ethernet. USB tethering can be enabled in the Settings UI after your OTG USB cable is connected to computer: Settings -> WIRELESS & NETWORKS -> More.. -> Tethering & portable hotspot -> USB tethering. In the meantime, make sure you have disabled ADB.

On a Linux OS computer, when USB tethering is enabled, you can easily get an USB network device. The IP and DNS server is automatically configured.

On a Windows OS computer, when you have connected the board with the computer and you can see an unknown device named "Android" in the device manager, you have to manually install the tethering driver by the tetherxp.inf file in android__tools.tar.gz. After it is successfully installed, the "Android USB RNDIS device" is displayed in the device manager. By this time, you can use USB RNDIS network device to access the network.

7 How do I use MTP?

The Media Transfer Protocol is a set of custom extensions to the Picture Transfer Protocol (PTP).

Whereas PTP was designed for downloading photographs from digital cameras, Media Transfer Protocol supports the transfer of music files on digital audio players and media files on portable media players, as well as personal information on personal digital assistants.

Starting with version 4.0, the Android platform supports MTP as a default protocol transfer files with computer, instead of the USB Mass Storage. In this release, as suggested by Google, we disabled the UMS and enabled MTP.

NOTE

Ensure that you disable the USB Tethering when using MTP. In the Windows® XP OS, you cannot make MTP work with ADB enabled. In the Windows® 7 OS, in theory MTP can work together with ADB, but it is also found that some hosts with the Windows 7 OS fail to support it.

When connecting the board to the computer by USB cable, an USB icon is shown in the notification bar. Then, you can click on the notification area, and select "Connected as a media device" to launch the USB computer connection option UI. There, MTP and PTP can be chosen as current transfer protocol. You can also launch the option UI by Settings -> Storage -> MENU -> USB computer connection.

MTP on the Windows XP OS

Windows XP supports PTP protocol by default. In order to support MTP protocol, you must install Windows Media Player (Version >= 10). When connecting to a computer, you can see MTP devices in Windows Explorer. Since Windows XP only supports copy/paste files in the explorer, you cannot directly open the files in MTP device.

MTP on the Windows 7 OS

Windows 7 supports MTP (PTP) protocol by default. When connecting to a computer, you can see MTP devices in Windows Explorer. You can perform any operations just as you would on your hard disk.

MTP on Ubuntu

Ubuntu supports PTP protocol by default. To support MTP protocol, you must install the following packages: libmtp, mtp-tools by using the following command:

```
$ sudo apt-get install mtp-tools
```

If your default libmtp version is not 1.1.1 (current latest libmtp on ubuntu is 1.1.0), you must upgrade it manually by:

```
$ sudo apt-get install libusb-dev
$ wget http://downloads.sourceforge.net/project/libmtp/libmtp/1.1.1/libmtp-1.1.1.tar.gz
$ tar -xvf libmtp-1.1.1.tar.gz
$ cd libmtp-1.1.1
$ ./configure --prefix=/usr
$ make
$ sudo make install
```

After you have done the steps outlined above, you can transfer the files between the computer and the device by using the following commands:

- mtp-detect finds current connected MTP device.
- mtp-files lists all the files on MTP device.
- mtp-getfile gets the files on MTP device by file ID listed by mtp-files.
- mtp-sendfile puts files onto MTP device.

There's an alternative GUI application, called gMtp, which makes it easier to access MTP device instead of using the commands above. You can install it by using the following command:

```
$ sudo apt-get install gmtp
```

After installation, you can launch gmtp and access MTP device in the file explorer.

8 How do I enter the Android recovery mode manually?

When the system is running, press **VOLUME DOWN** and **Power** to enter Recovery mode if the board has these keys. This check is in u-boot.git board support file, where you can change your preferred combo keys.

Also, you can input this command in the console:

```
reboot recovery              # the board reset to recovery mode.
```

to enter recovery mode.

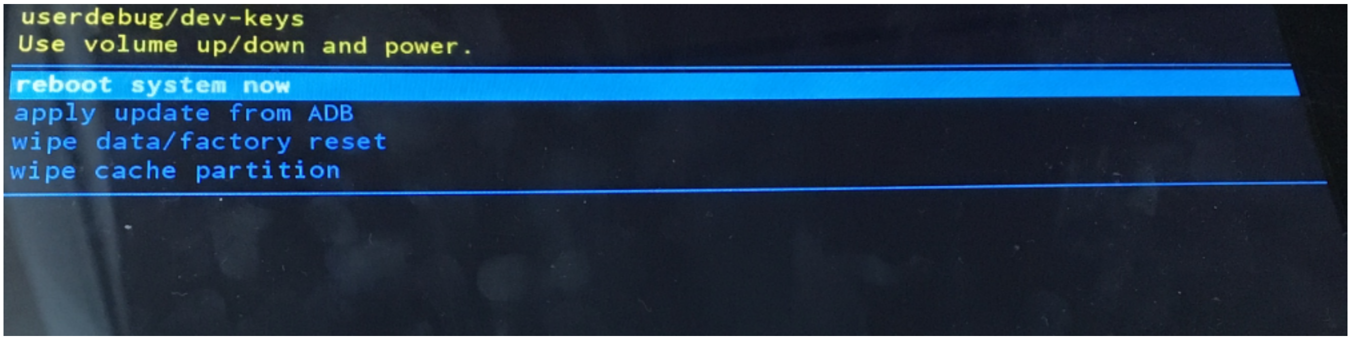
8.1 How do I enter the text menu in recovery mode?

NOTE

> This function only works on boards that have POWER/VOLUME UP/VOLUME DOWN keys.

To enter the text menu in recovery mode, follow the steps below:

1. When the system completes bootup, an Android Robot Logo displays.
2. Press the POWER KEY (keep pressed), and use the VOLUME DOWN KEY as follows:



```
userdebug/dev-keys
Use volume up/down and power.
reboot system now
apply update from ADB
wipe data/factory reset
wipe cache partition
```

Figure 1. How do I enter the Text menu in recovery mode

3. Move the menu item by VOLUME UP and VOLUME DOWN button.
4. Select the menu item by Power Key.
5. Select the required option using the direction keys on the keypad or keyboard.
6. Reboot the system
7. Apply updates from ADB. Note that you may update the software from update.zip by using the adb sideload command.

8. Wipe data/factory reset. /data and /cache partitions are formatted.
9. Wipe cache partition. /cache partition is formatted.
10. Reboot the system.

8.2 How do I use Android fastboot?

Fastboot is a feature which can be used to download images from a computer running either Windows OS or Linux OS to the target storage device.

This feature is released by Google in the Android SDK package, which can be downloaded from Android official site. Android release implements part of the fastboot commands in U-Boot such as: flash, reboot, getvar.

Before using fastboot, Android SDK must be installed on the host, and the target board must boot up to bootloader. Also, before using fastboot, U-Boot must be downloaded to the MMC/SD device with all the partitions created and formatted. Setup the correct board dip switches to boot up the board with U-Boot.

NOTE

The size of images downloaded by fastboot must be less than the size of the system partition.

Target side:

1. Power on the board with USB OTG connected.
2. Press any key to enter the U-Boot shell.
3. Select the correct device to do fastboot image download by command:
    - SD/MMC card

```
U-Boot > setenv fastboot_dev mmc3
```

- Run the fastboot command:

```
U-Boot > fastboot
               fastboot is in init......USB Mini b cable Connected!
               fastboot initialized
               USB_SUSPEND
               USB_RESET
               USB_RESET
```

Or launch the quick fastboot by "fastboot q" command

```
U-Boot > fastboot q
fastboot is in init......flash target is MMC:1
USB Mini b cable Connected!
```

Or you can input this command in the kernel:

```
# reboot bootloader              # the board reset to fastboot mode.
```

All commands can enter fastboot mode.

NOTE

- On a host computer, it prompts you that a new device is found and that you need to install the driver.
- The quick fastboot is a new implementation for fastboot utility. It increases the image download speed from computer to board up to about 28 MB/s, compared to the previous 1 MB/s. It only currently supports download and flash command now, which indicates that it supports image download.

Host side:

1. Enter the Android SDK tools directory and find the fastboot utility (fastboot.exe on the Windows OS, fastboot on the Linux OS).
2. Copy all downloaded images to the "images" folder.
3. Run the following commands to flash the SD or eMMC:

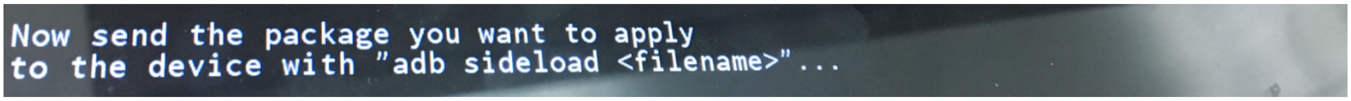
```
$ fastboot flash bootloader images\u-boot.imx
$ fastboot reboot bootloader
$ fastboot flash gpt images\partition-table.img
$ fastboot flash boot_a images\boot.img
$ fastboot flash boot_b images\boot.img
$ fastboot flash vbmeta_a images\vbmeta.img
$ fastboot flash vbmeta_b images\vbmeta.img
$ fastboot flash system_a images\system.img
$ fastboot flash system_b images\system.img
$ fastboot flash vendor_a images\vendor.img
$ fastboot flash vendor_b images\vendor.img
$ fastboot reboot
```

8.3 How do I update the system using ADB?

Before upgrading the system with the ADB tool, Windows OS users first need to install the ADB driver. To do so, see Section How do I set up a computer to support ADB in recovery mode?.

After the installation and setup of the driver is complete, follow the steps below:

1. Ensure that the system has entered recovery mode.
2. See How do I enter the Android recovery mode manually? and toggle the text menu.
3. Move the cursor to “Apply update from ADB.” The UI is dipslayed as follows.



```
Now send the package you want to apply
to the device with "adb sideload <filename>"...
```

Figure 2. How to upgrade system by ADB

4. On your computer, carry out the folllowing commands:

```
adb sideload $YOUR_UPDATE_PACKAGE.zip
```

5. After the package is sent, the system starts updating the firmware with the update file.

9 What is the key mapping of the USB keyboard?

The default DELL USB keyboard key mapping is defined as shown below.

Table 1. Default DELL USB keyboard key mapping

Key	Act as
ESC	BACK
F1	MENU
F2	SOFT_RIGHT
F3	CALL
F4	ENDCALL

Table continues on the next page...

Table 1. Default DELL USB keyboard key mapping (continued)

Key	Act as
F5	ENDCALL
F8	HOME
F9	DPAD_CENTER
UP	DPAD_UP
DOWN	DPAD_DOWN
BACK	DEL
ENTER	ENTER

10 How do I generate uramdisk.img?

To generate a RAMDISK image recognized by U-Boot, perform the following operations:

NOTE

Uramdisk is not used anymore.

Assume you have already built U-Boot and mkimage is generated under ${MY_ANDROID}/vendor/nxp-opensource/uboot-imx/tools/.

```
$ cd ${MY_ANDROID}/out/target/product/mek_8q
$ ${MY_ANDROID}/vendor/nxp-opensource/uboot-imx/tools/mkimage
-A arm -O linux -T ramdisk -C none -a 0x10308000 -n "Android Root Filesystem" -d ./
ramdisk.img
./uramdisk.img
```

11 How do I generate boot.img?

Use the following commands to generate boot.img

```
$ mkbootimg --kernel <kernel, Image> --ramdisk <ramdisk> --base <baseaddr> -second <board
dtb file> --cmdline <kernel command line> --board <board name> -o <output>
$ cd ${MY_ANDROID}
$ out/host/linux-x86/bin/mkbootimg --kernel vendor/nxp-opensource/kernel_imx/arch/arm64/boot/
Image --ramdisk
ramdisk.img --base 0x10800000 -second out/target/product/mek_8q/imx8q-mek.dtb --cmdline
"console=ttymxc0,115200 init=/init rw video=mxcfb0 vmalloc=800M" --board {board_name} -o
boot.img
```

Replace the {board_name} with your product name, such as mek_8q.

NOTE

To extract and edit the Image and ramdisk in the boot.img, see HOW to Unpack, Edit, and Re-Pack Boot Images.

12 How do I change the boot command line in boot.img?

After using boot.img, we store the default kernel boot command line inside this image.

It is packaged together during an Android build.

You can change this by changing BOARD_KERNEL_CMDLINE which is defined in ${MY_ANDROID}/device/fsl/{product}/BoardConfig.mk file.

NOTE

Replace {product} with your product, such as mek_8q.

13 How do I customize the boot animation?

The user can create his/her own boot animation for his/her device.

The Android platform provides an easy way to replace its default boot animation by putting bootanimation.zip file into /system/media/.

- To create your own bootanimation .zip file, see How To Change, Customize, and Create Android Boot Animation.
- How to install the boot animation
  - On the host, use ADB to download the bootanimation.zip file into the device, for example:

    ```
    $ adb push ~/Downloads/bootanimation.zip /mnt/sdcard
    ```
  - On the device, remount the /system to writable, and copy the file:

    ```
    $ mount -o remount -w /system
    $ busybox cp /mnt/sdcard/bootanimation.zip /system/media/
    $ mount -o remount -r /system
    ```

14 Why cannot certain APKs run on a device without a modem?

Some games require the TelephonyManager to return a device ID by getDeviceId() method of TelephonyManager, which is actually to return the mobile IMEI code with modem connected, but **null** without a modem.

They do not check the return value of getDeviceId(). Therefore, you can probably use null as device id without doing NULL as shown below:

```
TelephonyManager localTelephonyManager =
(TelephonyManager)oAppMain.getSystemService("phone");
      String str;
      if (localTelephonyManager != null)
      {
        nativeAddProperty("IMEI", localTelephonyManager.getDeviceId());
        Object[] arrayOfObject = new Object[1];
        arrayOfObject[0] = Integer.valueOf(Integer.parseInt(Build.VERSION.SDK));
        nativeAddProperty("DeviceID", String.format("%d", arrayOfObject));
        str = oAppMain.getClass().getPackage().getName();
        nativeAddProperty("Identifier", str);
      }
```

On the platform, when there is no modem connected, the TelephonyManager.getDeviceId() returns null. Therefore, the JNI calling from here nativeAddProperty crashes in the dalvik JNI module, which try to get strings from a null pointer.

For the tablet customer who does not have a modem connected, a workaround may need to be applied into framework/base.git:

```
diff --git a/telephony/java/android/telephony/TelephonyManager.java
b/telephony/java/android/telephony/TelephonyMa
index db78e2e..82cf059 100755
--- a/telephony/java/android/telephony/TelephonyManager.java
```

```
+++ b/telephony/java/android/telephony/TelephonyManager.java
@@ -192,6 +192,9 @@ public class TelephonyManager {
      *   {@link android.Manifest.permission#READ_PHONE_STATE READ_PHONE_STATE}
      */
     public String getDeviceId() {
+       String s = "2222222222";
+       return s;
+       /*
         try {
             return getSubscriberInfo().getDeviceId();
         } catch (RemoteException ex) {
@@ -199,6 +202,7 @@ public class TelephonyManager {
         } catch (NullPointerException ex) {
             return null;
         }
+       */
     }
```

To verify your IMEI hard code, start the phone application and dial *#06#. It shows the IMEI code.

15 How do I enable or disable the bus frequency feature?

The Bus Frequency driver is used to slow down DDR, AHB, and AXI bus frequency in the SoC when the IPs which need high bus frequency are not working.

This saves the power consumption in Android earlysuspend mode significantly (playing audio with screen off). In this release, the bus frequency driver is **enabled** by default. To enable or disable it, perform the following command in the console:

```
Disable:
$ echo 0 > /sys/bus/platform/drivers/imx6_busfreq/busfreq.x/enable
Enable:
$ echo 1 > /sys/bus/platform/drivers/imx6_busfreq/busfreq.x/enable
```

Note that if you are using Ethernet, the up operation enables the FEC clock and force bus frequency to be high. That means you cannot go into low bus mode anymore, regardless whether the Ethernet cable is plugged or unplugged. Therefore, if you want the system to go into the low bus mode, execute the 'netcfg eth0 down' command to shut down the FEC manually. To use the FEC again, do 'netcfg eth0 up' manually. When FEC is shut down with clock gated, the PHY cannot detect your cable in/out events.

16 How do I set networking proxy for Wi-Fi?

To configure the proxy settings for a Wi-Fi network, do as follows:

1. Tap and hold a network from the list of added Wi-Fi networks.
2. Select "Modify Network".
3. Choose "Show advanced options".
4. If no proxy settings are present in the network, you have to - Tap "None", Select "Manual" from the menu that opens.
5. Enter the proxy settings provided by the network administrator.
6. Finally tap on the button denoted as "Save"

17 How do I configure the logical display density?

The Android UI framework defines a set of standard logical densities to help the application developers target application resources. Device implementations MUST report one of the following logical Android framework densities:

- 120 dpi, known as 'ldpi'
- 160 dpi, known as 'mdpi'
- 213 dpi, known as 'tvdpi'
- 240 dpi, known as 'hdpi'
- 320 dpi, known as 'xhdpi'
- 480 dpi, known as 'xxhdpi'

Device implementations SHOULD define the standard Android framework density that is numerically closest to the physical density of the screen, unless that logical density pushes the reported screen size below the minimum supported. To configure the logical display density for framework, you must define the following line in the init.freescale.rc:

```
setprop ro.sf.lcd_density <density>
```

18 How do I enable developer settings on Android Jelly Bean and later versions?

Google has hidden the developer settings since the version of Jelly Bean. The following steps explain how to retrieve developer settings:

- Go to the settings menu and scroll down to "About tablet." Tap it.
- Scroll down to the bottom again until you see "Build number."
- Tap it seven (7) times. After the third tap, you'll see a playful dialog that says you're four taps away from being a developer.
- Keep on tapping, until you've got the developer settings back.

19 How do I reduce the RTSP streaming latency?

The OMXPlayer has a cache buffer 4 seconds of data in size. This buffer introduces some latency for audio/video streaming. To reduce the latency, you can modify the code below to control the cache size.

```
In file: ${MY_ANDROID}/external/fsl_imx_omx/OpenMAXIL/src/component/streaming_parser/
StreamingParser.cpp
    #define PACKET_CACHE_SIZE (4*OMX_TICKS_PER_SECOND)
```

This feature is available as part of the Extended Multimedia Feature Package. For more information and details about the package, send inquiry to "L2manager-android@nxp.com".

20 How do I set GPU minimal clock to balance performance and power consumption?

Normally GPU works at full speed. When thermal driver reports that the chip is too hot, the GPU driver adjusts the internal clock to reduce the power consumption and quickly cool down the chip. In theory the GPU clock should be set to 1/64 to ensure that chip can cool down faster. However, you may see a black screen or experience a flickering issue when the GPU works with such a slow clock especially in large resolutions (for example 1080P).

The steps below show how to customize the threshold of the GPU minimal clock based on the chip and the resolution of their product.

Customer can set the minimal GPU clock by adding the line below in init.rc file. The value can be set to any value from 1 to 64. The current default value is 3. The recommended value can be 3 on all i.MX 8 SoCs. A customer should tune and set a suitable value based on their test.

Write

```
/sys/module/galcore/parameters/initgpu3DMinClock 1
```

21 How do I build an OTA package?

The Android build system supports auto generation of update.zip. It can generate the updater_script and all system.img files.

You can use the following command to generate an OTA package.

```
$ make otapackage
```

For example, you use this command to build mek_8q_car product after the build finishes.

```
$ make PRODUCT=mek_8q_car-userdebug otapackage -j4
```

You can find a few packages in the following path for the OTA update.zip to do the OTA and make a different package:

out/target/proudct/mek_8q/mek_8q_car-ota-date-soc.zip

22 How do I sign the OTA package with my digital keys?

The Android platform requires that each application be signed with the developer's digital keys to enforce signature permissions and application request to use shared user ID or target process. For more information on the general Android security principles and signing requirements, see Section "System Permissions" in the Android Developer Guide. The core Android platform uses four keys to maintain security of core platform components:

- **platform**: a key for packages that are part of the core platform.
- **shared**: a key for content shared in the home/contacts process.
- **media**: a key for packages that are part of the media/download system.
- **releasekey**: default key to sign with if nothing specified.

These keys are used to sign applications separately for release images and are not used by the Android build system. The build system signs packages with the testkeys provided in build/target/product/security/. Because the testkeys are part of the standard Android open source distribution, they should never be used for production devices. Instead, device manufacturers should generate their own private keys for shipping release builds.

Generating keys

A device manufacturer's keys for each product should be stored under vendor/<vendor_name>/security/<product_name>, where <vendor_name> and <product_name> represent the manufacturer and product names. To simplify key creation, copy the script below to this directory in a file called mkkey.sh. To customize your keys, change the line that starts with AUTH to reflect the correct information for your company:

```
#!/bin/sh
AUTH='/C=US/ST=California/L=Mountain View/O=Android/OU=Android/CN=Android/
emailAddress=android@android.com'
if [ "$1" == "" ]; then
        echo "Create a test certificate key."
        echo "Usage: $0 NAME"
        echo "Will generate NAME.pk8 and NAME.x509.pem"
        echo "  $AUTH"
        exit
fi

openssl genrsa -3 -out $1.pem 2048

openssl req -new -x509 -key $1.pem -out $1.x509.pem -days 10000 \
    -subj "$AUTH"

echo "Please enter the password for this key:"
openssl pkcs8 -in $1.pem -topk8 -outform DER -out $1.pk8 -passout stdin
```

mkkey.sh is a helper script used to generate the platform keys.

The password that you enter is displayed in your terminal window. You need the password to sign release builds.

To generate the required four platform keys, run mkkey.sh four times, specifying the key name and password for each:

```
$sh mkkey.sh platform # enter password
$sh mkkey.sh media # enter password
$sh mkkey.sh shared # enter password
$sh mkkey.sh release # enter password
```

You should now have new keys for your product.

Signing a build for release

To sign a build for a release, perform the following steps:

1. Sign all the individual parts of the build.
2. Put the parts back together into image files.

Signing applications

Use build/tools/releasetools/sign_target_files_apks to sign a target_files (have all files, system and recovery.img boot.img) package. The target_files package is not built by default, specify the "dist" target when you call "make". For example:

```
make -j4 PRODUCT-<product_name>-user dist
```

This command above creates a file under out/dist called <product_name>-target_files.zip. This is the file you need to pass to the sign_target_files_apks script.

You would typically run the script like this:

```
./build/tools/releasetools/sign_target_files_apks -d vendor/<vendor_name>/security/
<product_name> <product_name>-target_files.zip signed-target-files.zip
```

If you have prebuilt and pre-signed APKs in your build that you do not want re-signed, you must ignore them by adding `-e Foo.apk=` to the command line for each APK you wish to ignore.

sign_target_files_apks also has many other options that could be useful for signing release builds. Run it with `-h` as the only option to see the full help.

Creating image files

Once you have signed-target-files.zip, create the images so that you can put it onto a device with the command below:

```
build/tools/releasetools/img_from_target_files signed-target-files.zip signed-img.zip
```

signed-img.zip contains all the .img files. You can use fastboot in fastboot update signed-img.zip to get them on the device.

23 How do I generate a different OTA package?

Assuming you can build the OTA package sabresd_6dq_target_files-eng.xxx.zip by following the steps in "How do I build an OTA package?", this file is a full package of your current build image. Save this file in a directory. For example:

```
mkdir ~/release-1
cp out/dist/imx6q_sabrelite_target_files-eng.xxx.zip ~/release-1
```

If you find a bug or if you want to generate a new release package later, do the same operation to make a "dist" build.

Save it to a new place, for example, ~/release-2

```
mkdir ~/release-2
cp out/dist/imx6q_sabrelite_target_files-eng.xxx.zip ~/release2
```

To generate a different package, carry out the following command:

```
cd mydroid; # you must be in your Android root dir to do this command
./build/tools/releasetools/ota_from_target_files -i ~/release-1/imx6q_sabrelite_target_files-
eng.xxx.zip \
~/release-2/imx6q_sabrelite_target_files-eng.xxx.zip ~/diff-from-release-1-to-2.zip
```

~/diff-from-release-1-to-2.zip is a diff package between release 1 and release 2.

In this example, a kernel commit and a framework/base are changed, and the zip package is only 4.5 MB.

24 How do I upgrade a/b system with OTA package?

The server side of this OTA application is a common HTTP server, such as lighttpd and apache.

The following is a unzip command output of a sample OTA site. payload.bin is OTA payload. payload_properties.txt records the information of payload.

```
$ unzip mek_8qm.ota.zip
Archive:  mek_8qm.ota.zip
signed by SignApk
 extracting: META-INF/com/android/metadata
 extracting: care_map.txt
 extracting: payload.bin
 extracting: payload_properties.txt
 inflating: META-INF/com/android/otacert
```

The following are the commands to upgrade the system. The 10.192.224.178 server is set as the OTA server.

The parameter in the headers is from the payload_properties.txt file. Make sure to use a new line (press **ENTER**) for every payload_properties parameter.

```
update_engine_client --payload=http://10.192.224.178:10888/mek_8q_car_oreo_8.1.0/payload-
imx8qxp.bin --update --headers="FILE_HASH=plr0c+Z2gNHQtoGL95BCWmRd5GIlb2czXOdZ6U5/XRI=
FILE_SIZE=377407025
METADATA_HASH=BOMQx53ELPt9RVazIXY4z8ZbWFMu7Q9dOs65jQ3IzbY=
METADATA_SIZE=46863"
```

The mek_8qm.ota.zip file is the output from "How do I build an OTA package?". Reboot the system after a/b OTA succeeds.

25 How do I customize the update script to update U-Boot?

Because the Android platform only upgrades the boot.img, system.img, and recovery partitions, the automatically generated update package does not support upgrading bootloader. To upgrade the bootloader, modify the update package and perform the signing work manually.

1. Unzip the update.zip, and then modify the updater_script by implementing the following operations.

   To upgrade U-Boot to NOR flash, see this script:

   ```
   ui_print("writting U-Boot...");
   write_raw_image("u-boot.bin", "/dev/mtd0");
   show_progress(0.1, 5);
   ```

   To upgrade U-Boot for eMMC storage, because U-Boot may be stored in the "boot partition" of eMMC, perform system file operations before dd, for example,

   ```
   # Write U-Boot to 1K position.
   # U-Boot binary should be a no padding U-Boot!
   # For eMMC(iNand) device, needs to unlock boot partition.
   ui_print("writting U-Boot...");
   package_extract_file("files/u-boot-no-padding.bin", "/tmp/u-boot-no-padding.bin");
   sysfs_file_write("class/mmc_host/mmc0/mmc0:0001/boot_config", "1");
   simple_dd("/tmp/u-boot-no-padding.bin", "/dev/block/mmcblk0", 1024);
   sysfs_file_write("class/mmc_host/mmc0/mmc0:0001/boot_config", "8");
   show_progress(0.1, 5);
   ```

2. Resign the update package by using the following command:

   ```
   $ make_update_zip.sh ~/mydroid ~/update-dir
   ```

26 How do I make a fake battery and charger status report to some applications?

There is no battery and charger in the i.MX 6DualQuad/6DualLite SABRE-AI board and i.MX 6SoloLite EVK board. The pre-condition of certain features or functions of specific application is the battery or charger status, such as data partition encryption features in the Setting application. Taking i.MX 6SoloLite EVK board as an example, you can make the system to report a fake battery and charger status to enable such certain features or functions as below. It makes the system show in fake 100% level of battery and AC charger plugged in.

```
diff --git a/evk_6sl/init.rc b/evk_6sl/init.rc
index 934828a..82f9c3f 100644
--- a/evk_6sl/init.rc
+++ b/evk_6sl/init.rc
@@ -19,6 +19,9 @@ on init
```

```
on boot

+    # emmulate battery property
+    setprop sys.emulated.battery 1
+
     # Set permission for IIM node
     symlink /dev/mxs_viim /dev/mxc_mem
```

27 How do I disable DM-verity?

Android platform 4.4 and later versions support verified boot through the optional device-mapper-verity (DM-verity) kernel feature, which provides transparent integrity checking on block devices. DM-verity helps prevent persistent rootkits that can hold onto root privileges and compromise devices. This experimental feature helps Android users be sure the booting device in the same state as when it was last used.

BOARD_SYSTEMIMAGE_PARTITION_SIZE defines the size of raw system.im in device/fsl. The sparse system.img contains sparse system.img, FEC image, and meta data. The meta data is used to verify the system partition. Dm_verity reads the last data on the block device as meta data.

You can disable the dm-verity through the following commands:

```
$adb root
$adb disable-verity
$adb reboot
```

28 How do I change the RSA for DM-verity?

RSA keys are used to sign the dm_verity table to produce a table signature. When verifying a partition, the table signature is validated first. This is done against a key on your boot image in a fixed location. Keys are typically included in /verity_key.

The 2048-bit private RSA key that is used to sign a table is generated by openSSL. It is included in build/target/product/security/verity_private_dev_key in the Android project.

The RSA public key used for verification needs to be in mincrypt format. Converting an OpenSSLRSA public key to mincrypt format requires some modular operations and it is not simply a binary format conversion.You can convert the PEM key using the pem2mincrypt tool. The public key is included in build/target/product/security/verity_key. The following commands change the default RSA key in the Android project.

```
cd build/target/product/security/
openssl genrsa -out verity_private_dev_key_tem 2048
openssl pkcs8 -topk8 -inform PEM -in verity_private_dev_key_tem  -out
verity_private_dev_key  -outform PEM -nocrypt
pem2mincrypt verity_private_dev_key_tem  verity_key
```

NOTE

You should install libssl0.9.8 with the following command:

```
$sudo apt-get install libssl0.9.8
```

The tool pem2mincrypt’s source code is in the following location: github.com/nelenkov/verity

29 How do I program the sparse system image into an SD Card?

The default system.img is in sparse format. It needs to be converted to raw image before flashing into SD Card. Below are the steps to do the work in mfgtool.

```
<CMD state="Updater" type="push" body="$ mount -o remount,size=800M rootfs /">change size of
tmpfs</CMD>
<CMD state="Updater" type="push" body="send" file="files/android/%board%/system.img">Sending
system.img</CMD>
<CMD state="Updater" type="push" body="$ simg2img $FILE /dev/mmcblk%mmc%p5">writting sparse
system.img</CMD>
```

1. Enlarge the size of rootfs.

   The default size of the rootfs is 800 MB. The size of sparse system.img may be larger than this. The rootfs size needs to be enlarged to hold the system.img. The size should be smaller if the memory is less than 800 MB.

2. Send the sparse image to rootfs.
3. Convert the sparse image to a raw image.

   simg2img is a tool that converts a sparse system image to a raw system image. The simg2img is located in init ramfs by default.

30 How do I disable GPU acceleration?

There are three parts using GPU acceleration on the Android platform. To reduce issues, perform the following operations to disable some of them separately:

1. Disable HWComposer.

   Select **Setting Application**, and then select **Settings** -> **{} Developer options** -> **Disable HW overlays**.

2. Disable OpenGL Renderer.

   You can disable OpenGL Renderer and use SKIA by force to draw the Android Application UI by setting “setprop sys.viewroot.hw false” and killing the surfaceflinger thread.

3. Disable OpenGL 3D draw.

   OpenGL 3D draw can be disabled only after OpenGL Renderer is disabled, as this operation may totally disable all 3D OpenGL accelerations. You can do it by “mv /system/lib/egl/libGLES_android.so /system/lib/egl/libGLES.so” and killing the surfaceflinger thread.

NOTE

The following example shows how to kill the surfaceflinger:

```
root@sabresd_6dq:/ # ps | grep surfaceflinger
system    159   1     168148 7828  ffffffff b6f05834 S /
system/bin/surfaceflinger
root@sabresd_6dq:/ # kill 159
```

31 Revision History

Table 2. Revision history

Revision number	Date	Substantive changes
O8.1.0_1.1.0_AUTO-EAR	02/2018	Initial release
O8.1.0_1.1.0_AUTO-beta	05/2018	i.MX 8QuadXPlus/8QuadMax Beta release
O8.1.0_2.0.0-AUTO-beta	01/2019	i.MX 8QuadXPlus/8QuadMax Beta release