Wireless Module Expert

Quectel Wireless Solutions Co.,Ltd.

Android 升级方案简述

文档标题	Android 升级方案简述
版本	V0.1
日期	2015-12-24
状态	

目录

修改记录

版本	日期	作者	修改内容记录
1.0	2015-12-24	Law.zhu	初始版本
2.0	2015-12-27	Law.zhu	修改内容排版
3.0	2016-1-4	Law.zhu	新加入内容
4.0	2016-1-25	Law.zhu	补充升级包签名原理 2.6 节

相关参考文档

表 1: 参考文档表

序号	文档名	备注
1	80-NL409-1_FIRMWARE OVER THE AIR (FOTA) WORKFLOW	

Android 升级方案简述

本文主要参考 80-NL409-1_FIRMWARE OVER THE AIR (FOTA) WORKFLOW.pdf；


Qualcomm Technologies, Inc.

Firmware Over the Air (FOTA) Workflow

Application Note

80-NL409-1 C

July 17, 2015

文档中主要描述 Full OTA 升级和 Incremental OTA 升级两种，本文分别介绍

1. SC10 Full OTA 方式升级介绍

Full OTA 升级就是对整个下载包做打包，然后通过下载到 SD 卡，用 recover 系统执行升级动作。操作步骤如下说明。

升级对象：boot.img，cache.img, system.img, userdata.img, emmc_appsboot.mbn, non-hlos.bin, sbl1.mbn, rz.mbn, rpm.mbn ….

1.1. Full OTA 制作第一步：生成 msm8909-target_files-eng.XXX.zip

首先在我们通过 make 编译了整个代码后会在代码的 out 目录下自动生成制作整包的原始文件：

out/target/product/msm8909/obj/PACKAGING/target_files_intermediates 下会有 msm8909-target_files-eng.XXX.zip 这个文件，这个文件就是制作这个版本全包的原始文件.

但是这个包中只是包含了安卓部分的整包制作原始材料，如果要对于 modem 等非 AP 部分的整包制作我们可以这样做：

1.2. Full OTA 制作第二步：Modem 等非 HLOS 加入升级包的方法

有两种方法：

官方，也就是高通给出的方案是这样的：

Create a folder named RADIO in the path /device/qcom/<target>/ and add the non-HLOS files (non-hlos.mbn, tz.mbn, rpm.mbn, etc.) that must be upgraded into this folder.

在安卓代码目录下 device/qcom/<target>/ ：

对于我们 8909 就是：

device/qcom/msm8909 下创建 radio 这个文件夹：

```
AndroidBoard.mk      init.qcom.modem_links.sh        recovery.fstab
Android.mk           init.qti.synaptics_dsx_qhd.sh   recovery_nand.fstab
AndroidProducts.mk   init.target.rc                  snd_soc_msm
atmel_mxt_ts.kl      listen_platform_info.xml        sound_trigger_mixer_paths.xml
audio_effects.conf   media                           sound_trigger_platform_info.xml
audio_policy.conf    mixer_paths_msm8909_pm8916.xml  synaptics_dsx.kl
BoardConfig.mk       mixer_paths_qrd_skuh.xml        synaptics_rmi4_i2c.kl
cdrom_res            mixer_paths_qrd_skui.xml        system.prop
egl.cfg              mixer_paths_skua.xml            vold.fstab
fstab.qcom           mixer_paths_skuc.xml            WCNSS_qcom_cfg.ini
ft5x06_ts.kl         mixer_paths_skue.xml            WCNSS_qcom_wlan_nv.bin
gpio-keys.kl         mixer_paths.xml                 WCNSS_wlan_dictionary.dat
hostapd.accept       msm8909.mk                      wpa_supplicant.conf
hostapd.conf         overlay                         wpa_supplicant_overlay.conf
hostapd.deny         p2p_supplicant_overlay.conf
init.carrier.rc      radio
root@will-OptiPlex-790:/home/sdc/law/bak/sc10/device/qcom/msm8909#
```

这个文件默认已经存在了，所以我们在编译版本时只要将非 AP 部分加入到这个文件夹中就可以了。

比如我们可以先将 modem 部分加入到这里：

cp (modem 所在路径) ./device/qcom/msm8909/radio

我们看下这个文件中有什么文件：

```
root@will-OptiPlex-790:/home/sdc/law/bak/sc10/device/qcom/msm8909/radio# ls
filesmap
```

这个名为 filesmap 的文件内容是这样的：

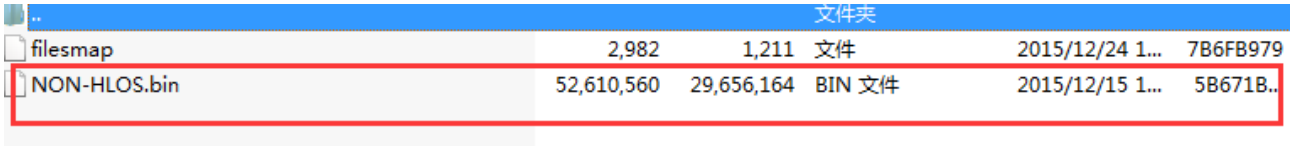
```
# filename              partition

NON-HLOS.bin            /dev/block/bootdevice/by-name/modem

sbl1.mbn                /dev/block/bootdevice/by-name/sbl1
tz.mbn                  /dev/block/bootdevice/by-name/tz
rpm.mbn                 /dev/block/bootdevice/by-name/rpm
emmc_appsboot.mbn       /dev/block/bootdevice/by-name/aboot
hyp.mbn                 /dev/block/bootdevice/by-name/hyp

# filename + .bak       backup partition

sbl1.mbn.bak            /dev/block/bootdevice/by-name/sbl1bak
tz.mbn.bak              /dev/block/bootdevice/by-name/tzbak
rpm.mbn.bak             /dev/block/bootdevice/by-name/rpmbak
emmc_appsboot.mbn.bak   /dev/block/bootdevice/by-name/abootbak
hyp.mbn.bak             /dev/block/bootdevice/by-name/hypbak

# For multiple file firmware images that differ from *.mbn and *.bin
# you can specify filename.* to direct all files to the same location.
# For example for modem.mdt, modem.b00, modem.b01,... modem.bxx files
# writting 'modem.*   location' will direct all files to 'location'.
# If still some files need to go to different location give the full
# file name also, for example 'modem.b01   other_location'

# filename              location

modem.*                 /dev/block/bootdevice/by-name/modem
wcnss.*                 /dev/block/bootdevice/by-name/modem
widevine.*              /dev/block/bootdevice/by-name/modem
adsp.*                  /dev/block/bootdevice/by-name/modem
```

这个文件定义了对应放入的文件名称和其相应的分区。

添加完后编译生成的整包中（我们上面描述的）：

out/target/product/msm8909/obj/PACKAGING/target_files_intermediates 目录下的整包原始文件中：

的 RADIO 文件夹下就会有我们添加的非 AP 部分升级文件：（这里我只添加了 modem 部分）



然后第 2 种方法，也是经过我实际测试可行的：
我们不用把非 AP 部分放入 device/qcom/msm8909 目录下
当我们编译了版本，生成了：
out/target/product/msm8909/obj/PACKAGING/target_files_intermediates/XXX.zip 这几个文件后
我们可以通过手动将 modem 等非 AP 部分文件放入：

```
root@will-OptiPlex-790:/home/sdc/law/bak/sc10/out/target/product/msm8909/obj/PACKAGI
NG/target_files_intermediates# ls
msm8909-target_files-eng.root  msm8909-target_files-eng.root.zip
```

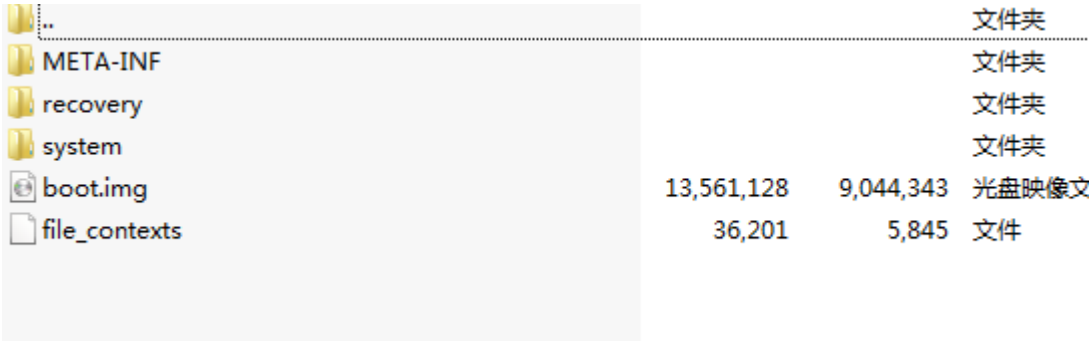
cp (modem 所在位置) \ out/target/product/msm8909/obj/PACKAGING/target_files_intermediates\msm8909-target_files-eng.root
这个文件夹的 RADIO 中，然后在压缩成 zip 格式

1.3. Full OTA 制作第三步：生成 update.zip 这个最终的 FULL 升级包

我们将这个文件拷贝到一个文件夹中：
cp msm8909-target_files-eng.root.zip ~/ msm8909-target_files-eng.root.zip(随便哪个目录)

```
root@will-OptiPlex-790:/home/sdc/law/bak/sc10/out/target/product/msm8909/obj/PACKAGING/target_files_intermediates# cp
  msm8909-target_files-eng.root.zip  ~/ msm8909-target_files-eng.root.zip
```

拷贝完后我们回到代码根目录：
如果配置好了环境可以通过 croot 来回到代码根目录：
croot
到了代码根目录下，我们可以执行：
./build/tools/releasetools/ota_from_target_files -p out/host/linux-x86 -k build/target/product/security/testkey -v ~/msm8909-target_files-eng.root.zip update.zip
其中
-k build/target/product/security/testkey 是指的对包进行签名，如果不签也可省略
-v 是显示命令
~/msm8909-target_files-eng.root.zip 是指的全包的原始文件所在的位置
Update.zip 是我们生成的文件名（当然也可以自己定义生成的包的位置，默认是在根目录下）
最后执行完后会在代码根目录下生成 update.zip 这个全包升级文件。我们可以看下全包的内容：

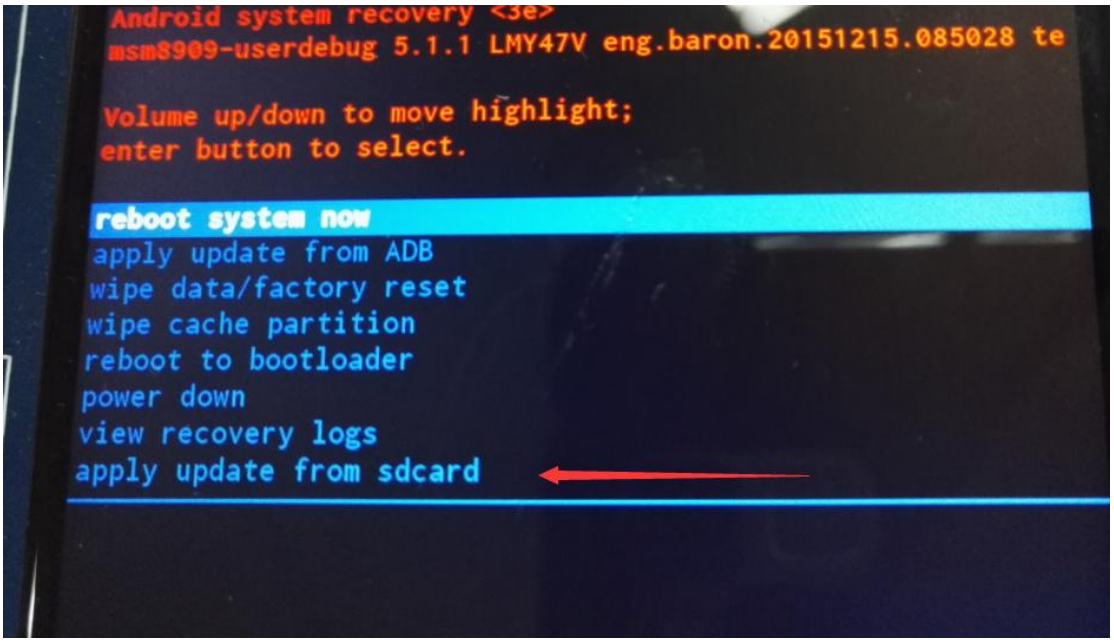


接下来我们将

1.4. Full OTA 制作第四步：复制到 SD 卡启动 RECOVERY 进行升级

这个刷机包拷贝到 sd 中。
进入 recovery，升级就可以了

SC10 通过 PWRKEY 开机+ 同时 VOL_UP 拉低 进入 Recover 界面



选择最后一项：apply update from sdcard
然后找到升级包在 sd 卡中的位置，点击就能升级了。
当然这里需要提的一点的是，对于 recovery 的升级不是通过这种方式，毕竟我们不能站在桥上去拆桥

1.5. Full OTA 强调注意：生成的 msm8909-target_files-eng.XXX.zip 要保存维护好

每次发布版本，制作 factory. Update,debug ， 同时也要保存好生成 msm8909-target_files-eng.XXX.zip

1.6. Full OTA 的 update 升级包签名保护

签名的证书，密钥，管理，系统实现，待研究后进一步补充完善。

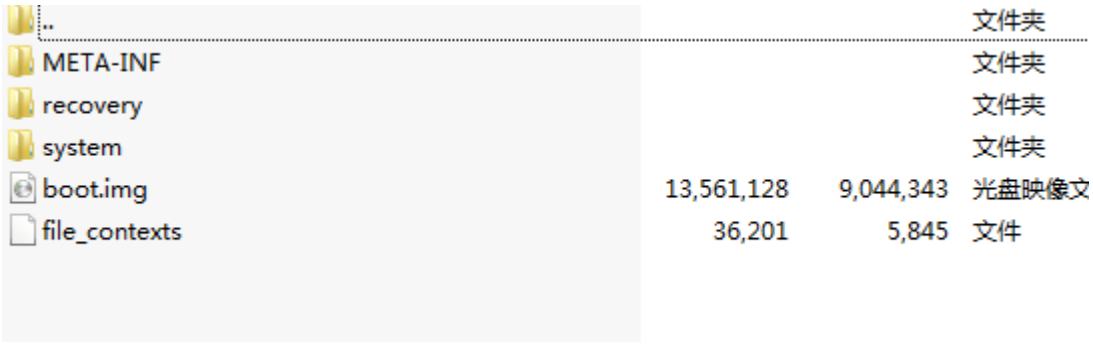
Update.zip 生成时会签名保护，如下：
./build/tools/releasetools/ota_from_target_files -p out/host/linux-x86 -k build/target/product/security/testkey -v ~/msm8909-target_files-eng.root.zip update.zip
其中
-k build/target/product/security/testkey 是指的对包进行签名，如果不签也可省略
-v 是显示命令

~/msm8909-target_files-eng.root.zip 是指的全包的原始文件所在的位置

1.7. Full OTA 升级实现原理分析

举例如下：

首先我们看下一个整包的内容：



在 META-INF 这个目录下 META-INF\com\google\android 中：

有这么两个文件，

update-binary---升级用的脚步解析器， 被 recovery 系统调用执行

updater-script---升级脚本，被 update-binary 解析执行,脚本内容如下图

```
ifelse(getprop("ro.product.name") != "cancro_wc_lte", (getprop(ro.product.name) == "cancro" || a
mount("ext4", "EMMC", "/dev/block/platform/msm_sdcc.1/by-name/userdata", "/data");
show_progress(0.500000, 0);
format("ext4", "EMMC", "/dev/block/platform/msm_sdcc.1/by-name/system", "0", "/system");
mount("ext4", "EMMC", "/dev/block/platform/msm_sdcc.1/by-name/system", "/system");
package_extract_dir("recovery", "/system") || abort("Failed to extract dir fro \"recovery\" to
package_extract_dir("system", "/system") || abort("Failed to extract dir from \"system\" to \"/s
symlink("../ui/MessageComplete.ogg", "/system/media/audio/notifications/MessageComplete.ogg");
symlink("../ui/MessageIncoming.ogg", "/system/media/audio/notifications/MessageIncoming.ogg");
symlink("/data/misc/audio/mbhc.bin", "/system/etc/firmware/wcd9320/wcd9320_mbhc.bin");
symlink("/data/misc/audio/wcd9320_anc.bin", "/system/etc/firmware/wcd9320/wcd9320_anc.bin");
symlink("/data/misc/audio/wcd9320_mad_audio.bin", "/system/etc/firmware/wcd9320/wcd9320_mad_audi
symlink("/system/etc/firmware/ath6k/AR6004/hw1.3/bdata.bin_usb", "/system/etc/firmware/ath6k/AR6
symlink("/system/etc/firmware/ath6k/AR6004/hw1.3/fw.ram.bin_usb", "/system/etc/firmware/ath6k/AR
symlink("/system/etc/firmware/ath6k/AR6004/hw3.0/bdata.bin_usb", "/system/etc/firmware/ath6k/AR6
symlink("/system/lib/modules/pronto/pronto_wlan.ko", "/system/lib/modules/wlan.ko");
symlink("Roboto-Bold.ttf", "/system/fonts/DroidSans-Bold.ttf");
symlink("Roboto-Regular.ttf", "/system/fonts/DroidSans.ttf");
symlink("libGLESv2.so", "/system/lib/libGLESv3.so");
symlink("mksh", "/system/bin/sh");
symlink("toolbox", "/system/bin/cat", "/system/bin/chcon",
        "/system/bin/chmod", "/system/bin/chown", "/system/bin/clear",
        "/system/bin/cmp", "/system/bin/cp", "/system/bin/date",
        "/system/bin/dd", "/system/bin/df", "/system/bin/dmesg",
        "/system/bin/du", "/system/bin/getenforce", "/system/bin/getevent",
```

内容很简单比如第 2 句：

mount("ext4", "EMMC", "/dev/block/platform/msm_sdcc.1/by-name/userdata", "/data");

就是挂载 userdata 分区
第 4 句：
format("ext4", "EMMC", "/dev/block/platform/msm_sdcc.1/by-name/system", "0", "/system");
擦除 system 分区。等等
updater-script 脚本中内容被 update-binary 读取解析执行，update-binary 被 recovery 系统调用。

而在全包的其他一些目录，就是我们要更新系统的资源。比如整包目录下的这些 boot.img 等：

到时候就会在脚本中就会被如下语句执行，刷入相应的分区，

```
95 package_extract_file("boot.img", "/dev/block/platform/msm_sdcc.1/by-name/boot");
96 show_progress(0.100000, 0);
97 ui_print("Writing image emmc_appsboot.mbn...");
98 package_extract_file("emmc_appsboot.mbn", "/dev/block/platform/msm_sdcc.1/by-name/aboot");
99 ui_print("Writing image tz.mbn...");
100 package_extract_file("tz.mbn", "/dev/block/platform/msm_sdcc.1/by-name/tz");
101 ui_print("Writing image NON-HLOS.bin...");
102 unmount("/vendor/firmware");
103 package_extract_file("NON-HLOS.bin", "/dev/block/platform/msm_sdcc.1/by-name/modem");
104 ui_print("Writing image DDR.img...");
105 package_extract_file("DDR.img", "/dev/block/platform/msm_sdcc.1/by-name/DDR");
106 ui_print("Writing image rpm.mbn...");
107 package_extract_file("rpm.mbn", "/dev/block/platform/msm_sdcc.1/by-name/rpm");
108 ui_print("Writing image sdi.mbn...");
109 package_extract_file("sdi.mbn", "/dev/block/platform/msm_sdcc.1/by-name/dbi");
110 ui_print("Writing image sbl1.mbn...");
111 package_extract_file("sbl1.mbn", "/dev/block/platform/msm_sdcc.1/by-name/sbl1");
112 delete("/data/miui/apps/GoogleTTS.apk");
113 delete("/data/app/GoogleTTS.apk");
```

2. SC10 Incremental OTA 方式升级介绍

Incremental OTA 升级就是针对两个版本，通过工具生成 Incremental 包，做 Incremental 升级的方案。所以注意要有版本升级对应关系。

升级对象：boot.img， emmc_appsboot.mbn, non-hlos.bin, sbl1.mbn, rz.mbn, rpm.mbn ….
不确定： cache.img, system.img, userdata.img 是以映像 Incremental 来升级的，还是以具体文件或者 android 中应用和库为差异对象来进行升级的。----------

(根据制作升级包的方式----file base 和 block base ， file base 是以安卓中具体文件差异做差分，block base 是以映像文件差异来做差异)

2.1. Incremental OTA 制作第一步：生成各版本的 msm8909-target_files-eng.XXX.zip

假设我们编译了版本 V1.
在编译完成后的 out/target/product/msm8909/obj/PACKAGING/target_files_intermediates 目录下，会自动生成这个版本的整包压缩文件：msm8909-target_files-eng.XXX.zip（可能名字会有所区别）
我们把这个文件拷贝到一个文件夹
cp msm8909-target_files-eng.XXX.zip ~/update/msm8909-target_files-eng.V1.zip

然后我们修改编译了版本 V2 这个版本相对上个版本做了一些修改
同样在 out/target/product/msm8909/obj/PACKAGING/target_files_intermediates 这个目录下有同样的一个文件，我们拷贝为 V2, cp msm8909-target_files-eng.XXX.zip ~/update/msm8909-target_files-eng.V2.zip

2.2. Incremental OTA 制作第二步：Modem 等非 HLOS 加入升级包的方法

当然如果我们要加入制作非 AP 部分也就是 modem 等部分的差分包内容，这点和我们在第一节中说明的方法是一样的：
当我们编译了版本，生成了：
out/target/product/msm8909/obj/PACKAGING/target_files_intermediates/XXX.zip 这几个文件后
我们可以通过手动将 modem 等非 AP 部分文件放入：

```
root@will-OptiPlex-790:/home/sdc/law/bak/sc10/out/target/product/msm8909/obj/PACKAGI
NG/target_files_intermediates# ls
msm8909-target_files-eng.root  msm8909-target_files-eng.root.zip
```

cp (modem 所在位置) \ out/target/product/msm8909/obj/PACKAGING/target_files_intermediates\msm8909-target_files-eng.root
这个文件夹的 RADIO 中，然后在压缩成 zip 格式

2.3. Incremental OTA 制作第三步：生成 update.zip 这个最终的 Incremental 升级包

接下来我们来到安卓源码根目录：

```
abi         build.sh      device       kernel            out         sdk
art         cts           docs         libcore           out.bak     system
bionic      dalvik        external     libnativehelper   packages    tools
bootable    developers    frameworks   Makefile          pdk         vendor
build       development   hardware     ndk               prebuilts
law@will-OptiPlex-790:~/law/sc10$
```

执行

./build/tools/releasetools/ota_from_target_files -v --block -p out/host/linux-x86 -i \
~/update/msm8909-target_files-eng.V1zip \
~/update/msm8909-target_files-eng.V2.zip update.zip

需要注意的是我们必须在安卓编译源码目录下执行这个文件，
-v 显示具体命令，-i 为产生增量包。--block 为以 block-based OTA 的方式产生，不填则默认 file-based OTA 的方式产生升级包，5.0 后使用 block-based OTA 方式为好，两者区别，下面会说明

最后会在安卓源码根目录下产生 update.zip 文件：

```
abi         build.sh      device       kernel            out         sdk
art         cts           docs         libcore           out.bak     system
bionic      dalvik        external     libnativehelper   packages    tools
bootable    developers    frameworks   Makefile          pdk         update.zip
build       development   hardware     ndk               prebuilts   vendor
```

我们可以看下这个包的大小

```
root@will-OptiPlex-790:/home/sdc/law/bak/sc10# du -h update.zip
20M     update.zip
```

只有 20M 的大小，其包含了整个 AP 部分的差分信息，相对于上面的卡刷升级包，已经小了很多，当然这个产生的包也能通过卡刷升级。因为通过我们第一节分析了解这其中的原理都是一样的，下图是包的内容：

..			文件夹
META-INF			文件夹
patch			文件夹
system.new.dat	1,699,840	1,042,866	DAT 文件
system.patch.dat	15,041,753	15,041,753	DAT 文件
system.transfer.list	304,878	147,331	LIST 文件

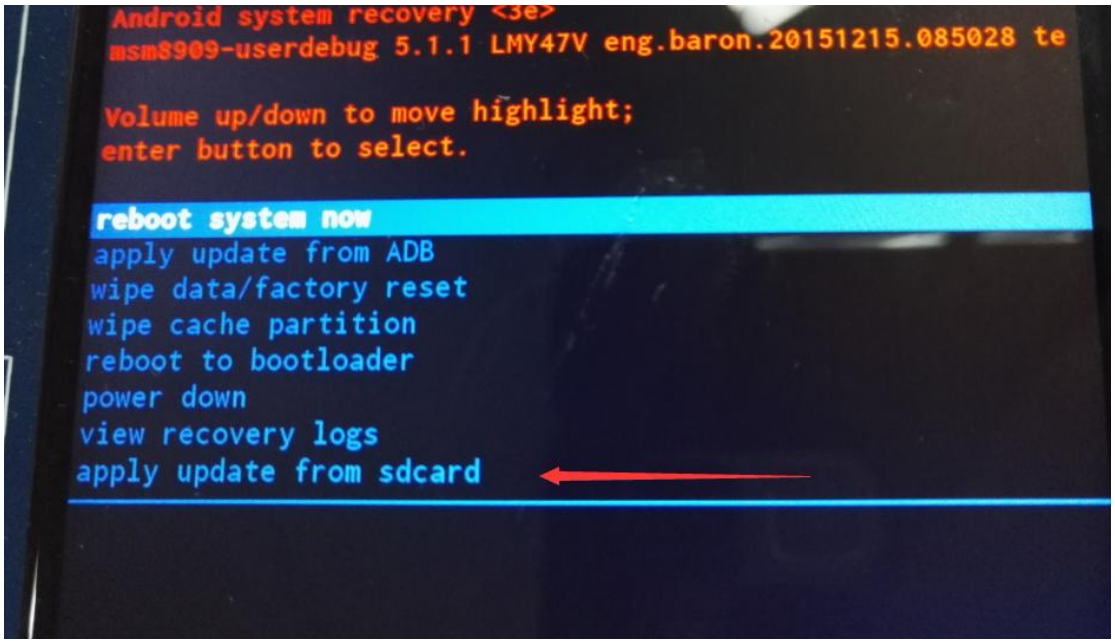
在 patch 这个目录下：

..			文件夹
boot.img.p	7,060,650	7,060,431	P 文件

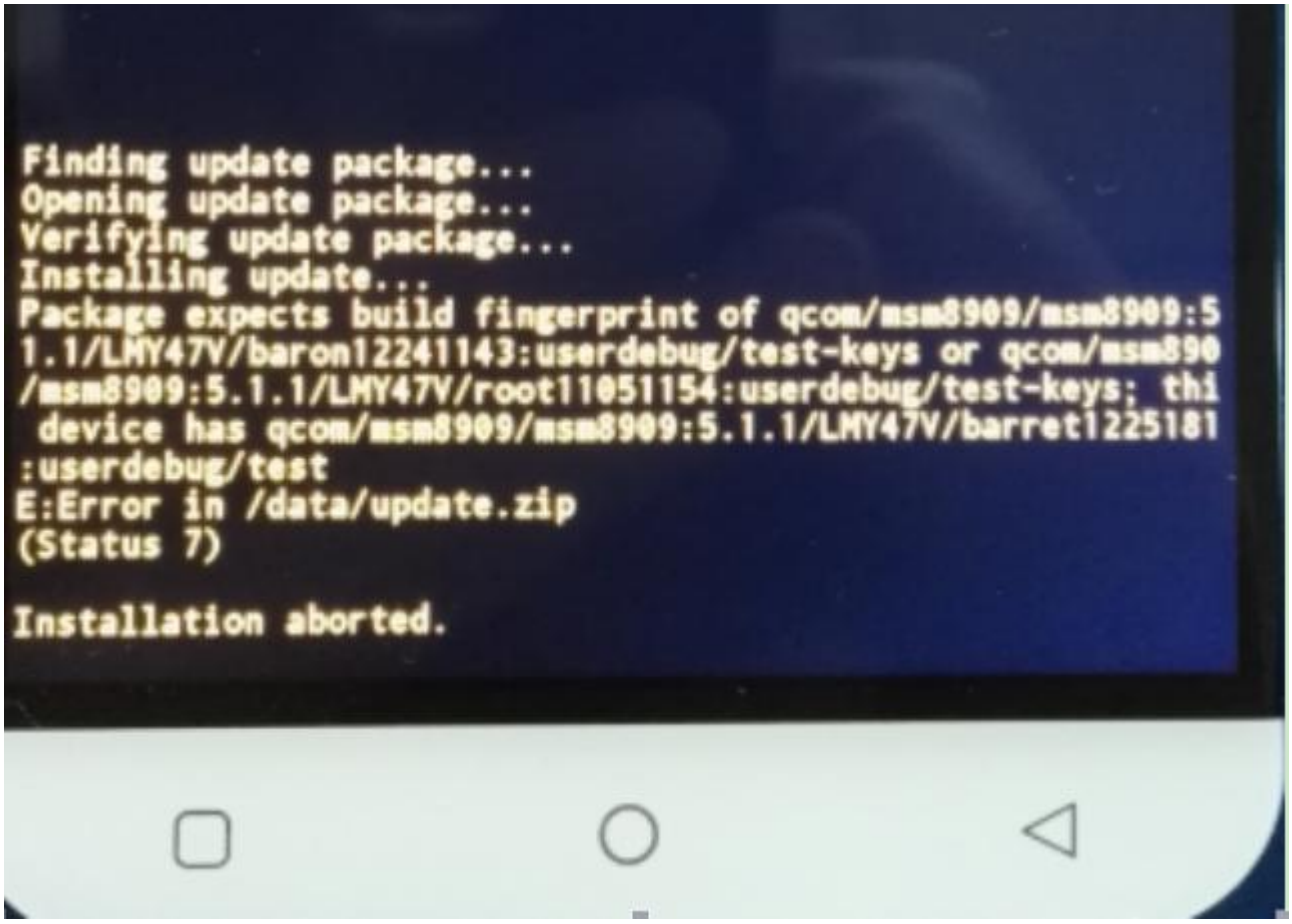
可以发现以 block-based 的方式产生的升级包，其内容很少

2.4. Incremental OTA 制作第四步：复制到 SD 卡启动 RECOVERY 进行升级

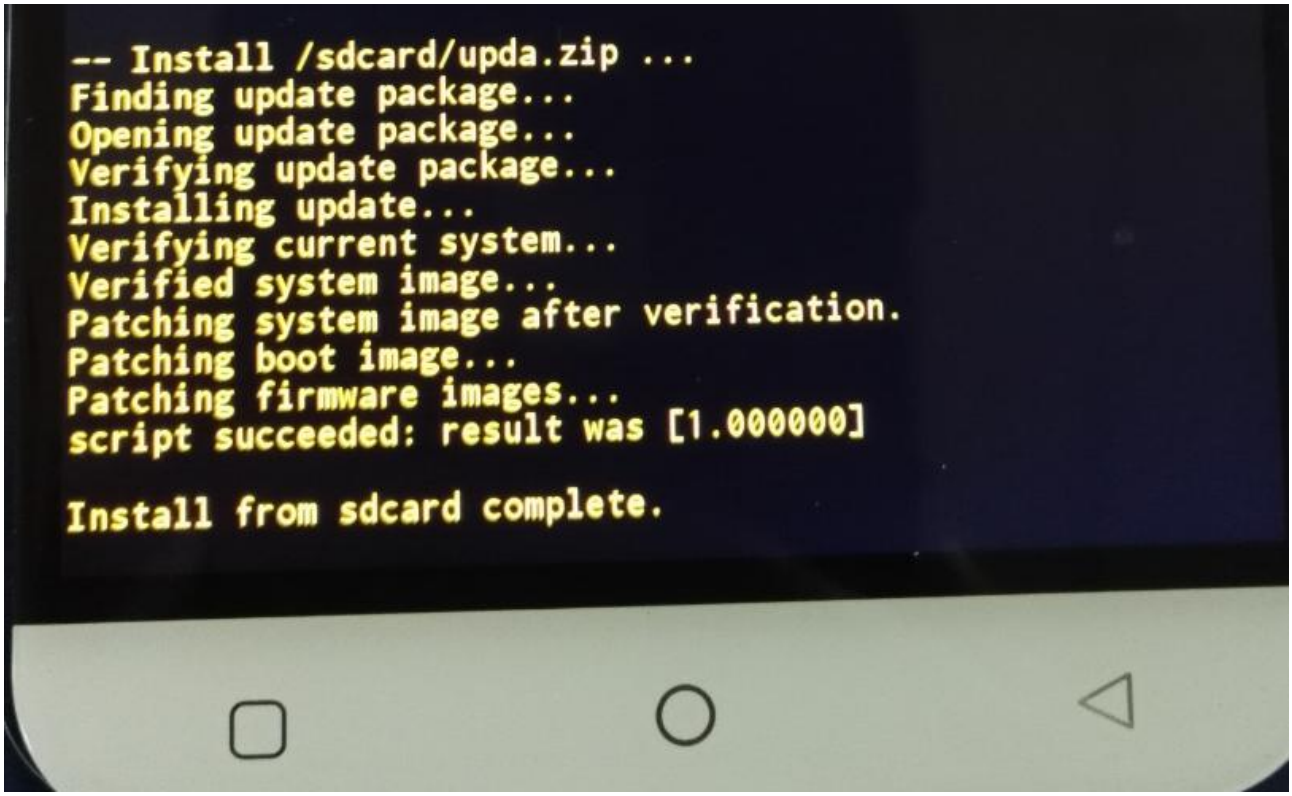
当制作完 v1 版本和 v2 版本的差分包后。
我们的目标板上现在运行的是 V1 版本的软件（这个是必须的）
然后 SC10 通过 PWRKEY 开机+ 同时 VOL_UP 拉低 进入 Recover 界面



选择最后一项：apply update from sdcard
然后找到升级包在 sd 卡中的位置，点击就能升级了。
如果目标板子上运行的不是 V1 版本的软件，那么这个 V1 和 V2 的差分包在升级的时候就会出错如下图：



成功后的显示大致是这样的：



2.5. Incremental OTA 强调注意：各个版本的 msm8909-target_files-eng.XXX.zip 要保存维护好

每次发布版本，制作 factory. Update,debug ， 同时也要保存好生成 msm8909-target_files-eng.XXX.zip

2.6. Incremental OTA 的 Update.zip 升级包签名保护

在我们制作升级包的时候，可以 用 -k build/target/product/security/testkey 是指的对包进行签名
这里可以了解下详细的分析流程，根据高通的说明文档我们可以明确：
build/target/product/security/testkey 就是升级包的签名文件，那么如何改变这个签名？
build/target/product/security/README 中说明了详细的方法

```
root@will-OptiPlex-790:/home/sdc/law/bak/sc10/build/target/product/security# ls
Android.mk      platform.pk8       shared.pk8       testkey.pk8       verity.pk8
media.pk8       platform.x509.pem  shared.x509.pem  testkey.x509.pem  verity.x509.pem
media.x509.pem  README             testkey.pem      verity_key
```

development/tools/make_key \
testkey '/C=US/ST=California/L=Mountain View/O=Android/OU=Android/CN=Andro \
id/emailAddress=android@android.com'
这个是签名文件的生成方法。另外这个说明文件中也说明了对其他文件，如 system.img 的签名方法：

```
signing using the openssl commandline (for boot/system images)
---------------------------------------------------------------

1. convert pk8 format key to pem format
   % openssl pkcs8 -inform DER -nocrypt -in testkey.pk8 -out testkey.pem

2. create a signature using the pem format key
   % openssl dgst -binary -sha1 -sign testkey.pem FILE > FILE.sig

extracting public keys for embedding
```

这里先不分析对 system.img 的签名，来看对升级包的签名。

development/tools/make_key \
testkey '/C=US/ST=California/L=Mountain View/O=Android/OU=Android/CN=Andro \
id/emailAddress=android@android.com'

首先看下 development/tools/make_key 这个文件

```
40 tmpdir=$(mktemp -d)
41 trap 'rm -rf ${tmpdir}; echo; exit 1' EXIT INT QUIT
42
43 one=${tmpdir}/one
44 two=${tmpdir}/two
45 mknod ${one} p
46 mknod ${two} p
47 chmod 0600 ${one} ${two}
48
49 read -p "Enter password for '$1' (blank for none; password will be visible): " \
50   password
51
52 if [ "${3}" = "rsa" -o "$#" -eq 2 ]; then
53   ( openssl genrsa -f4 2048 | tee ${one} > ${two} ) &
54   hash="-sha1"
55 elif [ "${3}" = "ec" ]; then
56   ( openssl ecparam -name prime256v1 -genkey -noout | tee ${one} > ${two} ) &
57   hash="-sha256"
58 else
59   echo "Only accepts RSA or EC keytypes."
60   exit 1
61 fi
62
63 openssl req -new -x509 ${hash} -key ${two} -out $1.x509.pem \
64   -days 10000 -subj "$2" &
65
66 if [ "${password}" == "" ]; then
67   echo "creating ${1}.pk8 with no password"
68   openssl pkcs8 -in ${one} -topk8 -outform DER -out $1.pk8 -nocrypt
69 else
70   echo "creating ${1}.pk8 with password [${password}]"
71   export password
72   openssl pkcs8 -in ${one} -topk8 -outform DER -out $1.pk8 \
73     -passout env:password
74   unset password
75 fi
```

这是一个 shell 脚本文件。

看下第 52 行：

```
52 if [ "${3}" = "rsa" -o "$#" -eq 2 ]; then
53   ( openssl genrsa -f4 2048 | tee ${one} > ${two} ) &
54   hash="-sha1"
```

根据上面的分析，可以明确这里走的是这条分支，

openssl genrsa -f4 2048 | tee ${one} > ${two}

这句的意思就是产生秘钥分别保存到${one} 和${two} 这两个缓存文件，这两个缓存文件存放在：/tmp/tmp.PfcSLY5GMe 这个文件夹中。

生成秘钥后接下来看第 63 行：

```
63 openssl req -new -x509 ${hash} -key ${two} -out $1.x509.pem \
64   -days 10000 -subj "$2" &
```

来看下这些参数：

其中 ${two}表示前面生成的秘钥文件，${hash}表示的是 -sha1 选项

-new：本选项产生一个新的 CSR，它会要用户输入创建 CSR 的一些必须的信息。至于需要哪些信息，是在 config 文件里面定义好了的。如果-key 没有被设置,，那么就将根据 config 文件里的信息先产生一对新

的 RSA 密钥值。

-x509：本选项将产生自签名的证书。

-key filename：证书私钥文件的来源。

-days n：指定自签名证书的有效期限。默认为 30 天。

-subj arg：用于指定生成的证书请求的用户信息，或者处理证书请求时用指定参数替换。生成证书请求时，如果不指定此选项，程序会提示用户来输入各个用户信息，包括国名、组织等信息，如果采用此选择，则不需要用户输入了。比如：-subj /CN=china/OU=test/O=abc/CN=forxy，注意这里等属性必须大写。

这个语句的作用就是产生一个自签名的证书。

接下来看：

```
66 if [ "${password}" == "" ]; then
67   echo "creating ${1}.pk8 with no password"
68   openssl pkcs8 -in ${one} -topk8 -outform DER -out $1.pk8 -nocrypt
69 else
70   echo "creating ${1}.pk8 with password [${password}]"
71   export password
72   openssl pkcs8 -in ${one} -topk8 -outform DER -out $1.pk8 \
73     -passout env:password
74   unset password
75 fi
```

根据命令行有无密码输入会走不同的分析，看下参数的含义，

-in filename：输入的密钥文件，默认为标准输入。如果密钥被加密，会提示输入一个密钥口令。

-topk8：通常的是输入一个 pkcs8 文件和传统的格式私钥文件将会被写出。设置了此选项后，位置转换过来：输入一个传统格式的私钥文件，输出一个 PKCS#8 格式的文件。

-outform DER|PEM：输出文件格式，DER 或者 PEM 格式。

-out filename：输出文件，默认为标准输出。如果任何加密操作已经执行，会提示输入一个密钥值。输出的文件名字不能和输入的文件名一样。

-passout arg：输出文件口令保护来源。

这里将秘钥文件转换为 pkcs8 文件。

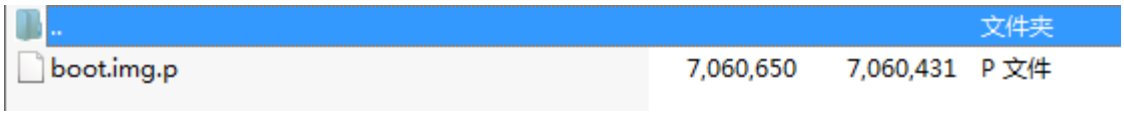
根据分析，明确我们可以产生新的签名文件来对升级包进行签名

2.7. Incremental OTA 升级实现原理分析

当差分包制作完成后，其目录结构大致是这样的：

..			文件夹
META-INF			文件夹
patch			文件夹
system.new.dat	1,699,840	1,042,866	DAT 文件
system.patch.dat	15,041,753	15,041,753	DAT 文件
system.transfer.list	304,878	147,331	LIST 文件

在 patch 目录中，包含的是对上一版本不同处的差分 patch：



虽然是做的差分，但是其大小也不是很小，差不多有 7M 左右，相对于一些专业的差分软件，还是有些差距。

以这个 boot.img.p 为例，其到时候会被差分包的：

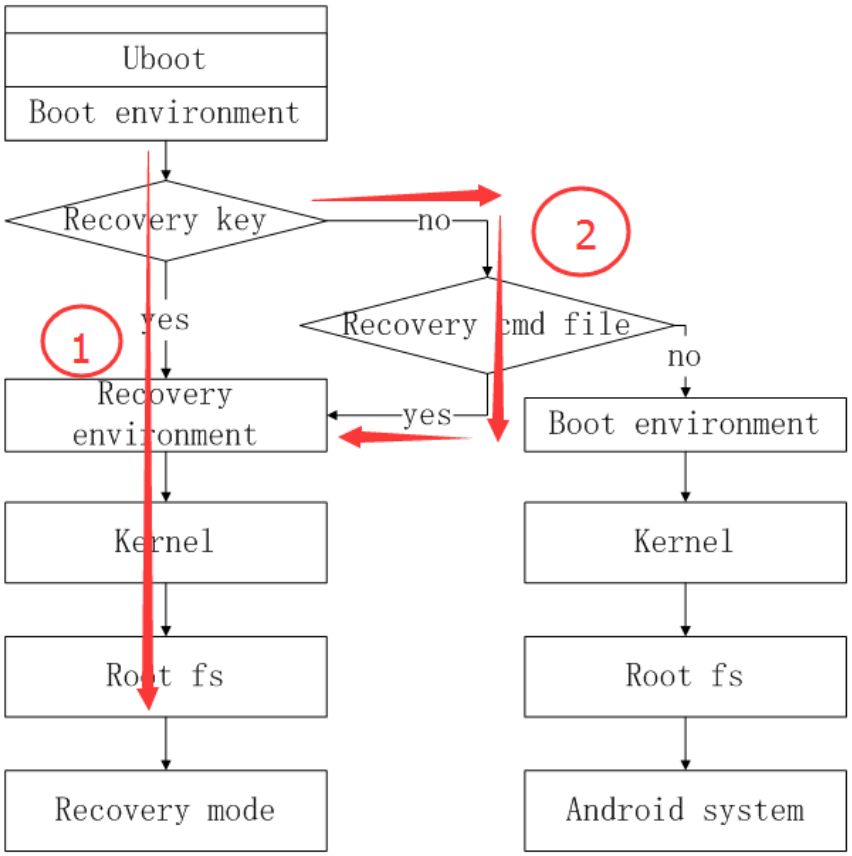
META-INF/com/google/android/updater-script 这个脚本解析，并执行，如下图

```
get_device_compatible("msm8909") == "OK" || abort("This package is for \"msm8909\" devices; this is a \"" + getpro
ui_print("Verifying current system...");
getprop("ro.build.fingerprint") == "qcom/msm8909/msm8909:5.1.1/LMY47V/root12292026:userdebug/test-keys" ||
    getprop("ro.build.fingerprint") == "qcom/msm8909/msm8909:5.1.1/LMY47V/barret12251818:userdebug/test-keys" ||
    abort("Package expects build fingerprint of qcom/msm8909/msm8909:5.1.1/LMY47V/root12292026:userdebug/test-keys
apply_patch_check("EMMC:/dev/block/bootdevice/by-name/boot:13561128:c7b9cd770e0e32d3236ad8bb256a23013fff064c:13563
if block_image_verify("/dev/block/bootdevice/by-name/system", package_extract_file("system.transfer.list"), "syste
ui_print("Verified system image...");
else
```

3. SC10 OTA 无线升级下载方案介绍

3.1. SD 卡升级和无线升级 RECOVER 流程图

在这里找了一幅网上的图来说明



图中 1 就是卡刷进入的流程，直接通过组合按键进入 recovery

图 2 就是无线升级进入的方式，首先会有一个 apk，来检查更新，至于 apk 部分检测更新的流程这里就不说明，当 apk 检测到系统更新了后，会下载差分包，下载完后就会
加入/cache/recovery/command 这个文件，然后重启机子，接下来的流程就如上图 2 中的流程。系统开机检测到有/cache/recovery/command 这个文件，然后就会进入 recovery 执行环境，接下来 command 中包含的命令就会执行第一节中分析的内容，这里就是升级流程的大致框架。接下来我会详细说明。

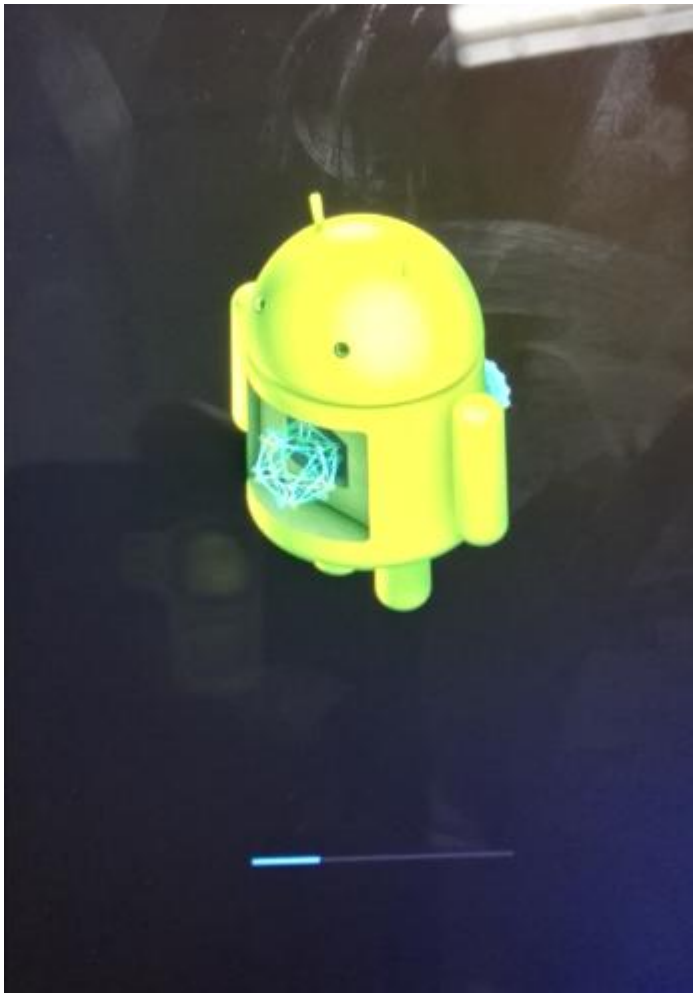
3.2. Android APK 设置升级标志进入 RECOVERY 升级系统

安卓提供了接口给我们来进行升级。
首先通过

adb root
adb push update.zip /data/update.zip
将升级包导入手机，这个步骤，到时候我们就可以直接通过 apk 下载升级包到/data/ 目录或 /cache 目录或 SD 下。
接下来我们在 cache 目录下创建 recovery 文件夹：
adb shell "mkdir /cache/recovery"
然后在 recovery 目录下创建文件 command
adb shell "touch /cache/recovery/command"
下一步就是往 command 这个文件中写入：
adb shell
echo "--update_package=/data/update.zip" > /cache/recovery/command

最后我们重启到 recovery，系统就会自动进行更新了：
adb reboot recovery

这几步都可以通过 apk 来执行，执行完后，其界面就是这样的，recovery 会根据传入 command 中 update.zip 的位置，来自动进行更新



3.3. SC10 的 OTA 空中下载要自己实施进入 RECOVERY 升级系统

对于 OTA，高通并没有提供相应的方式，只是提供了如上章节“Android APK 设置升级标志进入 RECOVERY 升级系统”的上升级接口，如果我们要实施 OTA 的升级，我们需要自行完成

4. innopath 公司的 Diff package 升级方案分析

参考文档 80-N5576-89_APPLICATION NOTE_ INNOPATH FOTA SOLUTION FOR AMSS_HLOS ON MDM9X15 PLATFORM.pdf

4.1. Incremental OTA 和 Innopath Diff package 技术对比分析

待续.....

5. OTA 升级中一些问题的补充

5.1. rpm/aboot 等升级过程断电分析和处理方案

参考文档
KBA-000000028438_What are sbl1bak-tzbak-rpmbak-abootbak partitions used for.pdf

Background:
On most B-family chipset Android build partition table you can find additional backup partitions:
For example:

<partition label="sbl1" size_in_kb="512" type="DEA0BA2C-CBDD-4805-B4F9-F428251C3E98" bootable="false" readonly="false" filename="sbl1.mbn"/>
<partition label="sbl1bak" size_in_kb="512" type="DEA0BA2C-CBDD-4805-B4F9-F428251C3E98" bootable="false" readonly="false" filename="sbl1.mbn"/>
<partition label="aboot" size_in_kb="1024" type="400FFDCD-22E0-47E7-9A23-F16ED9382388" bootable="false" readonly="false" filename="emmc_appsboot.mbn"/>
<partition label="abootbak" size_in_kb="1024" type="400FFDCD-22E0-47E7-9A23-F16ED9382388" bootable="false" readonly="false" filename="emmc_appsboot.mbn"/>
<partition label="rpm" size_in_kb="500" type="098DF793-D712-413D-9D4E-89D711772228" bootable="false" readonly="false" filename="rpm.mbn"/>
<partition label="rpmbak" size_in_kb="500" type="098DF793-D712-413D-9D4E-89D711772228" bootable="false" readonly="false" filename="rpm.mbn"/>
<partition label="tz" size_in_kb="500" type="A053AA7F-40B8-4B1C-BA08-2F68AC71A4F4" bootable="false" readonly="false" filename="tz.mbn"/>
<partition label="tzbak" size_in_kb="500" type="A053AA7F-40B8-4B1C-BA08-2F68AC71A4F4" bootable="false" readonly="false" filename="tz.mbn"/>

Question:
What are these partition used for?
Should we keep them?
Why other non-hlos partitions like modem/sdi don't have such backup partitions?

Answer:
1. These partitions are not touched during normal boot, they are only used during Android FOTA upgrade.
2. Recovery image is in recovery partition and is loaded by LK when doing a FOTA upgrade by checking cookies in misc parition.
3. The recovery executable will extract all HLOS and NON-HLOS images from the upgrade package and start upgrading the corresponding partitions.
4. For boot-critical images (SBL1, TZ, RPM, ABOOT), an additional precaution is taken to ensure that if a crash occurs while these images are being updated, we can return to recovery and try again. Before updating these partitions, the GPT table is modified to use the backup partitions on a subsequent boot. Thus, if a crash occurs, it is possible to reboot back into recovery using the backup partitions, since they are guaranteed always to be intact.
5. When updates to the critical partitions are done, the GPT table is restored to its earlier state and the backup partitions are updated.
6. Related code can be found in Android code device/qcom/common/recovery/oem-recovery/gpt-utils.c
The mechanism is to swap the GUID of the main partition entry and the bak partition entry, like swap the GUID of sbl1 and sbl1bak paritition.
So if crash happens during updating, on next boot process, the bak partition will be loaded.
7. So, in order to get it work, must make sure that the main partition and bak partition have different GUIDs.
8. Only boot critical partitions are needed to do such backup, so SBL/RPM/TZ/Aboot are needed.

对于 sc10，其在断电过程中的包护：
For boot-critical images (SBL1, TZ, RPM), an additional precaution is taken to ensure that if a crash occurs while these images are being updated, we can return to recovery and try again. Before updating these partitions, the GPT table is modified to use the backup partitions on a subsequent boot. Thus, if a crash occurs, it is possible to reboot back into recovery using the backup partitions, since they are guaranteed always to be intact.
When updates to the critical partitions are done, the GPT table is restored to its earlier state and the backup partitions are updated.
简单说明下，就是在升级的时候会备份一些关键分区，如（SBL1, TZ, RPM）。以防系统更新出错时造成系统无法开启的问题

5.2. Block-based OTA 和 file-base OTA 的区别

File vs. Block OTAs
During a file-based OTA, Android attempts to change the contents of the system partition at the filesystem layer (on a file-by-file basis). The update is not guaranteed to write files in a consistent order, have a consistent last modified time or superblock, or even place the blocks in the same location on the block device. For this reason, file-based OTAs fail on a dm-verity-enabled device; after the OTA attempt, the device does not boot.

During a block-based OTA, Android serves the device the difference between the two block images (rather than two sets of files). The update checks a device build against the corresponding build server at the block level (below the filesystem) using one of the following methods:

Full update. Copying the full system image is simple and makes patch generation easy but also generates large images that can make applying patches expensive.
Incremental update. Using a binary differ tool generates smaller images and makes patch application easy, but is memory-intensive when generating the patch itself.
Note: adb fastboot places the exact same bits on the device as a full OTA, so flashing is compatible with block

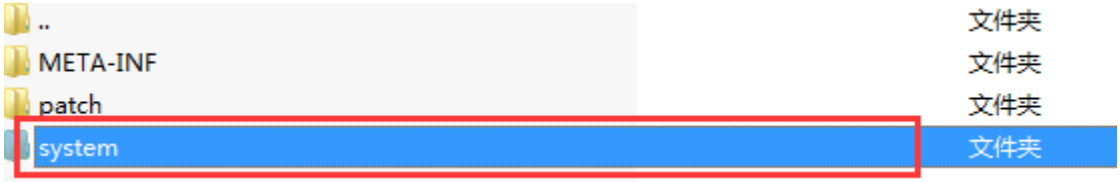
OTA.
简单来说， file-based OTA 是根据具体的文件差异来做差分，如 system 分区中有一个文件 A 做了修改，那么在这个升级包中会单独对 A 文件做差分 patch
而 block-based OTA,则是根据具体的映像差异来做差分包

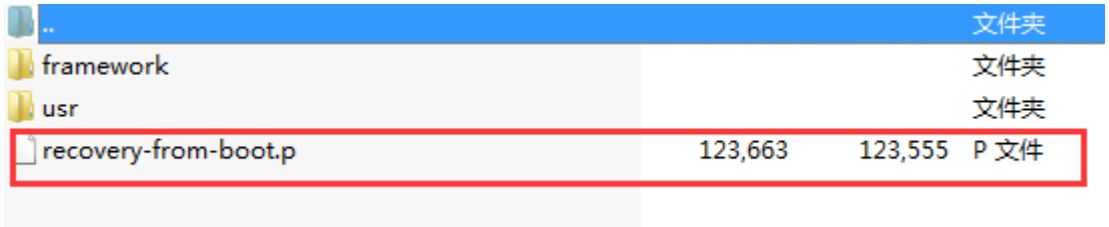
5.3. Full oTA/ Incremental OTA 升级对已经客户安装的 APK 影响

客户安装的 APK 不是存放在 system 这个分区中，其会安装在 userdata 这个分区中，而我们使用升级包进行升级的时候，userdata 这块分区是不会进行升级的，所以升级包升级不会对客户信息造成影响。当然我们可以在升级时选择清除这个分区，来格式化用户数据

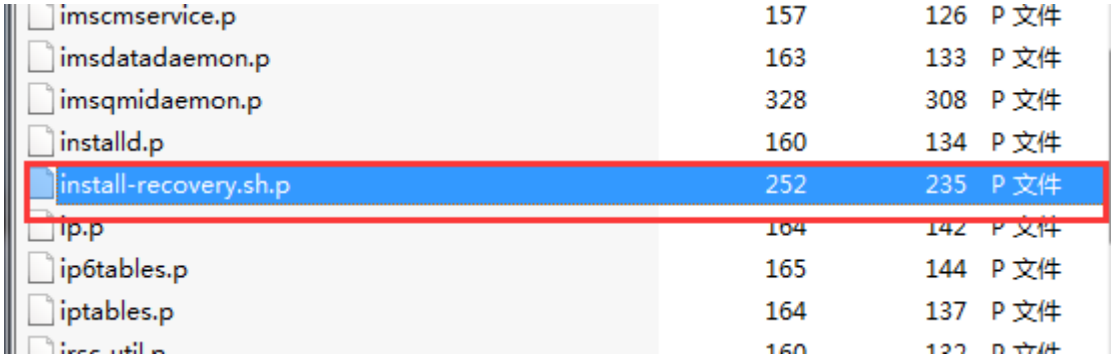
5.4. OTA 方案中 RECOVER 及其 rootfs 升级需求处理

关于 recovery 的更新
当制作完升级包，以 file-base OTA 为例：
在升级包的 system 目录下





会有 recovery-from-boot.p 这个文件
然后在升级包的 patch 文件夹下会有/patch/system/bin/install-recovery.sh



这个文件，由于这个是一个补丁文件我们没办法看，我们可以看打好 patch 后，或者系统原来就有的这个文件。

```
root@msm8909:/system/bin # cat install-recovery.sh
cat install-recovery.sh
#!/system/bin/sh
if ! applypatch -c EMMC:/dev/block/bootdevice/by-name/recovery:14118188:6c4
  applypatch -b /system/etc/recovery-resource.dat EMMC:/dev/block/bootdevic
else
  log -t recovery "Recovery image already installed"
fi
```

#!/system/bin/sh
if ! applypatch -c EMMC:/dev/block/bootdevice/by-name/recovery:14118188:6c4d552992b8bce944cb8104adf6dcff0afe5ea8; then
  applypatch -b /system/etc/recovery-resource.dat EMMC:/dev/block/bootdevice/by-name/boot:13561128:6d03e34bcd3ecfbb7f47b6ca3928dfe57e876fa5 \
   EMMC:/dev/block/bootdevice/by-name/recovery 6c4d552992b8bce944cb8104adf6dcff0afe5ea8 14118188 6d03e34bcd3ecfbb7f47b6ca3928dfe57e876fa5:/system/recovery-from-boot.p && log -t recovery "Installing new recovery image: succeeded" || log -t recovery "Installing new recovery image: failed"
else
  log -t recovery "Recovery image already installed"
fi

每次系统从安卓端进入 recovery 的时候，都会预先运行这个脚本，如果 recovery 有改变，然后会更加 recovery-from-boot.p 刷入新的 recovery

5.5. OTA 方案中 SecureBoot 签名对升级的影响