OTA 升级指导文档

Document Number:		**Document Version:**	1.0
Owner:	Spreadtrum	**Date:**	2016.5.30
Document Type:			
NOTE:	ALL MATERIALS INCLUDED HEREIN ARE COPYRIGHTED AND CONFIDENTIAL UNLESS OTHERWISE INDICATED. The information is intended only for the person or entity to which it is addressed and may contain confidential and/or privileged material. Any review, retransmission, dissemination, or other use of or taking of any action in reliance upon this information by persons or entities other than the intended recipient is prohibited. This document is subject to change without notice. Please verify that your company has the most recent specification. Copyright © 2013 Spreadtrum Communications Inc.		

SPREADTRUM®

www.spreadtrum.com

Table of Contents

1. 概述 3
1.1. Request & Purpose 3
1.2. Definitions & Abbreviations 3
2. 简介 3
2.1. OTA 简介 3
2.2. OTA 升级 6
2.2.1. Recovery 升级 6
2.2.2. 几个关键部分 6
2.2.3. 展讯特有分区以及其升级 9
3. Recovery 系统 9
3.1. 启动及运行流程 11
3.1.1. 启动流程 11
3.1.2. 运行流程 12
4. 升级包制作与升级 13
4.1. 制作原理 13
4.1.1. Target 文件说明 13
4.1.2. 升级包简单说明 13
4.2. 制作方法 14
4.2.1. 整体升级包步骤 14
4.2.2. 差分升级包步骤 15
4.3. Modem bins 改名方法 16
5. 展讯平台部分细节说明 19
5.1. nvmerge 和 splmerge 19
5.2. nvmerge 具体说明 19
5.3. Secure boot 支持 19
5.4. 关于 adb 的支持 20
6. OTA 升级测试说明 20
7. 4.4/5.0&5.1/6.0 之间分区变更的大版本升级 21
7.1. 参数使用说明 21
7.2. 升级方法 21
7.3. 大版本升级注意事项 21
8. Recovery log 抓取办法 22

1. 概述

1.1. Request & Purpose

本文重点描述以下几个方面

- 什么是 OTA 升级
- 什么是 Recovery 升级
- Recovery 升级的原理
- Reocvery 升级的流程
- 升级包制作步骤
- 展讯特有分区的升级策略
- Recovery log 的获取

1.2. Definitions & Abbreviations

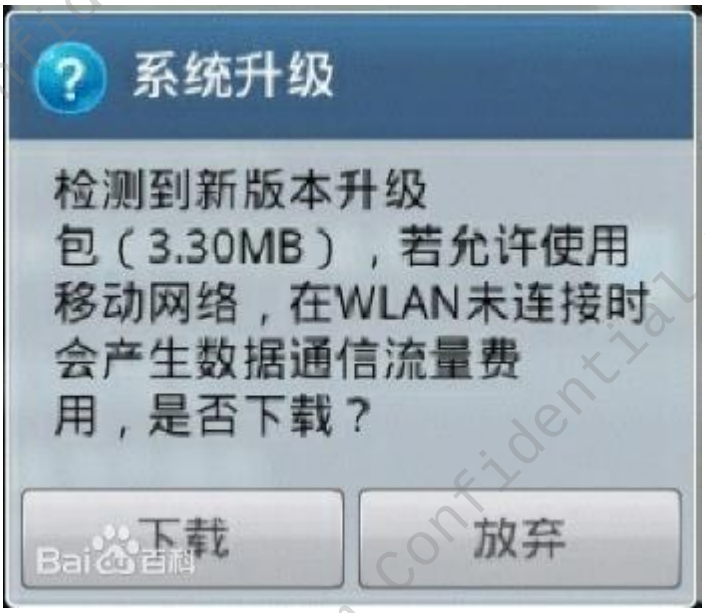
OTA	Over-The-Air Technology
BP	Baseband Processor

2. 简介

2.1. OTA 简介

OTA(Over-The-Air)一项基于短消息机制，通过手机终端或服务器（网上）方式实现 SIM 卡内业务菜单的动态下载、删除与更新，使用户获取个性化信息服务的数据增值业务（简称 OTA 业务），是通过移动通信（GSM 或 CDMA）的空中接口对 SIM 卡数据及应用进行远程管理的技术。空中接口可以采用 WAP、 GPRS、CDMA1X 及短消息技术。OTA 技术的应用，使得移动通信不仅可以提供语音和数据服务，而且还能提供新业务下载。

较早的系统升级界面。



较新的智能手机的系统升级界面（此例为联想手机图片，请勿转发）：

2.2. OTA 升级

OTA 升级是 Android 系统提供的标准软件升级方式。它功能强大，可以无损失升级系统，主要通过网络（例如 WIFI、3G/4G）自动下载 OTA 升级包、自动升级，但是也支持通过下载 OTA 升级包到 SD 卡升级。

OTA 升级包非常的小，一般几兆到十几兆，如果你用网络升级，非常的方便，基本是在系统上点击几下就完成了升级，并且重要的是，OTA 升级无需备份数据，短短几钟搞定所有升级工作，所有数据都会完好无损的保留下来。

广义上来讲，OTA 升级是一个复杂的系统。包括编译、版本发布、终端版本检测、在终端上检测服务器上最新版本、通过终端版本制作对应差分升级包、终端下载服务器上的升级包、本地升级等一系列过程。

而平时通常所说 OTA 升级仅涉及到本地升级，即把升级包下载到本地之后，进行升级的过程。而升级包放置到服务器和手机的“系统升级”app 检测到升级包并下载到本地的过程主要由售后服务部门以及 app 来完成，不在本文档介绍之列。由于升级这个过程是通过 Recovery 系统进行的，因此又称之为 Recovery 升级。本文接下来对 Recovery 升级展开描述。

2.2.1. Recovery 升级

Recovery 升级会涉及到两部分内容，Recovery 系统和升级包。这两个部分是相对独立的。在升级的过程中，Recovery 系统仅提供基本的功能（比如挂载需要用到的分区到特定目录、输出 log 到 log 文件），所有与升级有关的操作均由升级包中的 update-binary（升级程序）来驱动完成。

2.2.2. 几个关键部分

在源码中与 Recovery 升级有关的关键部分有这么几个：init.rc、recovery.fstab、recovery 主程序、updater 主程序、制作升级包的脚本。下面简单介绍一下。

1) Init.rc
   Recovery.img 中的 init.rc 仅包含了最简单的初始化和少数几个服务，包括 adb、recovery 等。
2) recovery.fstab
   用于 recovery 系统处理各个分区。包括格式化、升级都会用到。
3) recovery 主程序
   在 recovery 系统启动后 init 会将其作为服务启动。
4) updater 主程序
   即上文所述的升级程序 update-binary，用于真正的升级操作。在制作升级包时，会将其放到升级包内执行升级脚本，完成升级操作。

5) 制作升级包的脚本

主要用于生成升级包中由 update-binary 来解释执行的升级脚本 updater-script, 同时将需要用的文件打包为升级包。

Init.rc

在编译系统中，展讯 4.4 和 5.1 代码中，在芯片配置文件中定义 TARGET_RECOVERY_INITRC 为芯片目录下 recovery/init.rc，如 scx35 的定义：devicesprd/scx35/BoardConfigCommon.mk:TARGET_RECOVERY_INITRC := device/sprd/scx35/recovery/init.rc。

编译后，这个文件存放在：out/target/product/$(TARGET_DEVICE)/recovery/root/init.rc，最终编译到 recovery.img

recovery.fstab

正常 Android 系统的 fstab 用于系统启动时 system、data 等分区的挂载以及存储设备的挂载、卸载、格式化等操作，而 Recovery 系统中的 fstab 用于对所有可能需要升级的分区进行操作。这两种 fstab 的格式是一致的，但是包含的分区不同，正常 Android 系统的 fstab 仅包含 system、data、sdcard 等分区，Recovery 系统的 fstab 需要包含所有可能需要升级的分区。

在编译系统中，用变量 TARGET_RECOVERY_FSTAB 来定义 recovery.img 中的 recovery.fstab，如未定义默认使用$(TARGET_DEVICE_DIR)/recovery.fstab，即产品 makefie 所在目录下的文件 recovery.fstab。

展讯 4.4 按芯片和存储方案定义了变量 TARGET_RECOVERY_FSTAB，具体如下：

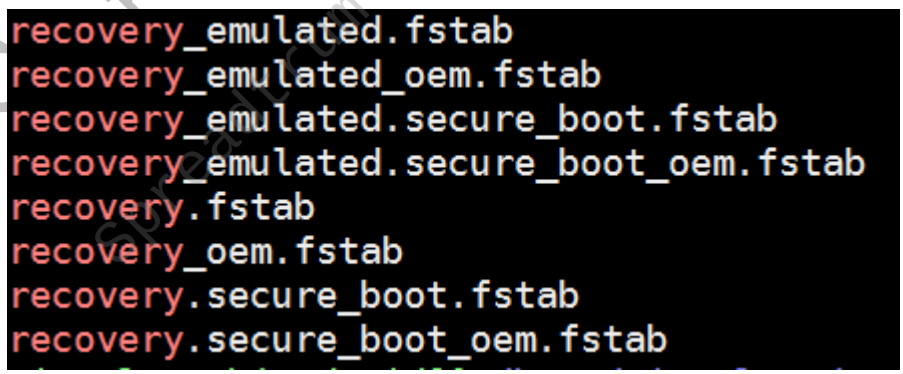
scx15 系列 nand 方案：device/sprd/scx15/nand/recovery.fstab

scx15 系列 emmc 方案：device/sprd/scx15/emmc/recovery.fstab

scx35 系列 nand 方案：device/sprd/scx35/nand/recovery.fstab

scx35 系列 emmc 方案：device/sprd/scx35/emmc/recovery.fstab

展讯 5.0&5.1 按照芯片和存储方案规定了跟 4.4 一样的存放路径，还根据是否 secureboot 版本是否独立物理内卡分区等情况定义了不同的 recovery.fstab 文件，而具体使用哪个文件则根据项目的 device 定义组合成想要的名字对应的 fstab。具体可参见下图：



在这个文件中定义了需要用到的或者需要修改的分区列表。

原生的 recovery.fstab 一般会包含这几个分区：sdcard、system、data、boot、recovery。

如果需要对 bootloader 或者 modem 文件进行升级，或者支持使用内置 SD 卡，就需要将对应的分区加到这里。**制作升级包时，脚本会读取这个文件，以便根据预置升级步骤生成对应的脚本。**

recovery 主程序

源码文件可以查看 bootable/recovery/Androd.mk 中关于编译模块 recovery 的定义，对应编译的源文件如下：

```
LOCAL_SRC_FILES := \
    recovery.cpp \
    bootloader.cpp \
    install.cpp \
    roots.cpp \
    ui.cpp \
    screen_ui.cpp \
    verifier.cpp \
    adb_install.cpp
```

recovery 主程序主要就是对 recovery 系统的流程进行控制，或者升级，或者恢复出厂设置。**错误!未找到引用源。**

updater 主程序

Recovery 系统在进行升级时，会执行升级包中的文件/META-INF/com/google/android/update-binary。这个文件就是 updater 主程序。

查看 bootable/recovery/updater/Android.mk 可以得知这个程序的源码包含如下文件：

```
updater_src_files := \
    install.c \
    updater.c
```

updater 实际是个解释器，它运行后实际是运行升级包中的脚本文件/META-INF/com/google/android/updater-script。制作升级包时，会生成这个脚本文件，将 updater 和生成的脚本打包到升级包。

制作升级包的脚本

不管用哪种方式制作升级包，最终都是使用 build/tools/releasetools/ota_from_target_files 来生成的升级包。

这个脚本会根据 target 文件来生成升级包。target 文件包含 system.img、boot.img、recovery.img 中的所有文件，以及其它信息文件。当然其中包含了上面提到的 recovery.fstab。

制作升级包有两种方式：一种是一个特定版本的 target 文件制作为一个包含完整系统的升级包，称为**整体升级包**；另一种是使用两个版本的 target 文件，将其中一个版本作为基础版本，将另一个版本与基础版本的差异对比出来，制作成升级包，称为**差分升级包**。

这个脚本在执行时，会根据制作升级包的方式生成以及需要升级的文件，生成升级脚本，并将需要用到的文件打包，最终生成升级包。

2.2.3. 展讯特有分区以及其升级

展讯特有的分区及其升级：

- Spl：分区对应的 bin 为 u-boot-spl-16k.bin，主要是负责对硬件信息的确认与初始化，负责对 u-boot.bin 进行引导和 secureboot 校验。由于其为关键引导程序，会在 u-boot-spl-16k.bin 的基础上添加上一些校验值和 magicdata 等校验数据，故直接对此分区采用覆盖新版本 bin 的方法不适用，这部分展讯有专门的升级程序 splmerge 来进行升级，即根据新版本的 u-boot-spl-16k.bin（跟老版本不一样）计算新的 magicdata、校验值、pad_data 等写到到新版本的 bin 的头部，然后再写入到对应分区中去。
- Modem_bins：展讯的 modem 对应的各个 bins，其中有 CP 的各个 bins 和 wcn 的 bins，这个各平台 bins 会不一样，在 device 的项目定义中定义，后续在如何制作升级包中会详细讲到命名规则。这些 bins 的升级有的是直接全部覆盖，有的则是进行差分升级，会分开对待。进行何种升级是在 device/sprd/scxXX 下面的 modem.cfg 文件中来规定的。
- Nv 参数：包括 modem 的 nvitem 和 wcn 的 nvitem，由于一些参数跟硬件单体密切依赖，比如校准参数，所以这个分区升级也是采用展讯独有的升级程序 nvmerge 来完成，详细后边有介绍

在 4.4 平台上，这些分区的升级脚本是在展讯本地化脚本 releasetools.py 中，位置是 vendor/sprd/open-source/tools/ota/releasetools.py，然后在芯片级通用 Board 配置文件（如 scx15 系列为 device/sprd/scx15/BoardConfigCommon.mk）中通过变量 TARGET_RELEASETOOLS_EXTENSIONS 来指定这个路径的。

而 5.0&5.1 的展讯特有分区的升级则是由 releasetools.py 来读取 modem 配置文件 device/sprd/scx3XX/modem.cfg 来判断哪些 modem 需要升级升级方式如何。此文件也是在板级配置文件中定义了路径供编译系统使用：
scx35/BoardConfigCommon.mk:MODEM_UPDATE_CONFIG_FILE := device/sprd/scx35/modem.cfg。

3. Recovery 系统

Recovery 系统是一个区别于正常启动的系统（以下简称 Android 系统）的系统，可以用来对手机系统做出一系列的更改。如：恢复出厂设置、升级、root 等。

Android 系统包含 boot.img、system.img、userdata.img。而 Recovery 系统仅包含一个 recovery.img。和 boot.img 一样，recovery.img 包含了 kernel 和 ramdisk。而且正常编译出来的 recovery.img 和 boot.img 使用的 kernel 是一样的，recovery.img 和 boot.img 的区别在于 ramdisk。其中最重要的区别就是 recovery.img 和 boot.img 的 init.rc 不一样。Init.rc 的不同就决定了 kernel 在启动之后运行 Recovery 系统还是运行 Android 系统。

在系统启动进入 Recovery 模式后，可以从参数中获得升级包路径并进行升级，也可以在出现 Recovery 菜单后选择特定升级包进行升级。

3.1. 启动及运行流程

3.1.1. 启动流程

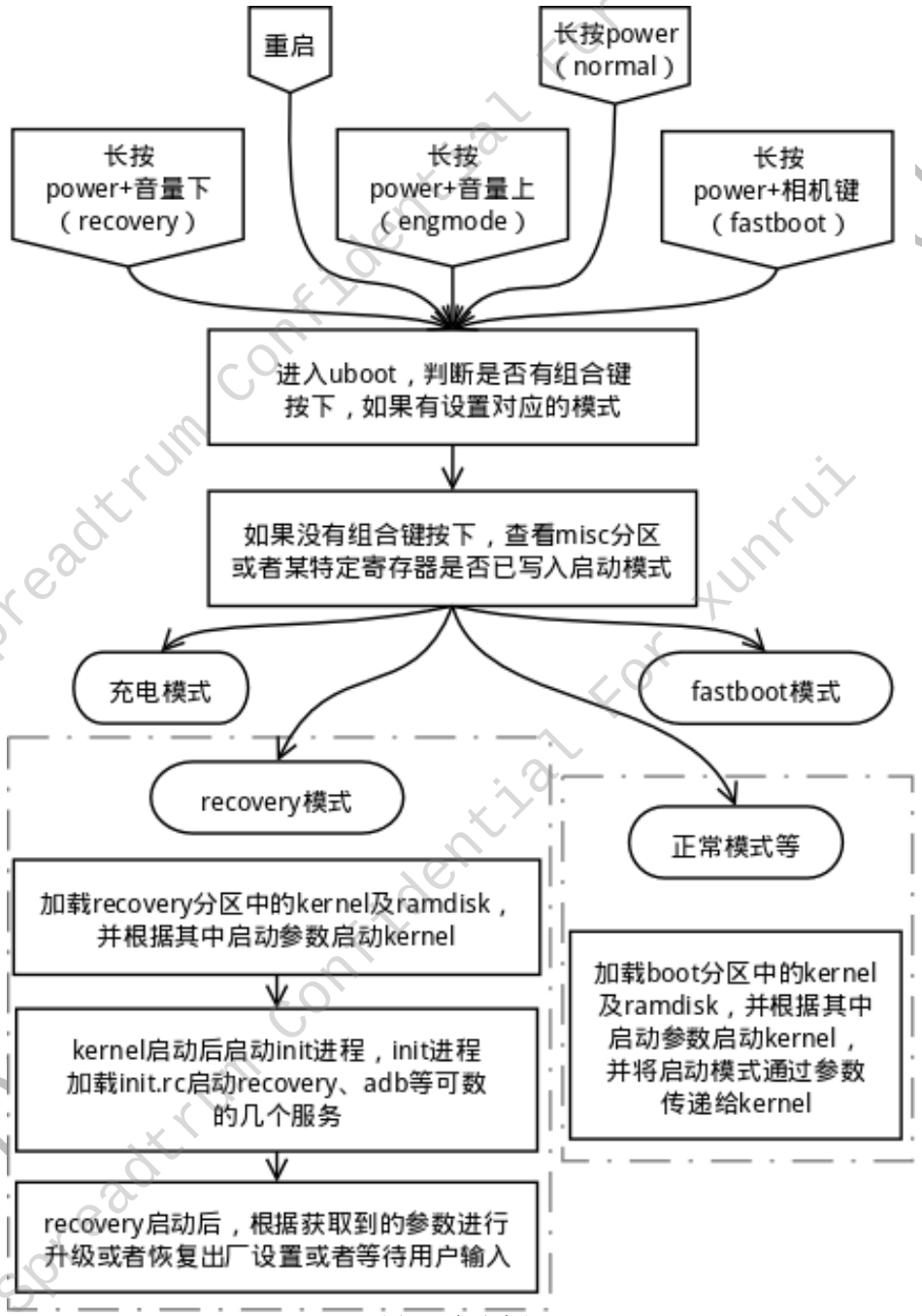


图 3-1 启动流程

3.1.2. 运行流程

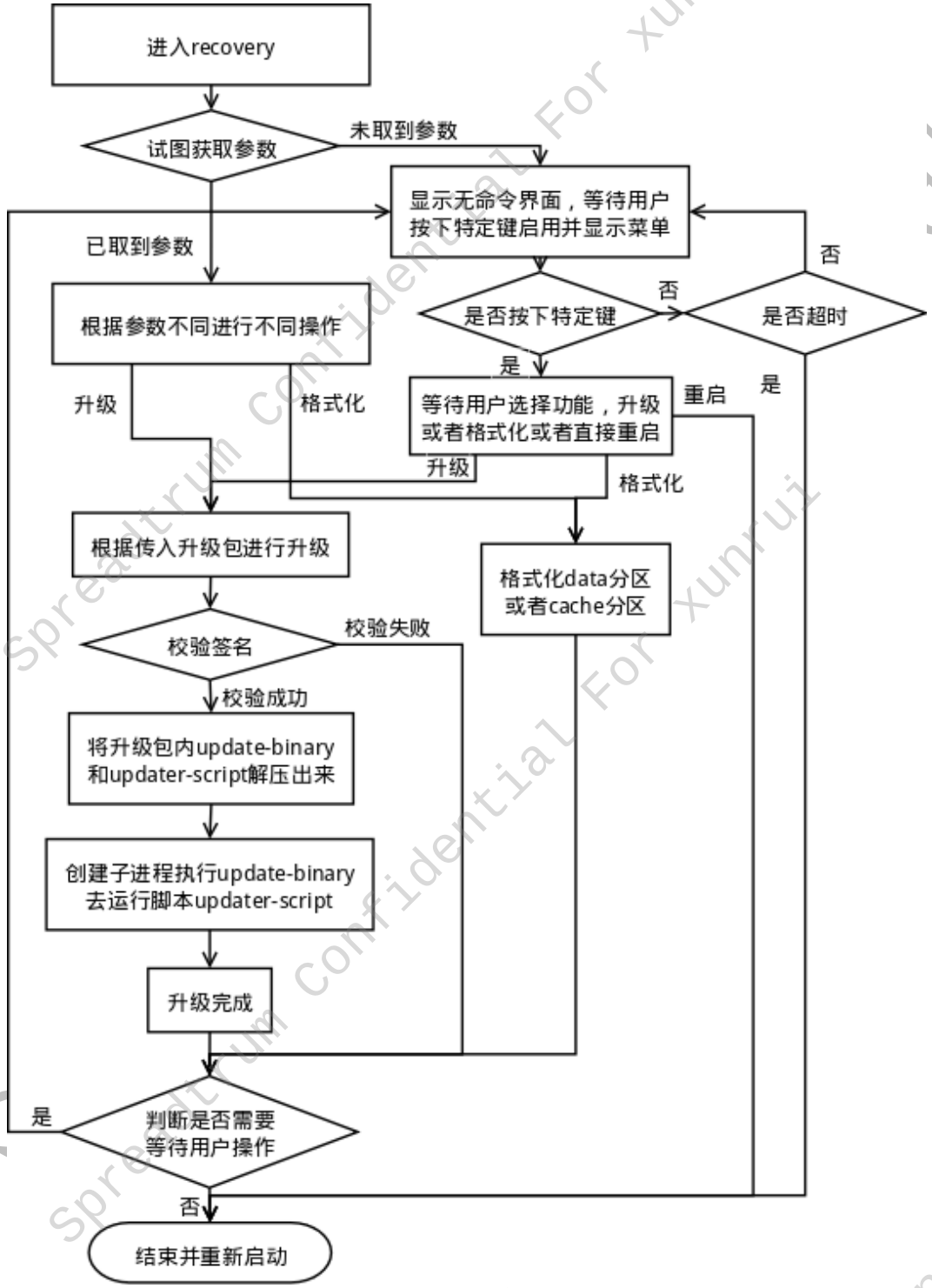


图 3-2 运行流程

4. 升级包制作与升级

4.1. 制作原理

编译升级包需要经历两个过程：

1. 将 image 以及一些参数制作成 target 文件
2. 从 target 文件制作成升级包文件

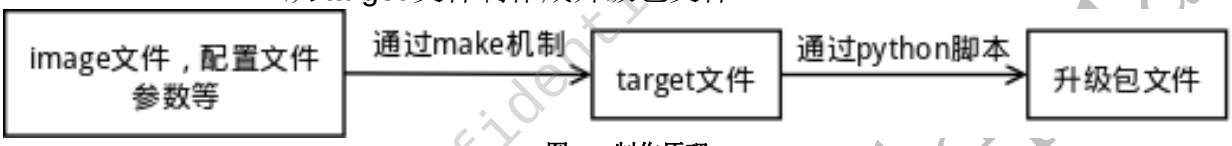


图 4-1 制作原理

4.1.1. Target 文件说明

target 文件是包含一个发布版本所有信息的一个集合。不但可以通过 target 文件制作升级包，也可以通过 target 文件制作所有的 image 文件。

除了 image 文件的信息，它还包含了制作升级包所需要的一些参数。这些参数大多存放在 target 文件的 META/misc_info.txt。在制作升级包的过程中这些参数会被读取出来，做不同的处理。

制作 target 文件是通过命令 make targe-files-package 完成，执行完后 target 文件的位置在：

out/target/product/board/obj/PACKAGING/target_files_intermediates/scx35_***-target_files-eng.apuser.zip

4.1.2. 升级包简单说明

按照制作方式来划分，升级包分两种：整体升级包和差分升级包。但从 Recovery 系统中升级的流程来看，两种升级包实质上是一样的。

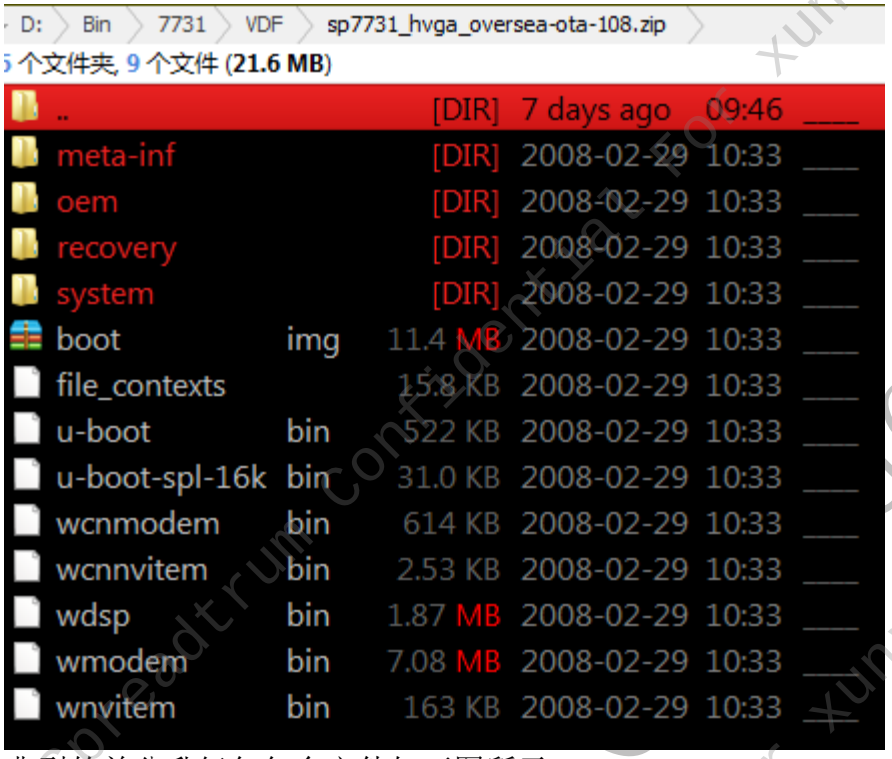
理论上来说，升级包中只要包含一个 update-binary 就好了，Recovery 系统在使用升级包升级时，只是简单的创建了一个进程去执行 update-binary 而已。

实际上，除了 update-binary，升级包中还包含了一个 updater-script。而这个 script 才是升级过程的真正主导者，update-binary 只是一个解释器，用来执行 script 中的命令。

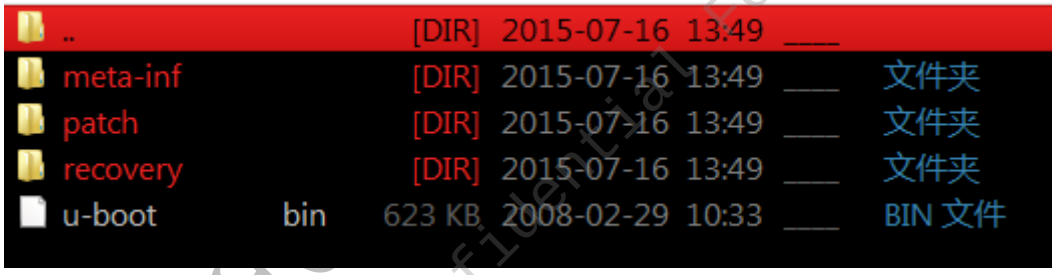
升级包内文件分布如下：

update-binary /META-INF/com/google/android/update-binary
updater-script /META-INF/com/google/android/updater-script
system 分区 /system 目录下所有文件
boot 分区 /boot.img

recovery 分区　/recovery 目录下文件
各个 modem.bin 即需要直接覆盖分区的 modem.bin
典型的整包升级包包含文件如下图所示：



典型的差分升级包包含文件如下图所示：



4.2. 制作方法

4.2.1. 整体升级包步骤

1 下载项目 AP 部分的代码
2 设置编译环境 source build/envsetup.sh lunch kheader
3 通过 make 命令全编整个工程
4 进入 device/sprd/XXXX/目录（XXXX 代表贵司项目名字的子目录），手动建立 modem_bins 子目录

5 然后将展讯发布的对应 AP 版本的 modem bins 按照 device/sprd/spXXXX/AndroidBoard.mk 中的规定更改名字后拷贝到 device/sprd/XXXX/modem_bins/目录下。

：不同的板子对应的 modem 文件是不一样的。如在展讯 4.0 平台上，几个 modem 文件的文件名必须这几个：dsp.bin，modem.bin，nvitem.bin。由于 shark 平台可能支持 TD 制式和 W 制式，需要添加对应的 TD 或者 W 文件，或者仅包含 TD 或者仅包含 W 或者两者都有；9620 平台还要支持 LTE 制式，还需要添加对应的 LTE 文件；如果采用 BT/WIFI/FM 三合一芯片的，还会有 wcnmodem.bin 和 wcnnvitem.bin。

最终需要拷贝的 bin 和要更改的名字还得根据 AndroidBoard.mk 文件中的定义来进行，以下是 7731 平台示例：

```
LOCAL_PATH := $(call my-dir)

$(call add-radio-file,modem_bins/wmodem.bin)
$(call add-radio-file,modem_bins/wnvitem.bin)
$(call add-radio-file,modem_bins/wdsp.bin)
ifeq ($(strip $(USE_SPRD_WCN)),true)
$(call add-radio-file,modem_bins/wcnnvitem.bin)
$(call add-radio-file,modem_bins/wcnmodem.bin)
endif
```

如果 USE_SPRD_WCN 是定义的，此型号就需要拷贝 5 个 bin 到上述目录。

具体名字更改办法下边小节会详细说明。

6 然后通过命令“make otapackage”编译 ota 整包 此命令运行完后会在 out 目录下得到 ota 整包：得到整体升级包：out/target/product/spXXXX/spXXXX-ota-*.zip。

注意：为了以后在版本升级时可以使用差分升级，同时要保留此版本对应的 target 文件。路径为：out/target/product/spXXXX/obj/PACKAGING/target_files_intermediates/*-target_files-*.zip

7 如果想制作跟 target 包对应的 pac 包，请在此时执行命令生成 pac 包

注意：请严格在执行完 make otapackage 后做 pac 包，因为 make otapackage 命令会对 system.img 等的时间做改动，只有在此步骤后做的 pac 包才是跟 target 包严格对应的！！

4.2.2. 差分升级包步骤

1 下载 A 版本代码，按照整包制作整包步骤 1~6 步执行，然后保存此版本对应的 target 包 A-target.zip

2 下载 B 版本代码，按照整包制作整包步骤 1~6 步执行，然后保存此版本对应的 target 包 B-target.zip

3 制作差分升级包，执行以下命令，由于 user 版本和 userdebug 版本使用的签名 key 不一样，故命令也不一样，而最终-k 后面参数为实际版本的 key 的放置目录，下面是 5.1 的编译命令：

Userdebug 版本：

```
./build/tools/releasetools/ota_from_target_files -i A-target.zip –k build/target/product/security/testkey B-target.zip A-B_update.zip
```

其中 A-B_update.zip 名字是最终的差分包名字

User 版本：

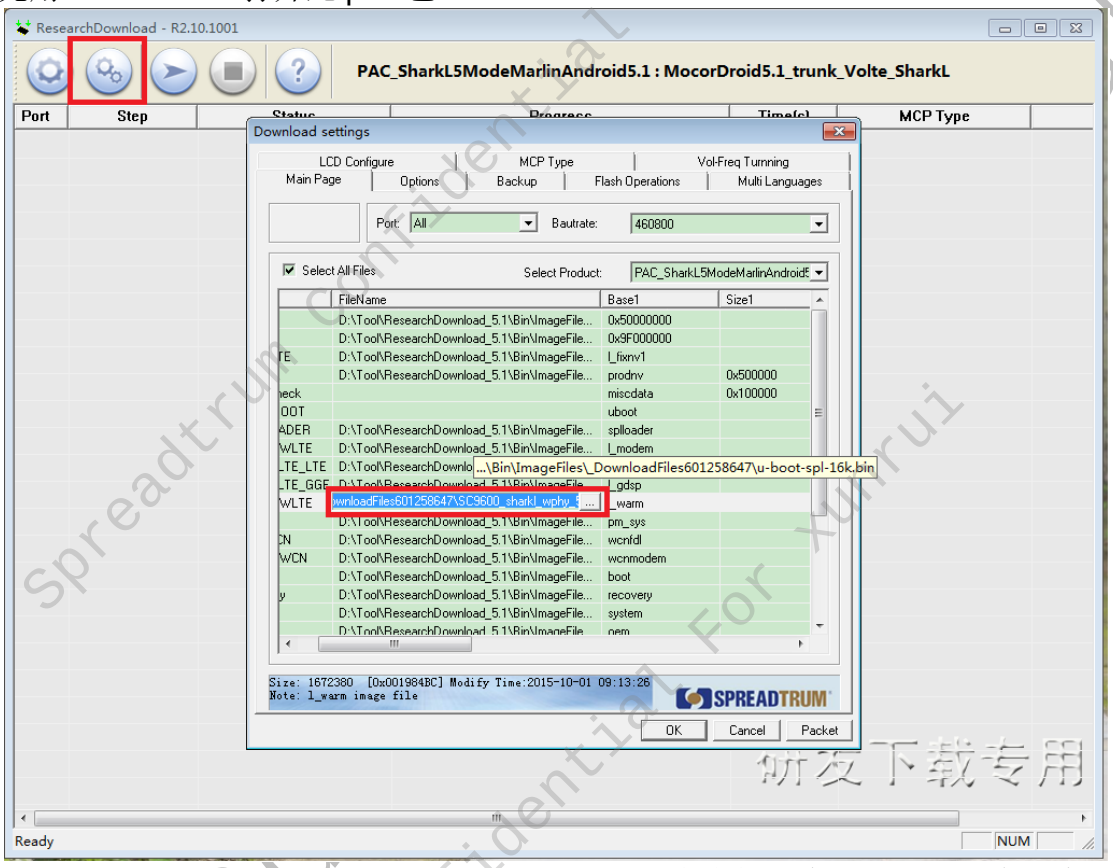
```
./build/tools/releasetools/ota_from_target_files -i  A-target.zip –k build/target/product/security/release/releasekey B-target.zip A-B_update.zip
```

如果是 6.0 平台使能了 dm_verify 功能，则 OTA 升级须使用块升级，做整包命令还是使用 make otapackage，而做差分包包命令变为如下：

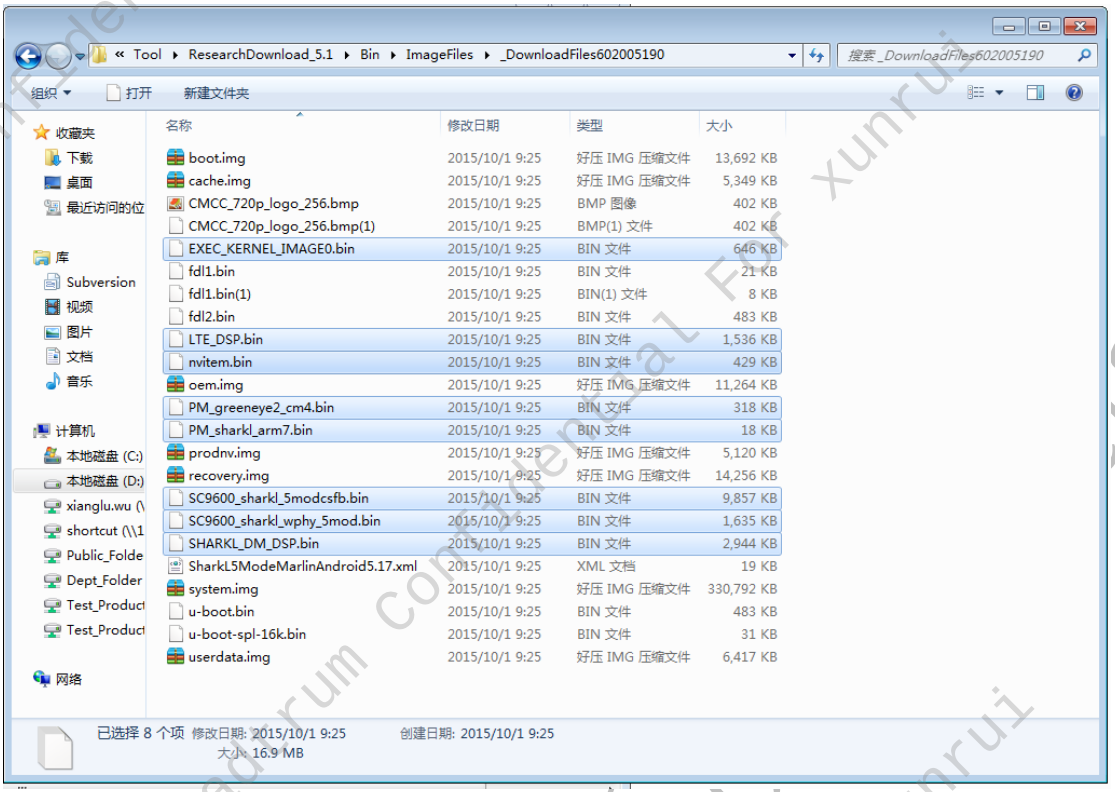
```
./build/tools/releasetools/ota_from_target_files –block -i  A-target.zip –k build/target/product/security/release/releasekey B-target.zip A-B_update.zip
```

4.3. Modem bins 改名方法

以下是以 9830 平台为例进行 modem bins 的改名说明。
拿到某个版本 SCM 提供的 pac 包后，按照下面示例的步骤来进行改名：
先用 downloader 打开此 pac 包：



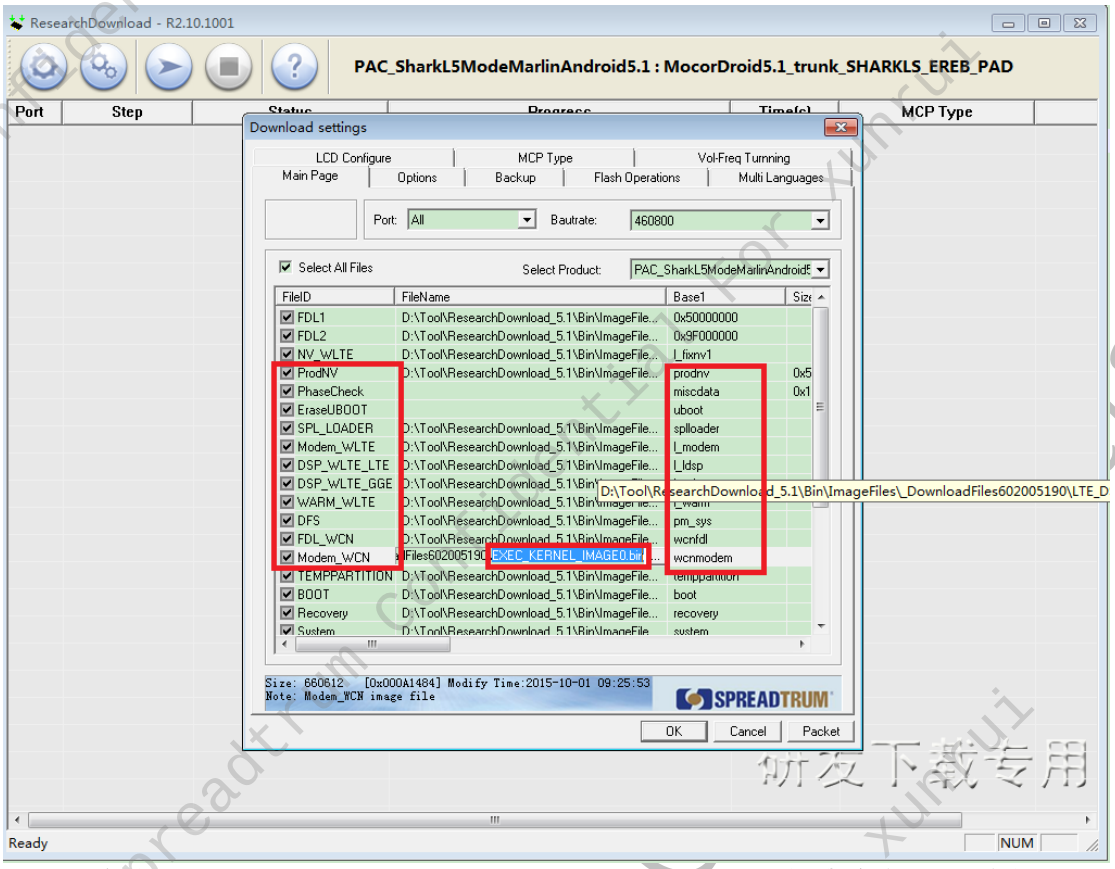
如上图，researchdownloader 打开 pac 包后，点击第二个设置按钮，会出来设置对话框，然后再对话框的第二列，是 pac 包的各个 img 的临时存放目录，如红圈标出所示，打开此目录后：



图中标出的都是 modem bins 和 wcn bins

其中跟 device/sprd/spXXX/AndroidBoard.mk 中要拷贝的 modem bins 的名字对照，可以从下载设置界面大体看出，如下图：

```
$(call add-radio-file,modem_bins/wcnmodem.bin)
$(call add-radio-file,modem_bins/wcnfdl.bin)
$(call add-radio-file,modem_bins/ltedsp.bin)
$(call add-radio-file,modem_bins/ltenvitem.bin)
$(call add-radio-file,modem_bins/pmsys.bin)
$(call add-radio-file,modem_bins/ltemodem.bin)
$(call add-radio-file,modem_bins/ltewarm.bin)
$(call add-radio-file,modem_bins/ltegdsp.bin)
```



一般的 device/sprd/spXXX/AndroidBoard.mk 列出的名字都跟右边的 base1 字段显示的名字差不多，以 wcnmodem 为例，它对应的 bin 就是 EXEC_KERNEL_IMAGE0.bin，把此 EXEC_KERNEL_IMAGE0.bin 改名为 wcnmodem.bin 即可，以此类推，所有 8 个 bin 的对应关系是：

EXEC_KERNEL_IMAGE0.bin -> wcnmodem.bin
fdl1.bin(1) -> wcnfdl.bin
PM_sharkl_arm7.bin -> pmsys.bin
SC9600_sharkl_wphy_5mod.bin -> ltewarm.bin
SHARKL_DM_DSP.bin -> ltegdsp.bin
LTE_DSP.bin - > ltedsp.bin
SC9600_sharkl_5modcsfb.bin -> ltemodem.bin
nvitem.bin -> ltenvitem.bin

由于平台众多，modem bins 的多少和名字也都有区别，如果不确定如何改名字，可以联系我们，我们会给出指导。

5. 展讯平台部分细节说明

5.1. nvmerge 和 splmerge

在展讯平台上，有一些特殊的分区。在烧录或者升级时不能将编译好的文件直接写入分区，需要进行一定的处理。如 nvitem、spl 就是这样的分区，为了能够正确的处理这两种分区的升级，开发了 nvmerge 和 splmerge，分别用于 nvitem 分区和 spl 分区的升级处理。

nvmerge 和 splmerge 的都是在升级包的 updater-script 中调用的，调用方式如下：

```
merge_spl("/cache/u-boot-spl-16k.bin", "/dev/block/mmcblk0boot0", "EMMC");
assert(run_program("/tmp/nvmerge", "/tmp/nvmerge.cfg", "/dev/block/platform/sprd-sdhci.3/by-name/wfixnv1", "/cache/wnvitem.bin", "/cache/merged_wnvitem.bin", "0x40000"));
write_emmc_image("/cache/merged_wnvitem.bin", "/dev/block/platform/sprd-sdhci.3/by-name/wfixnv1");
write_emmc_image("/cache/merged_wnvitem.bin", "/dev/block/platform/sprd-sdhci.3/by-name/wfixnv2");
```

5.2. nvmerge 具体说明

nvmerge 是用于处理 nvitem 分区升级的工具。可以将手机中的某些需要保留的 nv 项（如校准参数）备份下来，保证升级后不会被重置。这里的 nv 项，指的是与 modem 相关的一些参数，这些参数均保存在 nvitem 分区。

其中具体的要备份的 items 保存在 nvmerge.cfg 文件中，这个文件也会被打包进升级包，里面的内容可以根据需要进行增减，以达到备份某些特殊数据的目的，比如 IMEI 号等。

此文件在 4.4 平台上是存放在 bootable/recovery/nvmerge/下面，在 5.1 上是存放在 device/sprd/scx35/下面。

5.3. Secure boot 支持

Secure boot[1]是一个安全解决方案，以安全启动的方式保证手机系统没有被篡改过。通过对系统启动过程中用到的分区进行分级加密校验来实现。为了保证分区内容都是安全的，需要保证所有对分区进行操作的过程都是安全的，包括烧机和升级。

对于升级来说，制作升级包和使用升级包升级都会加入 secureboot 的判断。制作升级包时需要对相应的分区 image 进行签名。在使用升级包升级时，要在写入分区前校验整个分区数据是否正确。

如何在版本中使能 secureboot 请参考专门的 secureboot 文档。

支持 secure boot 后，升级包制作的方法没有变化。但由于 secure boot 签名时需要

[1] 可参见 secure boot 相关文档

用到密码，因此在制作升级包时增加了输入密码的过程。

根据现有的工具，每个 key 对应一个 product_name 和一个 password。由于目前采用单一 key 方式验证，因此只需输入一次就可以。在制作升级包时会在处理第一个需要签名的分区 image 时提示输入 product_name 和 password。处理后面需要签名的分区时直接使用这个 key 进行签名。

这种方式不适用于服务器制作升级包，因此提供了另一种方式。将 product_name 和 password 放入配置文件，在签名时直接从配置文件读取 product_name 和 password 信息进行签名。如果要使用这种方式，需要设置环境变量（SPRD_SECURE_BOOT_SIGN_CONFIG）为配置文件的路径（即把包含如下内容的文件的全路径赋给这个宏）。格式为：

```
[[[ passwd ]]] product_name
```

如： product_name 为 7731 密码为 12345678 的话，内容为：

```
[[[ 12345678 ]]] 7731
```

5.4. 关于 adb 的支持

在 4.4 上面，已经加入挂载 system 分区的内容，所以只要进入 recovery 模式，且是 userdebug 版本，就能连接使用 adb。

在 5.0&5.1 上面，需要先按音量上键，再按音量下键，重复此步骤 7 次以后，会挂载 system，则 adb 能使用。

6.0 上面，则把挂载 system 分区的动作放到 recovery 的菜单中（菜单名为“mount /system”）去了，选择此菜单执行后即可连接 adb（userdebug 版本）。

6. OTA 升级测试说明

1、准备升级前版本

1）整包升级需要保证手机版本比目标版本旧

2）差分升级需要保证手机版本为差分升级包的基础版本

2、升级

<1>通过 Android 进行升级

1）将“升级包”放到 sd 卡的根目录下并命名为 update.zip

2）设置->关于->系统软件更新，就会自动重启并升级

<2>直接进入 Recovery 模式升级

1）将“升级包”放入到 SD 卡或者 cache 分区根目录下

2）使手机置于关机状态

3）用组合键方式开机进入 Recovery 模式（手机上操作方法：关机状态下，按 power 键后立即按住音量下键，亮屏后松开 power 键和音量下键，进入 Recovery 模式）

4）根据“升级包”所在位置选择相应选项进入并选择升级包进行升级

5）升级完成后手动选择相应选项进行重启

注意： 6.0 平台上将 sd 卡使用为内部存储后已经不能再作为本地升级的存储升级包

的普通 sd 卡使用，因为其已经被挂在为 ext4 格式，而非 recovery 能识别的 FAT 格式。

7. 4.4/5.0&5.1/6.0 之间分区变更的大版本升级

此功能主要是进行分区重新划分，由于大版本间都伴随着分区大小的改变或者分区的增减，差分升级不适用，只能进行整包升级。这时候制作整包的办法是：

首先编译出目标 5.0&5.1/6.0 版本的 target 包，将 out/target/product/spXXXX/obj/PACKAGING/target_files_intermediates/*-target_files-*.zip 拷贝到工程根目录，然后运行以下命令制作变分区的大版本升级包：

./build/tools/releasetools/ota_from_target_files –HCSR *-target_files-*.zip full_partition_modified.zip

其中四个参数的意义分别是：

-H: 用于表示大版本升级，目前脚本中只是作为标志，做包时此参数无用

-C: 版本间分区名有变化，即各分区的设备名有变化，必须加入此参数

-S: 版本间分区大小有变化，或者有新加分区，必须加入此参数

-R: 需要内置 4.4 的 recovery image，主要用于断电保护，因为分区变更，recovery 分区会被覆盖，所以如果断电，无法继续升级，需备份 recovery 一下

7.1. 参数使用说明

- 其中在 4.4 升级到 5.0&5.1 或者是 6.0 时，伴随着分区变化和设备名的改变，上述四个参数都需要加上。如无分区变化，则 4.4 可以通过差分升级包升级到 5.0&5.1。而 4.4 到 6.0 或者 5.0&5.1 到 6.0 都会有分区变化，因为 modem bin 变大了。
- 5.0&5.1 升级到 6.0 时，分区有变化，但设备名无改变，只需加-HSR 参数即可。

7.2. 升级方法

手机里面请烧录一个 4.4 或者 5.0&5.1 的版本，然后再将 full_partition_modified.zip 改名为 update.zip 放入 sd 卡根目录后，从 recovery 模式的外部存储进行升级。此种情况只能通过 sd 卡进行升级，因为 data 分区的用户数据是备份到 sd 卡目录下的。如果分区有变化只能进行整包升级，无法进行差分包升级。

7.3. 大版本升级注意事项

- 排查系统应用以及内置第三方应用的数据库兼容性

  数据库的结构是否兼容，如4.4新增的字段但5.0没有，5.0本身的数据库可能有改变而4.4自己做了修改，导致升级后数据不兼容，无法读取，理论上来说，所有系

统应用（包含自己研发应用）和内置第三方应用都应该排查下， 重点关注的应用为Contacts(calllog)， MMS,Settings等

- 数据兼容问题

  需要各模块负责同事注意各自OTA升级的数据兼容问题，特别要注意采用物理内卡方案，从4.4进行大版本升级时，使用内外卡绝对路径的模块可能会存在数据丢失问题，例如mediaprovider。

- 需采用同一种存储方案

  必须采用同一种存储方案，例如，4.4上采用的是内卡为物理卡的方案，5.0&6.0上也要采用内卡为物理卡的方案，而不能采用内卡为虚拟卡的方案，否则会导致数据丢失

- Selinux开关

  大版本升级需要确认4.4版本selinux是否为打开的，否则升级后prodnv分区读写会有问题。因为AndroidL是启用selinux 的， 升级不会改变这个分区， 未启用selinux的prodnv分区的文件没有打标签，启用selinux后会有问题。

- 签名key保持一致

  这个签名key包括升级包本身的签名key和在编译版本时的各个apk的签名key，两个版本间必须统一。请参考build/core/Makefile中对DEFAULT_SYSTEM_DEV_CERTIFICATE的定义。

- Board名称是否一致

  大版本升级时，新旧版本的board名称须保持一致，否则会因为校验此名称失败而升级失败。

8. Recovery log 抓取办法

不论是恢复出厂设置还是进行 ota 升级，如果出现问题，可以按照如下办法抓取 recovery 的 log：

- 如果恢复出厂设置或者 ota 升级出错了，一般会在 recovery 里面进入 error 界面，这时候按音量上键或者下键（5.1 是 home 键或者 power+up 键）会进入菜单选择模式，这时候用 usb 线将设备链接电脑，在 4.4 上直接可以连 adb，而 5.1 需依次按上下键 7 次以上才会挂载 system 并连接 adb。通过如下命令序列获取 recovery 的 log，这种抓取办法需要版本为 userdebug 版本：

  adb root
  adb pull /tmp/recovery.log D:\log
  D 盘 log 下的 recovery.log 即是所需要的 log

- 如果恢复出厂设置或者 ota 升级出错了，停在 recovery 的 error 界面，按键进入菜单选择模式，然后选择“reboot system”，系统重启进入主系统的 Idle 界面了，可以选择如下命令获取有效 log：

  adb root　//须是 userdebug 版本
  adb pull /cache/recovery/last_log D:\log
  D 盘下面 log 里面的 last_log 即是刚重启前失败动作的 log。

- 如果是 user 版本在 ota 升级或者恢复出厂设置时失败，则 recovery 模式无法连接 adb 或者开机后也无法访问 cache 分区。这时可升级一个 userdebug 版本，但是保持 cache 分区不更新，然后开机获取 cache 分区的/cache/recovery/last_log 文件即可。
- 如果卡在 recovery 中的某些画面，按键也不会有动作。拔出电池重启还是会重新进入 recovery 模式，则可以如下获取 log
  连接 usb 线，看能否使用 adb，如果能则按照第一步办法获取 log
  如果不能使用 adb 则使用 reseachdownloader 擦除 misc 分区，然后开机进入主系统按照如下命令获取 log：
  adb root
  adb pull /tmp/recovery.log D:\log
  D 盘 log 下的 recovery.log 即是所需要的 log
- 另外一般恢复出厂或者升级出错后通过按键会显示出错界面，上面会显示最后的错误，虽然有时候无法看出错误详细信息，但也有一定作用。
- 5.1 和 6.0 上则在 recovery 的菜单里面添加了一项查看 log 的选项，拉到 log 的最后把错误拍照共享，也能帮助分析。具体操作步骤：
  出错后按照按键出现 recovery 菜单后，选择重启，然后关机，通过按键进入 recovery 模式，按键（5.1 是按 power+volup 或者 home 键，6.0 是按 power+volup 或者长按某键）出来菜单，选择“view recovery logs”这个菜单，然后选择 cache/recovery/last_log1（如果反复几次进入过 recovery，则需选对对应的后缀数字的文件）这个文件，滑动（或者按上下音量键）翻页到最后看看出错的语句，即为有用 log。