

Rockchip

Recovery 用户操作指南

发布版本:1.00

日期:2016.09

前言

概述

文档主要介绍 Rockchip Recovery 用户操作指南，旨在帮助软件开发工程师更快上手 Recovery 升级的开发及调试

产品版本

芯片名称	内核版本	Android 版本
		6.0.1

读者对象

本文档（本指南）主要适用于以下工程师：
技术支持工程师
软件开发工程师

修订记录

日期	版本	作者	修改说明
2016.09.02	V1.00	yhx、hkh	构建初始版本

目录

1 Recovery 简介	1-1
1.1 Recovery 模式简介	1-1
1.2 Recovery 模式在框架层的位置	1-1
2 编译 OTA 包.....	2-1
2.1 OTA 介绍.....	2-1
2.2 生成完整包	2-1
2.3 生成差异包.....	2-1
2.4 升级包打包脚本使用	2-2
2.5 注意事项	2-2
3 RK 平台分区升级说明.....	3-1
3.1 RkLoader 升级	3-1
3.2 Parameter 升级.....	3-1
3.3 Uboot、trust 升级	3-1
3.4 Resource 升级	3-1
3.5 releasetool 说明.....	3-2
4 Update.img 升级方式.....	4-1
4.1 支持存储方式	4-1
4.2 打包说明	4-1
5 配置说明.....	5-1
5.1 Log 重定向	5-1
5.2 屏幕旋转	5-1
5.3 屏幕分屏	5-1
5.4 文件保存	5-1
5.5 RecoveryUI 说明	5-1
5.6 支持 EXT4 文件系统.....	5-2
5.7 Update.img drm 签名校验	5-2
5.8 Boardid 串号	5-2
5.9 升级 U 盘升级包.....	5-3
5.10 代码结构	5-3
6 升级步骤.....	6-1
6.1 OTA 包本地升级操作步骤	6-1
6.2 OTA 包网络升级操作步骤	6-1
7 量产指导.....	7-1
7.1 升级原理简介	7-1
7.2 量产流程分析	7-3
7.3 编译固件注意事项.....	7-3
7.4 制作 pcba 小固件	7-3
7.5 制作带 pcba 测试功能的完整固件	7-4
7.6 制作 SD 升级卡	7-4

7.7 SD 升级卡的使用	7-5
7.8 注意事项	7-5
8 常见问题处理	8-1
8.1 4.2 OTA 升级到 4.4 注意事项	8-1
8.2 差分升级错误处理.....	8-2
8.3 升级包签名校验错误处理	8-2
8.4 NTFS 格式 U 盘升级支持	8-2
8.5 Logo 未更新成功	8-2
9 在线升级服务器	9-1
10 SecureBoot 签名工具	9-2

插图目录

错误！未找到目录项。

表格目录

错误！未找到目录项。

1 Recovery 简介

1.1 Recovery 模式简介

Recovery 模式指的是一种可以对安卓机内部的数据或系统进行修改的模式，（类似于 windows pe 或 DOS）。在这个模式下我们可以刷入新的安卓系统，或者对已有的系统进行备份或升级，也可以在此恢复出厂设置。

1.2 Recovery 模式在框架层的位置

Android 启动后，会先运行 bootloader。Bootloader 会根据某些判定条件决定是否进入 recovery 模式。Recovery 模式会装载 recovery 分区，该分区包含 recovery.img。Recovery.img 包含了标准内核(和 boot.img 中的内核相同)以及 recovery 根文件系统。

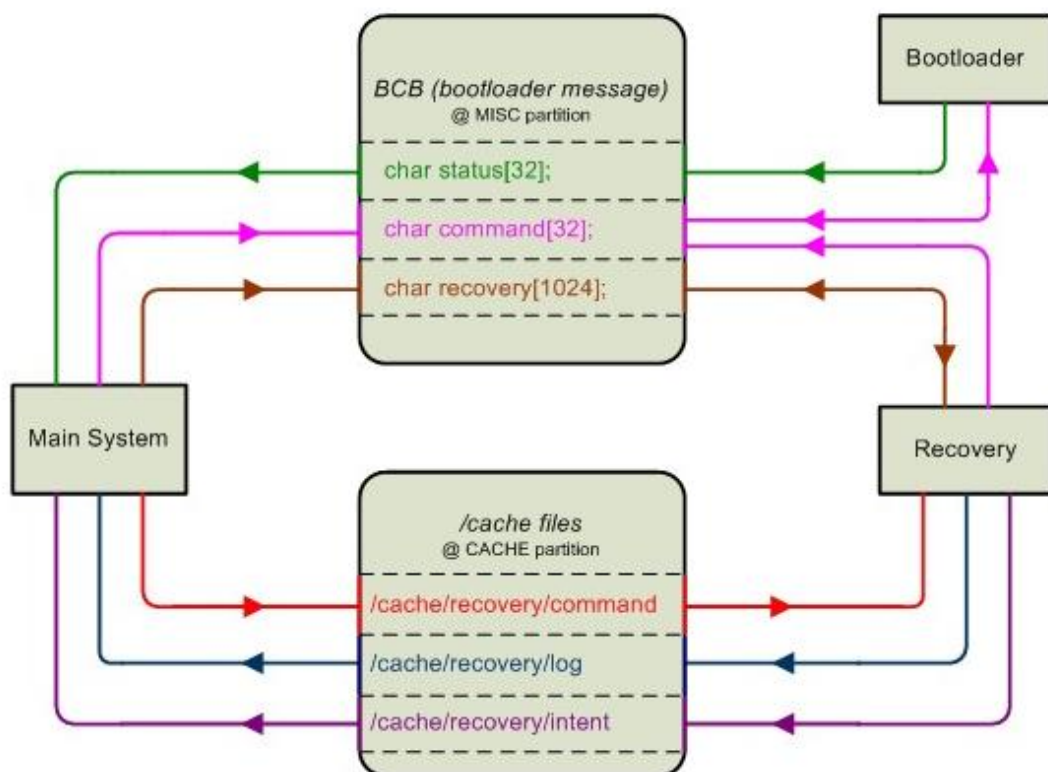
Android recovery 三个部分两个接口，recovery 的工作需要整个软件平台的配合，从架构角度看，有三个部分：

1. Main System: 用 boot.img 启动的 Linux 系统，Android 的正常工作模式。
2. Recovery: 用 recovery.img 启动的 Linux 系统，主要是运行 recovery 程序。
3. Bootloader: 除了加载、启动系统，还会通过读取 flash 的 MISC 分区获得来自 Main System 和 Recovery 的消息，并以此决定做何种操作。

两个通信接口：

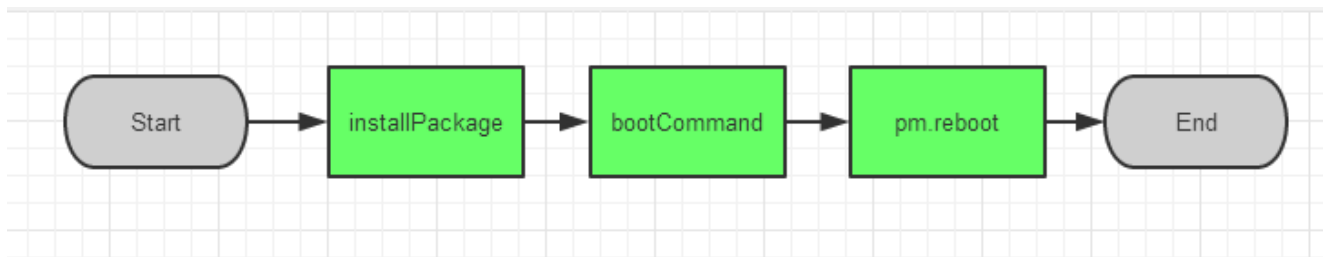
1. /cache/recovery/: command、log、intent
2. BCB (Bootloader Control Block) : misc 分区

我们先来看以上三部分是如何通信的，先下图：



Main System 如何进入 Recovery 模式： 当我们在 Main System 使用 update.zip 包进行升级时，系统会重启并进入 recovery 模式。在系统重启前，我们可以看到 Main System 定会向 recovery 域写入 boot-recovery（粉红色线），用来告知 bootloader 重启后进入 Recovery 模式。这一步是必须的，至于 Main System 是否会向 recovery 域写入值我们在源码中不能肯定这一点。即便如此，重启进入 Recovery 模式后，Bootloader 会从/cache/recovery/command 中读取值并放入到 BCB 的 recovery 域。而 Main System 在重启之前肯定会向/cache/recovery/command 中写入 Recovery 将要进行的操作命令。

下图是 Main System 进入 Recovery 模式调用接口的流程图：



- 1.installPackage: RecoverySystem 的接口，完成升级包路径转换，并调用 bootCommand。
- 2.bootCommand: RecoverySystem 的接口，将命令写入/cache/recovery/command，并调用 pm.reboot。
- 3.Pm.reboot:PowerManager 的接口，重启并进入 Recovery 模式。

2 编译 OTA 包

2.1 OTA 介绍

OTA (over the air) 升级是 Android 系统提供的标准软件升级方式。它功能强大，提供了完全升级（完整包）、增量升级模式（差异包），可以通过本地升级，也可以通过网络升级。

2.2 生成完整包

完整包所包含内容：system、recovery、boot.img

发布一个固件正确的顺序：

1. make -j4
2. make otapackage -j4
3. ./mkimage.sh ota

发布固件必须使用 ./mkimage.sh ota, 将 boot 与 kernel 打包，不需要单独烧 kernel，如果量产固件是分开的，将会影响后面差异包升级，除非你不需要用差异升级。

在 out/target/product/rkxxxx/目录下会生成 ota 完整包 rkxxxx-ota-eng.root.zip，改成 update.zip 即可拷贝到 T 卡或者内置的 flash 进行升级。

2.3 生成差异包

OTA 差异包只有差异内容，包大小比较小，主要用于 OTA 在线升级，也可 T 卡本地升级。OTA 差异包制作需要特殊的编译进行手动制作。

(1) 首先发布 v1 版本的固件，生成 v1 版本的完整包

(2) 保存

out/target/product/rkxxxx/obj/PACKAGING/target_files_intermediates/rk3188-target_files-eng.root.zip 为 rkxxxx-target_files-v1.zip，作为 v1 版本的基础素材包。

(3) 修改 kernel 代码或者 android 代码，发布 v2 版本固件，生成 v2 版本完整包

(4) 保存

out/target/product/rkxxxx/obj/PACKAGING/target_files_intermediates/rk3188-target_files-eng.root.zip 为 rkxxxx-target_files-v2.zip，作为 v2 版本的基础素材包。

(5) 生成 v1-v2 的差异升级包：

```
./build/tools/releasetools/ota_from_target_files -v -i rkxxxx-target_files-v1.zip  
-p out/host/linux-x86 -k build/target/product/security/testkey rkxxxx-target_files-v2.zip  
out/target/product/rk3188/rkxxxx-v1-v2.zip
```

说明：生成差异包命令格式：

ota_from_target_files

-v -i 用于比较的前一个 target file

-p host 主机编译环境

-k 打包密钥

用于比较的后一个 target file

最后生成的 ota 差异包

2.4 升级包打包脚本使用

2.5 注意事项

完整包和素材包是不一样的。

从生成差异包的方法可以知道，差异包是两个版本的素材包做差异生成，所以每发布一个版本固件必须保存 obj 下的素材包以及升级的完整包和./mkimage.sh ota 生成的各个 image。

如果有使用在线升级，必须给每个版本固件配置唯一的 ota 版本号：

device/rockchip/rksdk/rksdk.mk 中的

ro.product.version = 1.0.0（也可以是其他格式的版本号字符串）

3 RK 平台分区升级说明

3.1 RkLoader 升级

一、Loader 升级操作步骤

- a) 将需要升级的 loader, 以 RK*Loader*.bin 格式放到 device/rockchip/rkxxxx/ota/loader 目录下, OTA 打包时就会加入到升级包中。

二、注意事项:

- a) android5.1 放到目录 device/rockchip/common/loader, 如果没有请创建。
- b) android 4.4 放到目录 device/rockchip/rksdk/loader, 如果没有请创建。
- c) Loader 必须保证版本号比固件中的版本号更高。

3.2 Parameter 升级

一、parameter 升级操作步骤:

- a) 将需要升级的 parameter 文件到 device/rockchip/rkxxxx/ota/parameter/目录下, 以 parameter 名称开头即可, 如 parameter_sdk。该目录下只能存在一个 parameter 文件。

二、目前谷歌标准 ota 包方式升级能支持 parameter 更新, 但有一些限制:

- a) 更新 parameter 只能使用 ota 完整包升级, 不支持差异升级 parameter。
- b) 根据 RK29 Parameter File Format Ver1.1.pdf 描述, backup 区不能改大, 只能改小。如果想改大 backup 区, 必须擦除 idb, 所以本机升级不能去改大 backup 区。
- c) backup 区以及之前的分区大小和位置不能改变。之后的分区能随意改变和增加分区。
- d) user 区可能被改变, 所以不支持升级包放在内部 falsh 的升级, 只能放在外部 sd 卡升级。或者能保证其所在分区位置大小不变。
- e) recovery 区的地址不能改变, 否则重启后就找不到 recovery 了。

三、注意事项:

- a) android5.1 放到目录 device/rockchip/common/parameter/目录下, 如果没有请创建。
- b) Android4.4 放到目录 device/rockchip/rksdk/parameter/目录下, 如果没有请创建。

3.3 Uboot、trust 升级

3.4 Resource 升级

Resource.img, 已经默认包含在 boot.img 和 recover.img 里面了, 所以不建议再单独对 resource.img 进行升级。但是这里仍然提供单独升级步骤的方法:

1. ./build/core/Makefile 拷贝 resource.img 文件到 update.zip 中, 增加如下图红框那行代码

```
1553 ifdef TARGET_PREBUILT_RESOURCE
1554     $(info package add resource.img to BOOT and RECOVERY)
1555     $(hide) cp $(TARGET_PREBUILT_RESOURCE) $(zip_root)/BOOT/resource.img
1556     $(hide) cp $(TARGET_PREBUILT_RESOURCE) $(zip_root)/resource.img
1557     $(hide) cp $(TARGET_PREBUILT_RESOURCE) $(zip_root)/RECOVERY/resource.img
1558 endif
```

2. device/rockchip/common/releasetools.py 文件中添加如下修改，如下图

```
85
86 #*****
87 #resource package in the boot.img and recovery.img,so we suggest not to update alone resource.img
88 #*****
89
90 try:
91     resource = info.input_zip.read("resource.img")
92     print "wirte resource now..."
93     InstallResource(resource, info.input_zip, info)
94 except KeyError:
95     print "info: no resource image; ignore it."
96
97 try:
98     loader_bin = info.input_zip.read("LOADER/RKLoader.img")
99 except KeyError:
100     # print "warning: no rk loader bin in input target_files; not flashing loader"
101     print "no rk loader bin in input target_files; not flashing loader"
102     print "to add clear misc command"
103     info.script.ClearMiscCommand()
104     return
105
```

3.5 releasetool 说明

releasetool 是一个描述 RK 平台分区升级的 Python 脚本。

1. 下面是一些必须要实现的 extension 接口，供外部调用：

FullOTA_Assertions:	全量包升级脚本头部加入升级命令
IncrementalOTA_Assertions:	增量包生脚本头部加入升级命令
FullOTA_InstallEnd:	全量包尾部加入升级命令
IncrementalOTA_InstallEnd:	增量包尾部加入升级命令

2. 下面是 RK 平台分区升级接口：

Install_Parameter:	parameter 升级
InstallRKLoader:	loader 升级
InstallUboot :	uboot 升级
InstallTrust :	trust 升级
InstallCharge :	charge 升级
InstallResource :	resource 升级

4 Update.img 升级方式

4.1 支持存储方式

可以将 update.img 存到 SD 卡根目录，USB 根目录，内部存储/data/media/0/ 下，如果系统检测到有该包，会自动检测包是否合法，如果合法会重启进入 recovery 模式，并进行升级。

4.2 打包说明

1. 进入到目录 RKTools/linux/Linux_Pack_Firmware/rockdev/
2. 修改 package-file 文件，指向生成 img 的路径，或者将 img 拷贝进来。如图：

```
1 # NAME      Relative path
2 #
3 #HWDEF      HWDEF
4 package-file package-file
5 bootloader  Image-rk3399_box/RK3399MiniLoaderAll_V1.05.bin
6 parameter   Image-rk3399_box/parameter.txt
7 trust       Image-rk3399_box/trust.img
8 uboot       Image-rk3399_box/uboot.img
9 misc        Image-rk3399_box/misc.img
10 resource    Image-rk3399_box/resource.img
11 kernel      Image-rk3399_box/kernel.img
12 boot        Image-rk3399_box/boot.img
13 recovery    Image-rk3399_box/recovery.img
14 system      Image-rk3399_box/system.img
15 # 要写入backup分区的文件就是自身 (update.img)
16 # SELF 是关键字，表示升级文件 (update.img) 自身
17 # 在生成升级文件时，不加入SELF文件的内容，但在头部信息中有记录
18 # 在解包升级文件时，不解包SELF文件的内容。
19 backup      RESERVED
20 #update-script update-script
21 #recover-script recover-script
```

3. Parameter.txt：确认你的 parameter.txt 中 MACHINE_MODEL 的值是否正确，该值将用来校验 img 包是否合法，并且不能包含空格。如图所示，rk3399-box 前后都不能有空格。

```
1 FIRMWARE_VER: 6.0.1
2 MACHINE_MODEL:rk3399-box
3 MACHINE_ID: 007
4 MANUFACTURER: RK3399
```

- 4.运行 mkupdate.sh 脚本,会在该目录下生成 update.img,将拷贝到目标路径，即可进行升级。

5 配置说明

5.1 Log 重定向

1. 功能说明: Log 可以输出到串口、SD 卡、/cache/recovery/、三个地方。
2. 打开方式: 修改 bootable/recovery/Android.mk 文件:

REDIRECT_LOG_TO := UART	将日志输出到串口
REDIRECT_LOG_TO := CACHE	将日志输出到/cache/recovery/目录下
REDIRECT_LOG_TO := SDCARD	将日志输出到 SD 卡中
3. 支持平台: android 6.0。

5.2 屏幕旋转

1. 功能说明: Recovery 屏幕可以旋转 0° , 90° , 180° , 270° 。
2. 打开方式: 修改 device/rockchip/common/BoardConfig.mk 文件:

ROTATE_SCREEN := rotate_0	不旋转
ROTATE_SCREEN := rotate_90	旋转 90°
ROTATE_SCREEN := rotate_180	旋转 180°
ROTATE_SCREEN := rotate_270	旋转 270°
BOARD_HAS_FLIPPED_SCREEN := true	旋转 180° , 针对 LCD 屏幕装反情况。
3. 支持平台: android 6.0, 5.1 以补丁的形式存在。

5.3 屏幕分屏

1. 功能说明: Recovery 屏幕分屏, 将屏幕分为左右两部分, 显示内容一样, 用于 VR 功能。
2. 打开方式: 修改 bootable/recovery/Android.m 文件:

DOUBLE_SCREEN := NO	不分屏
DOUBLE_SCREEN := YES	分屏
3. 支持平台: android 6.0, 5.1 以补丁的形式存在。

5.4 文件保存

1. 功能说明: 将文件保存到/cache/recovery/Recovery_*, 格式化不会清除该文件。
2. 支持平台: android 6.0。

5.5 RecoveryUI 说明

recovery 作为一个简单的 rootfs, 提供非常有限的几个功能, 只包含了几个简单的库, UI 显示采用的

是直接刷 framebuffer 的形式。首先浏览一下 recovery main 函数里面的流程

```
Device* device = make_device();  
ui = device->GetUI();  
gCurrentUI = ui;  
  
ui->SetLocale(locale);  
ui->Init();
```

如上代码所示，make_device()函数返回一个 Device 设备，所以我们只需要继承 Device 类（一些虚函数接口），就可对 RecoveryUI 进行简单的修改。

1. 功能说明：自定义 recovery 菜单与按键实现
2. 打开方式：修改 device/rockchip/common/BoardConfig.mk 文件：
TARGET_RECOVERY_UI_LIB ?= librecovery_ui_rk30sdk
3. 修改方式：device/rockchip/common/recovery，里面有例子，可修改或参照。

5.6 支持 EXT4 文件系统

1. 功能说明：
2. 打开方式：修改 device/rockchip/common/BoardConfig.mk 文件：
 - a) TARGET_USERIMAGES_USE_EXT4 ?= true

5.7 Update.img drm 签名校验

- 1.功能说明：update.img 在升级时进行 drm 签名校验，防止第三方非法固件升级。Update.img 必须使用 secureboot 工具签名，并勾选 sign check 复选框。
- 2.打开方式：修改 device/rockchip/common/BoardConfig.mk 文件：
 - a) RECOVERY_UPDATEIMG_RSA_CHECK ?= true
- 3.支持平台：android4.4, android5.1

5.8 Boardid 串号

- 1.功能说明：一种根据烧录 Boardid 串号实现一种固件支持不同硬件不同国家定制的 one image，该方案支持谷歌 OTA 完整升级和差异升级。
- 2.打开方式：修改 device/rockchip/common/BoardConfig.mk 文件：
 - a) RECOVERY_BOARD_ID ?= true
- 3.支持平台：android 4.4,android5.1

5.9 升级 U 盘升级包

- 1.功能说明：当按键进入 recovery 时，自动升级 U 盘中的升级包(update.img 或 update.zip)。
- 2.打开方式：修改 device/rockchip/common/BoardConfig.mk 文件：
 - a) RECOVERY_AUTO_USB_UPDATE ?= false
- 3.支持平台：android 5.1

5.10 代码结构

本节将对 recovery 涉及的相关代码结构进行简单的说明。

1. Bootable/recovery/:
 - a) Recovery 的主程序代码，其中 recovery.cpp 是入口，
 - b) rkimage.cpp 是处理 update.img 的升级流程，
 - c) install.cpp 是处理 update.zip 的升级流程。
2. Build/tools/releasetools/:
 - a) OTA 升级包的 Python 脚本，控制完整包与差异包的生成。
3. Build/tools/drmsigntool/:
 - a) 如果开启 drm，生成 ota 包时对 boot.img 进行签名，使用 build/target/product/security/private.key,保证进行 OTA 升级后，drm 功能还能正常使用。
4. Build/tools/mkparameter/:
 - a) 编译 OTA 时打包可升级的 parameter 的工具。
5. Build/tools/remkloader/:
 - a) 编译 OTA 时打包可升级的 loader 的工具。
6. Build/target/product/security/:
 - a) 编译 OTA 包时签名使用的密钥。
- 7.device/rockchip/common/recovery/:
 - a) Recovery 菜单及按键定制
 - b) 在 android 4.4 上是 device/rockchip/rksdk/recovery 目录
8. Device/rockchip/common/loader:
 - a) 将需要升级的 loader 放在该目录，可以打包到 ota 升级包中。
 - b) 4.4 目录 device/rockchip/rksdk/loader
 - c) 6.0 目录 device/rockchip/rkxxxx/ota/loader

9. Device/rockchip/common/parameter:

- a) 将需要升级的 parameter 放在该目录，可以打包到 ota 升级包中。
- b) 4.4 目录 device/rockchip/rksdk/loader
- c) 6.0 目录 device/rockchip/rkxxxx/ota/loader。

10. out/target/product/rk3188/obj/PACKAGING/target_files_intermediates:

- a) OTA 升级包编译生成的素材吧，要做差异包必须保存该素材包。

11. Device/rockchip/common/releasetools.py

- a) 定义 OTA 包中包含的内容。

6 升级步骤

6.1 OTA 包本地升级操作步骤

1. 参照第二章，生成 OTA 包，并重新命名为 update.zip
2. 将 update.zip 拷贝到 USB、SD 卡根目录，或/data/media/0/ 目录下
3. 会自动检测升级包，并弹出升级对话框。
4. 自动重启升级：重启进入 Recovery 系统中自动完成 OTA 包的升级，此时不能断电，等待升级完成会自动重启到 Android 主界面。

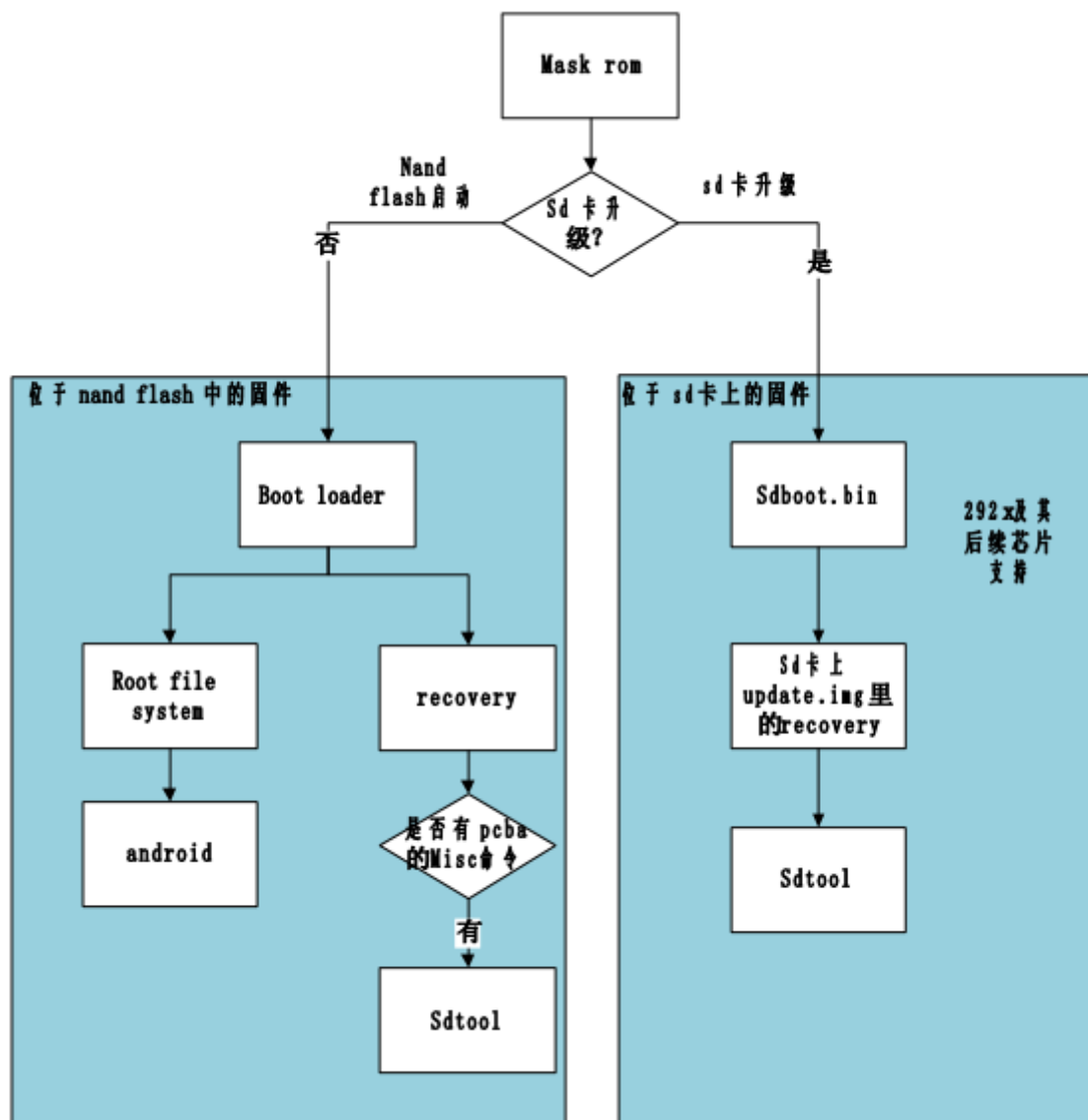
6.2 OTA 包网络升级操作步骤

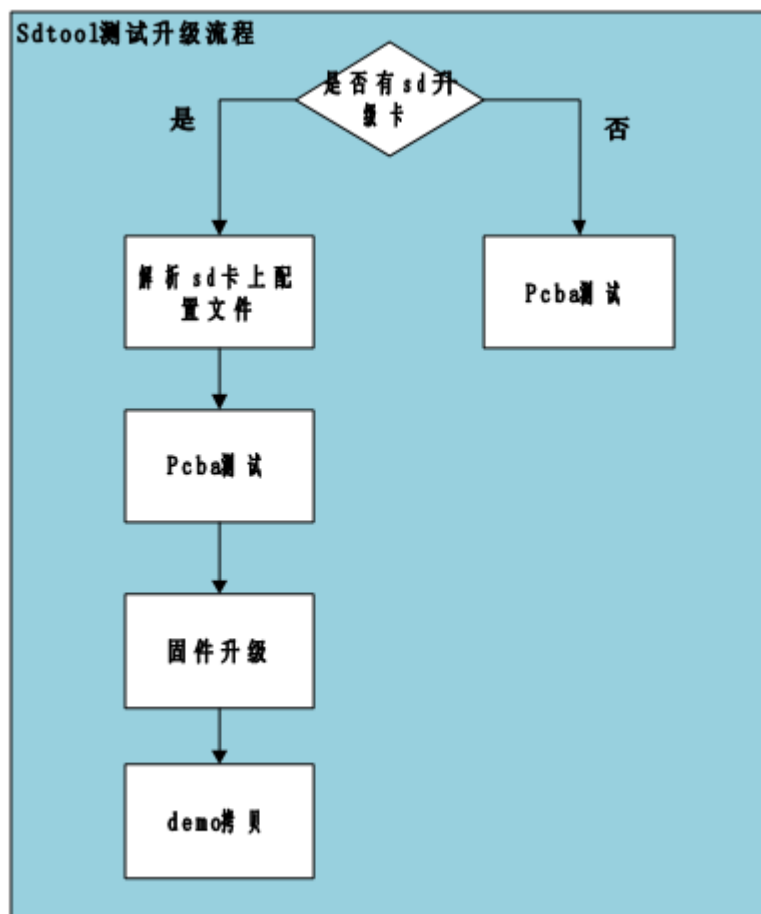
1. 自动检测网络 OTA 升级包，提示下载升级。
2. 自动重启升级。

7 量产指导

7.1 升级原理简介

本节旨在指导开发商如何对 RK 平台 Android 终端进行量产升级，以及让开发工程师理解升级原理。在介绍升级流程之前，我们先大致了解一下 RK 平台的固件加载流程。





上电第一步，首先启动的是固化在芯片内部的 MaskRom: MaskRom 完成芯片基础的初始化后，将判断是否插入特殊的 SD 升级卡。这里分两种情况：

第一种从 flash 启动，去加载位于 flash 中的 Bootloader，Bootloader 再根据 misc 分区中指令决定加载 recovery 还是 android 的根文件系统，如果是加载根文件系统，那文件系统会把 android 的 system 挂载起来，启动各项服务。

第二种从 SD 卡启动，maskrom 引导卡上的 sdboot.bin，sdboot 会做一些文件系统初始化等操作，然后将 update.img 从 recovery 中解压出来在内存中运行，此时 recovery 会识别到命令启动工厂模式将控制权交给 sdtool，sdtool 里可以做一些工厂量产相关的处理流程，比如 pcba 测试，大固件升级，demo 文件拷贝等等。

这样我们可以得到几种升级模式：

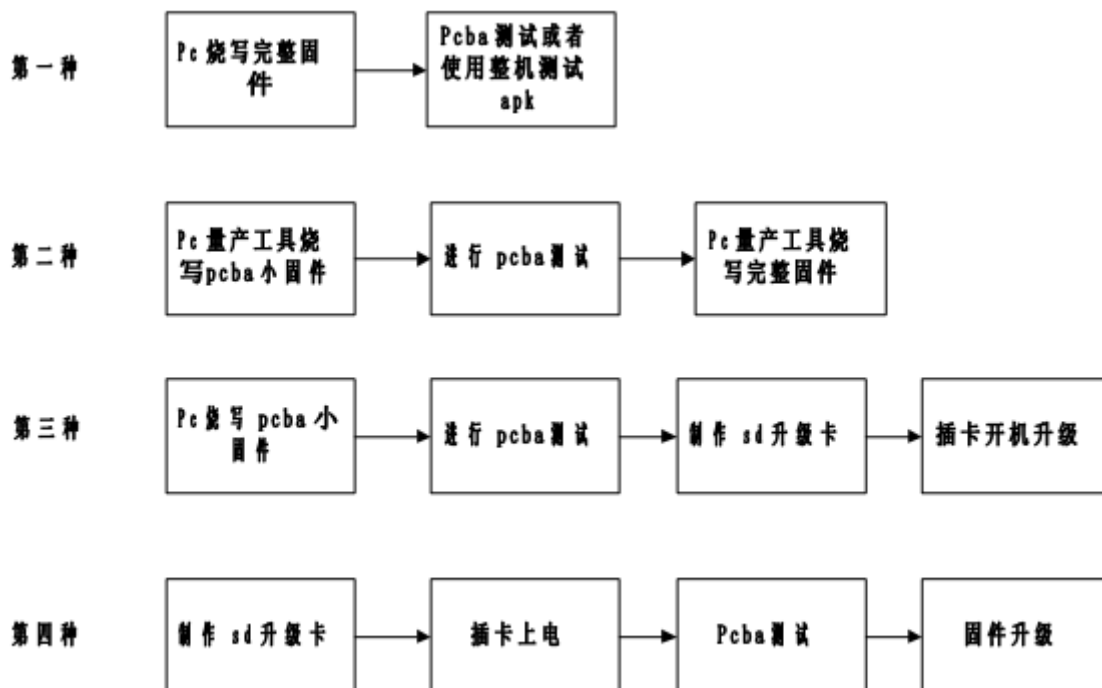
1. MaskRom 模式，系统在 Flash 中找不到 BootLoader，系统就会停留在 MaskRom 状态，等待 USB 连接 PC 升级。一般在贴片完成后，第一次开机，机器就停留在 MaskRom 状态；也可以通过开机时短接 Flash 的数据线，让系统扫描 Flash 出错而停留在 MaskRom 状态。该状态下可以通过 pc 端工具直接烧写固件。

2. BootLoader 模式，BootLoader 模式生效的前提是在机器中已经有烧入可工作的 Loader.bin。进入 BootLoader 模式的方法一般是通过组合键开机，在其他分区校验出错时，系统也会停留在 BootLoader 状态。这种模式下也可以通过 pc 端工具进行固件烧写。

3. Recovery 模式，Recovery 包含两种模式，通过不同的启动命令区分，一种是最终用户模式，一种是工厂模式（也就是上图中的 sd 卡升级的情况）。最终用户模式包含本地升级功能，也是 OTA 升级的基础。工厂模式包括 pcba 测试、固件升级、音视频 demo 文件拷贝，主要为了支持工厂生产的相关需求。

7.2 量产流程分析

目前的量产流程主要分为四种：



前三种升级流程全系列芯片支持，第四种流程目前只有 292x 及以后的芯片支持。使用第四种流程可以完全脱离 pc，因此生产效率高。对于 maskrom 非支持 sd 卡引导的芯片，建议使用第三种流程，贴片厂用 pc 烧一个 pcba 小固件进行测试，整机厂商使用 sd 卡升级完整固件。

7.3 编译固件注意事项

这里再次强调，为了与 google 的 ota 升级统一，sd 卡升级时不会单独烧写 kernel 分区，kernel 需要跟 boot 一起打包。

编译完，生成固件时使用 ./mkimage.sh ota 命令，这样生成出来的固件即是 boot 里包含 kernel 的固件。

7.4 制作 pcba 小固件

1. pcba_small_misc.img 用来定义一个 pcba 小固件，也就是说打包了该 misc 的 update.img 就具备了 pcba 测试小固件的功能。

2. 将 loader、parameter、pcba_small_misc.img、recovery 打包到 update.img 里：

请看打包工具的 package-file 文件配置方法：

```
1 # NAME      Relative path
2 #
3 #HWDEF      HWDEF
4 package-file package-file
5 bootloader  RK292xLoader(L)_V1.18.bin
6 parameter   parameter
7 misc        Image/pcba_small_misc.img
8 #kernel     Image/kernel.img
9 #boot       Image/boot.img
10 recovery    Image/recovery.img
11 #system     Image/system.img
12 # 要写入backup分区的文件就是自身 (update.img)
13 # SELF 是关键字，表示升级文件 (update.img) 自身
14 # 在生成升级文件时，不加入SELF文件的内容，但在头部信息中有记录
15 # 在解包升级文件时，不解包SELF文件的内容。
16 # RESERVED不打包backup
17 #backup     backupimage/backup.img
18 backup      RESERVED
19 #update-script  update-script
20 #recover-script recover-script
```

3. 打包好 pcba 小固件后通过量产工具烧写到机器后，每次启动机器都能自动进入 pcba 测试。

7.5 制作带 pcba 测试功能的完整固件

1. pcba_whole_misc.img 用来定义带 pcba 功能的完整固件，也就是说打包了该 misc 的大固件具有烧写完第一次启动进行 pcba 测试的功能。

2. 将 loader、parameter、pcba_whole_misc.img、boot、recovery、system 等都打包到 update.img 里，即是带 pcba 测试功能的完整固件。用量产工具烧入机器第一次启动会进入 pcba 测试，测试通过后下次启动会进入 android，测试不通过下次启动还是进 pcba 测试。

3. 如果不需要第一次启动进入 pcba 测试，可以不打包 pcba_whole_misc.img，而使用普通的 misc，这样量产工具升级完第一次启动只做格式化操作然后正常启动 android。

7.6 制作 SD 升级卡

使用 SD_Firmware_Tool 创建一张 sd 升级卡，具体使用方法参考工具目录下的《SD Card BootUserGuide》文档。



1) 如果贴片厂不需要升级完整固件，只需要做 PCBA 测试，有以下两种方式：

- a) 只勾选 PCBA 测试
- b) 只勾选 SD 启动

注意：选择此 SD 启动时，需要将 kernel 配置中的（ Device Drivers -> MMC/SD/SDIO card support -> RK29 SDMMC0 controller support 关掉，不然会造成无法从 SD 卡启动）

使用以上两种方法都可以实现：机器直接插卡进行 PCBA 测试，由于方法 a)需要先将固件升级到 flash 中才进行测试，所需要的时间比方法 b)大概要长 10 秒左右，所以建议使用方法 b)进行 PCBA 测试，方法 b)无法在 PCBA 中进行 SD 卡测试，但是由于是 SD 卡启动，可以证明 SD 卡功能是正常的，所以就不需要测试 SD 卡。

2) 整机厂商需要升级完整固件并拷贝 demo 数据文件，制作的时候应该选择完整固件，并只勾选固件升级，不勾选 pcba 测试。

7.7 SD 升级卡的使用

1. 创建好 sd 卡后，对于支持 sd 卡引导的芯片，只需要将卡插入机器，重新开机即可
2. 对于不支持 sd 卡引导的芯片，如 3066，需要在机器烧过 pcba 小固件的情况下，插卡开机才能使用，也就是上文说的第三种量产升级流程。

7.8 注意事项

sd 升级卡不是普通卡，是带有引导信息的卡，支持 sd 卡引导的机器插着卡开机就会从卡启动，不要当成普通卡拷入 update.img 在 android 状态下去做本地升级。如果需要用卡去本地升级需要先用 SD_Firmware_Tool 工具恢复这张卡。

8 常见问题处理

8.1 4.2 OTA 升级到 4.4 注意事项

4.4 新加入一个 **metadata** 分区，而且要求这个分区必须要存在并格式化后才能正常启动，4.2 很多客户没有这个分区，这就要求要升级的同时更新 **parameter** 并在升级最后格式化 **metadata**。

我们的 **sdk** 升级 **parameter**，如果用 **update.img** 则不能放在内部 **flash**，必须放在外部 **sd** 卡，因为升级时会先烧写 **parameter** 后进行一次重启再升级其他分区，第二次重启新的 **parameter** 里 **user** 区可能地址已经变了造成找不到升级包。

而通过 **update.zip** 升级方式虽然可以放在内部 **flash** 升级 **parameter** 但还是要求新的 **parameter** 中 **user** 区地址不能改变。

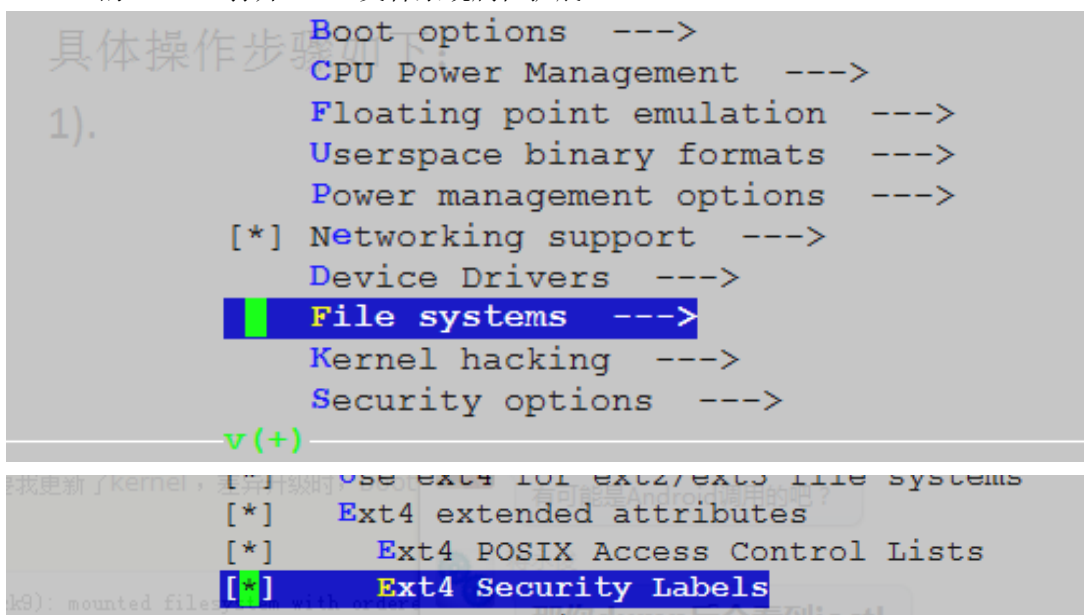
总结：

1. 如果 4.2 未出货的固件直接使用 4.4 上的 **parameter**(带有 **metadata** 分区)升级时加入 **metadata** 的格式化操作补丁，可以从 4.2 升级到 4.4。
2. 如果 4.2 已经出货，但是分区大小足够 4.4 固件的话，可以使用去掉 **metadata** 分区的补丁，并且 **parameter** 用 4.2 一样的，可以从 4.2 升级到 4.4。
3. 如果 4.2 已经出货，需要在线升级的，在满足 **user** 区地址不变的情况下可以升级 **parameter**。

如果满足以上条件则可以继续往下看，由于 4.2 的 **kernel** 没有打开 **ext4** 文件系统属性扩展支持，而 4.4 的升级程序需要使用到扩展中的 **api**，所以 4.2 系统通过 **ota** 升级到 4.4 系统需要先升级一次 4.2 的 **recovery** 才能升级 4.4。注意：只有通过 **ota** 包方式升级有如下限制，通过 **update.img** 升级可以直接升级到 4.4，不用升级两次。

具体操作步骤如下：

1. 4.2 的 **kernel** 打开 **ext4** 文件系统属性扩展



```
Boot options --->
CPU Power Management --->
Floating point emulation --->
Userspace binary formats --->
Power management options --->
[*] Networking support --->
Device Drivers --->
File systems --->
Kernel hacking --->
Security options --->

v (+)

Use ext4 for ext2/ext3 file systems
[*] Ext4 extended attributes
[*] Ext4 POSIX Access Control Lists
[*] Ext4 Security Labels
```

2. 生成 4.2 的完整包或差异包，升级一次。
3. 生成 4.2-》4.4 的完整包或者差异包，再次升级。

8.2 差分升级错误处理

差分升级容易出错，原因是生成的烧到机器里的 image 固件与 make otapackage 编译出来的完整包不一致，包括人为的对 image 的修改或是对完整包的修改，以下总结了几种容易造成差异升级失败的不当操作：

1. 发布固件未使用 ./mkimage.sh ota 生成，量产时单独烧写 kernel，原因是 ota 包中的 boot.img 是有带 kernel 的，所以生成的差分补丁都是基于带 kernel 的 boot，在需要差分升级到下一个版本时，boot.img 的补丁会打不上。
2. 未通过编译系统直接修改 out 目录下的文件后再生成 ota package。
3. 使用固件工厂工具直接修改 update.img。
4. build/tools/release/security/ 目录有 publicKey.bin 和 privateKey.bin 这两个文件，update.img 却未使用 secureboot 工具签名。原因是源码中包含 privateKey.bin 时 make otapackage 生成完整包时会对 boot.img 进行 drm 签名，但是量产 update.img 却没有对 boot.img 进行签名，造成两个固件不一致。
5. 发布每个版本固件未保存对应的 target 素材包，差异包是用 out/target/product/rk3188/obj/PACKAGING/target_files_intermediates/ 目录下的素材包制作的，如果发布的每一版固件没有保存对应的素材包将无法生成可用的差异包。

8.3 升级包签名校验错误处理

保证升级前后使用同一套签名密钥，即工程源码下 build/target/product/security/ 保持一致，比如 4.2 要升级到 4.4，要将 4.2 的该目录下的 key 覆盖到 4.4 下，再生成 ota 升级包

8.4 NTFS 格式 U 盘升级支持

内核默认不支持 NTFS 格式 U 盘的挂载，所以 recovery 中无法支持。如需支持 NTFS，需要打开 kernel 中的相应配置，recovery 默认已支持：

```
Symbol: NTFS_FS [=n]
Type : tristate
Prompt: NTFS file system support
Location:
  -> File systems
(3)   -> DOS/FAT/NT Filesystems
Defined at fs/ntfs/Kconfig:1
Depends on: BLOCK [=y]
Selects: NLS [=y]
```

8.5 Logo 未更新成功

1. 位置：Logo 包含 uboot logo 和 kernel logo，这两个 Logo 都存储于 resource.img 里面。
2. 显示策略：分为两种情况
boot.img 包含/不包含 resource.img

策略一：如果 parameter 里面有声明 resource 分区

优先从 resource 分区读取 Logo

策略二：如果 parameter 里面没有声明 resource 分区

优先从 boot 里面读取 Logo

由于 boot.img 较大，读取需要时间，为了较快的显示 Logo，我们采用策略一，所以如果出现 Logo 升级失败，应该优先检查 resource 是有升级成功。

3. 总结：所以 Logo 如果未更新成功，是 resource.img 升级失败，可以参考 recourse.img 升级。

9 在线升级服务器

9.1 服务器运行环境

Ota 服务器建议运行在 ubuntu 系统，文档以 ubuntu 系统下搭建方法为例进行说明。

9.2 Ubuntu 连接数优化设置

由于 Ubuntu 默认的进程最大开启文件句柄数为 1024（ulimit -n 命令查看），限制了服务器所能打开的 socket 连接最大值，无法发挥服务器的性能，可以通过以下步骤设置：

- 1) .打开/etc/security/limits.conf，里面有很详细的注释，找到如下设置(如果没有就插入)
* soft nofile 65535
* hard nofile 65535
- 2) .编辑/etc/pam.d/common-session，加入一行
session required pam_limits.so
- 3) .修改后重启 ubuntu 即可。

9.3 JDK 安装

Ota 服务器要求 JDK1.6 以上版本，具体如何安装配置可以参考网上，这里不做详细说明。

9.4 服务器配置

1. 解压 apache-tomcat-7.0.29.zip 到任意目录下

如：[mmk@hp-PC:~/webdevelop/apache-tomcat-7.0.29](#)

2. 修改整个目录的权限

如：`mmk@hp-PC:~/webdevelop$ chmod 775 -R apache-tomcat-7.0.29`

3. 服务器应用部署在 webapps/OtaUpdater

`mmk@hp-PC:~/webdevelop/apache-tomcat-7.0.29/webapps/OtaUpdater/WEB-INF$ ls`
`classes lib log4j.properties manifest.xml packages web.xml`
manifest.xml 和 packages 目录需要根据产品型号和版本号，手动进行配置

4. manifest.xml 配置文件写法说明

这里以新加入一个产品 TD8801 为例进行说明

```
<product name="TD8801" full_package_path="null" rkimage_path="null">
  <version name="1.0.0" package_path="packages/ TD8801/1.0.0/1.0.2.zip" />
  <version name="1.0.1" package_path="packages/ TD8801/1.0.1/1.0.2.zip" />
  <version name="1.0.2" package_path="packages/ TD8801/1.0.2/1.0.3.zip" />
</product>
```

product 标签定义了产品属性，产品名称为 TD8801，full_package_path 指的是 ota 最新完整包路径，如果没有就设为 null 一个产品下有多个版本通过 version 标签来定义，name 为版本号（与固件编译时设置的版本号一一对应），package_path 为该版本对应的升级包路径（相对主工作目录）。

5. 每个产品各个版本对应的升级包可以放在 packages 下任意位置，只要 manifest.xml 指定好相对路径

即可。

9.5 服务器监听端口修改

默认监听端口 2300，如果用户需要修改，按如下步骤：

mmk@hp-PC:~/webdevelop/apache-tomcat-7.0.29\$vimconf/server.xml

找到如下内容：

```
<Connector port="2300" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

将 2300 改为您需要的端口号就可以了。

9.6 服务器运行和停止

1. 服务器包部署在具有外网能访问 ip 地址的主机上，需要向网络运营商申请域名和 IP。
2. 配置完 manifest.xml，并将升级包都按规则放好后，就可以运行服务器了。
3. startup.sh 运行服务器。
4. shutdown.sh 停止服务器。

9.7 服务器运行日志

运行日志保存在 apache-tomcat-7.0.29/mylogs 目录下，每天会产生一个日志文件，方便 Debug。

9.8 编译注意事项

Device/rockchip/rk*****/rk*****.mk

```
PRODUCT_NAME := rk29sdk
PRODUCT_DEVICE := rk29sdk
PRODUCT_BRAND := Android
PRODUCT_MODEL := TD8801
PRODUCT_CHARACTERISTICS := tablet

PRODUCT_PROPERTY_OVERRIDES += \
    ro.product.version = 1.0.0 \
    ro.product.ota.host = www.rockchip.com:2300
```

1. 定义好产品型号：修改 PRODUCT_MODEL 值为相应的产品型号，如 TD8801，A22，TSB 等。注意产品型号不能带空格。

2.定义当前编译的固件版本号（重要） `ro.product.version` 属性值 `1.0.0` 即为当前系统版本号，这个版本号与服务器的 `manifest.xml` 文件中定义的必须一致，以后每发布一个新版本固件都切记更新版本号，如果不更新或是错误的版本号，将使服务器无法工作或是推送了错误的升级包， 将造成不可预计的后果。需要注意的是修改过这个属性要将 `out/target/product/rk29sdk/system/build.prop` 删除再 `make` 改变才能生效，或者 `makeclean` 后再 `make`。

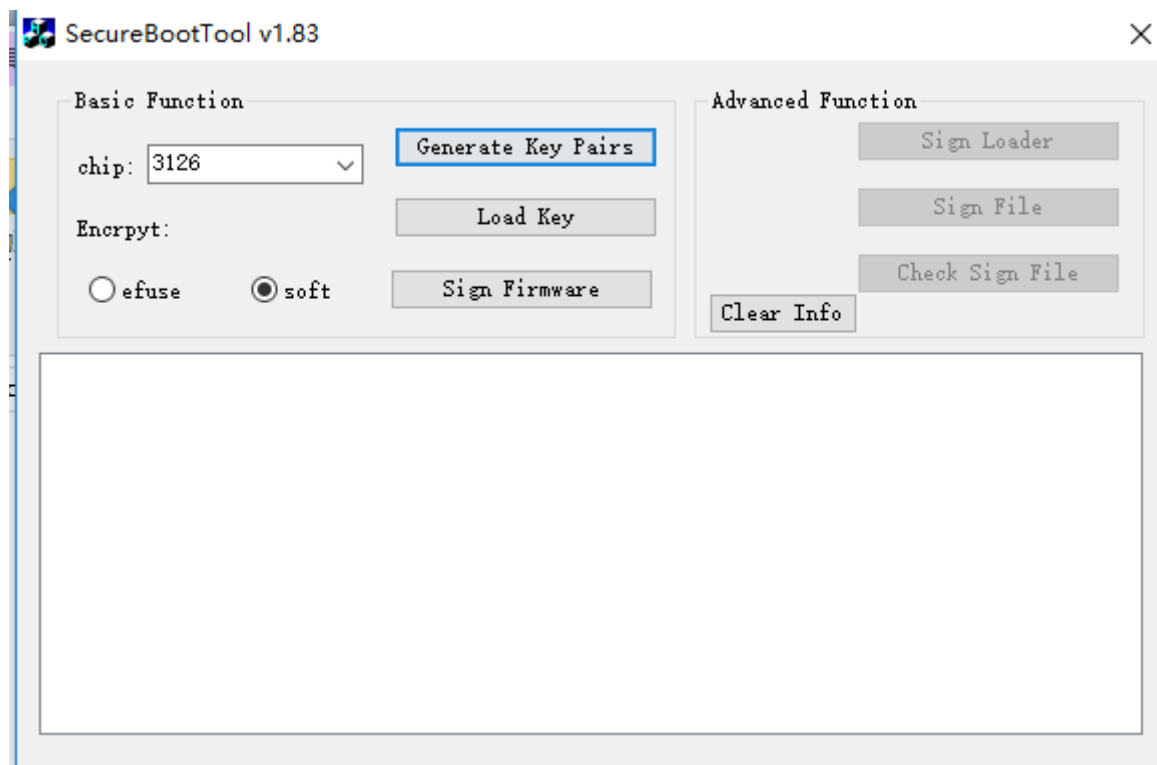
3.定义 ota 服务器地址：`ro.product.ota.host` 属性定义了 ota 服务器主机地址，这个域名需要向网络运营商 申请。`2300` 指的是端口号，对应服务器监听的端口。

10 SecureBoot 签名工具

本节只对工具的使用进行简单的介绍，更为详细的请参考工具目录下的文档说明。

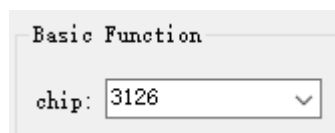
《RockchipSecureBootApplicationNote》

10.1 win 平台签名



如上图是 win 平台签名工具的界面，以下对签名工具做详细的介绍。

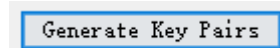
配置说明：



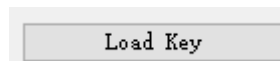
：选择芯片型号，3188 之前的芯片选择 others。



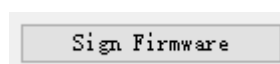
：配置加密类型是写 efuse 加密还是软件加密（注：3128, 3288, 3228, 3368 支持 efuse 级加密，其他的旧芯片不支持）。



：生成 RSAKEY，每款机器只能生成一次，请一定备份好 KEY，如果丢失，那么机器将不能再更新固件。



：加载之前备份的 RSAKEY（工具支持 openssl 生成的.pem 的 2048key）。



：进行签名，签名的固件必须使用 ./mkimage ota 生成。

以上步骤完成后，签名成功会有如下弹框：



10.2 Linux 平台签名

```
projects@bogon:~/hkh_test/rk3228-4.4/SignOtaPackageRelease$ ./SecureBootConsole
SecureBootConsole v1.83
*****Usage*****
SecureBootConsole -k|-kk SaveKeyDir //generate key -k(1024) -kk(2048)
SecureBootConsole -si privatekey image//Sign Image
SecureBootConsole -sl publickey loader //Sign Loader
SecureBootConsole -slx privatekey publickey loader [LE|BE]//SignEx Loader
SecureBootConsole -sh privateKey firmware //Sign Firmware's digest
SecureBootConsole -sf privateKey file //Sign file
SecureBootConsole -gh file [160|256|256LE]//Generate hash
SecureBootConsole -sz privateKey otazipfile //Sign ota zip file
***** End *****
```

Linux 平台签名工具，如上图，运行不带参数的 SecureBootConsole 会有 Usage，这里不再做重复说明。

11 Block 升级方式

11.1 配置文件

1. 编辑 build/core/Makefile, 增加如下一行

```
diff --git a/core/Makefile b/core/Makefile
index cce2647..db61c25 100644
--- a/core/Makefile
+++ b/core/Makefile
@@ -1667,6 +1667,7 @@ @@ $(INTERNAL_OTA_PACKAGE_TARGET): $(BUILT_TARGET_FILES_PACKAGE) $(DISTTOOLS)
    @echo "Package OTA: $"
    $(hide) MKBOOTIMG=$(MKBOOTIMG) \
    ./build/tools/releasetools/ota_from_target_files -v \
+   --block \
    -p $(HOST_OUT) \
    -k $(KEY_CERT_PAIR) \
    $(if $(OEM_OTA_CONFIG), -o $(OEM_OTA_CONFIG)) \
```

2. 编辑 device/rockchip/common/device.mk

```
hkh@RD-DEP1-SERVER-163:~/rk3368/5.1/device/rockchip/common$ git diff
diff --git a/device.mk b/device.mk
index 03eeb4c..291be88 100755
--- a/device.mk
+++ b/device.mk
@@ -505,8 +505,8 @@ @@ endif

# setup dm-verity configs.
# uncomment the two lines if use verity
-#PRODUCT_SYSTEM_VERITY_PARTITION := /dev/block/platform/ff0f0000.rksdmmc/by-name/system
-#$(call inherit-product, build/target/product/verity.mk)
+PRODUCT_SYSTEM_VERITY_PARTITION := /dev/block/platform/ff0f0000.rksdmmc/by-name/system
+$(call inherit-product, build/target/product/verity.mk)

ifeq ($(strip $(BUILD_WITH_GOOGLE_MARKET)), true)
ifeq ($(strip $(BUILD_WITH_GOOGLE_MARKET_ALL)), true)
```

注意: 打开的同时请确保 system 分区的路径正确, 不同芯片这个路径是不一样的。

3. 编辑 device/rockchip/rkxxxx/fstab.rk30board.bootmode.emmc, 切换 system 的配置到 verify

```
diff --git a/fstab.rk30board.bootmode.emmc b/fstab.rk30board.bootmode.emmc
index 2e49d90..a38b119 100644
--- a/fstab.rk30board.bootmode.emmc
+++ b/fstab.rk30board.bootmode.emmc
@@ -3,9 +3,9 @@
# The filesystem that contains the filesystem checker binary (typically /system) cannot
# specify MF_CHECK, and must come before any filesystems that do specify MF_CHECK

-/dev/block/platform/ff0f0000.rksdmmc/by-name/system          /system          ext4          ro,noatime,nodiratime,noauto_da_alloc
+#/dev/block/platform/ff0f0000.rksdmmc/by-name/system          /system          ext4          ro,noatime,nodiratime,noauto_da_alloc
# use this line below instead to enable verity
-/dev/block/platform/ff0f0000.rksdmmc/by-name/system          /system          ext4          ro,noatime,nodiratime,noauto_da_alloc
+/dev/block/platform/ff0f0000.rksdmmc/by-name/system          /system          ext4          ro,noatime,nodiratime,noauto_da_alloc
/dev/block/platform/ff0f0000.rksdmmc/by-name/cache            /cache           ext4          noatime,nodiratime,nosuid,nodev,noauto_da_alloc,discard
/dev/block/platform/ff0f0000.rksdmmc/by-name/metadata         /metadata        ext4          noatime,nodiratime,nosuid,nodev,noauto_da_alloc,discard
/dev/block/platform/ff0f0000.rksdmmc/by-name/userdata         /data            f2fs         noatime,nodiratime,nosuid,nodev    wait,check,encryptable=/met
```

4. 编辑 device/rockchip/rkxxxx/BoardConfig.mk, 按你的需求修改 system 分区大小, (请考虑后续 OTA 的需求, 且与 parameter 中的值一致)


```
//MAX-SIZE=512M, for generate out/.../system.img  
BOARD_SYSTEMIMAGE_PARTITION_SIZE ?= 1073741824  
BOARD_FLASH_BLOCK_SIZE ?= 131072
```

或者路径 device/rockchip/common/BoardConfig.mk

5. 拷贝打包脚本到源码根目录 cp device/rockchip/common/mkimage_verity.sh ./
6. 按正常编译流程编译，最后一步打包的时候把 mkimage.sh ota 替换成 mkimage_verity.sh ota
7. 差异包生成也需要增加--block 选项

```
6 ./build/tools/releasetools/ota_from_target_files --block v -i $source_zip -p out/host/linux-x86 -k build/target/product/security/testkey  
$target_zip $target_name
```