全志科技
Allwinner Technology

H6 Android N

OTA 使用说明文档

0.8

2017.2.27

文档履历

版本号	日期	制/修订人	内容描述
0.8	2017.2.27		正式版本

目录

1. 前言

1.1 编写目的

本文档目的是让系统开发人员了解 OTA 升级的概念与整体架构，掌握 OTA 升级的流程以及使用方法，并了解 H6 平台 Android N 系统的定制化设计。

1.2 适用范围

本模块适用于 H6 平台 Android N 系统。

1.3 相关人员

系统开发人员。

1.4 专业术语

- OTA: (over-the-air) 空中下载技术。指 Android 系统提供的标准软件升级方式，即通过无线网络下载更新包并无损失地升级系统，而无需通过有线方式进行连接。
- boot: boot 分区，包含 Linux 内核以及最小的 root 文件系统。它负责挂载 system 分区以及其他分区。
- system: system 分区，包含 AOSP(Android Open Source Project) 的系统应用程序以及库文件。正常操作下，该分区是以只读形式挂载。它的内容只能够在 OTA 升级过程中改变。
- recovery: 包含第二个 Linux 系统，包括 Linux 内核以及名为 recovery 的二进制可执行文件，recovery 的作用是用于读取更新包并将其内容更新至其他分区。
- vendor: vendor 分区包含在 AOSP(Android Open Source Project) 中不包含源码的系统程序以及库文件，该分区以只读形式挂载，它的内容只有在 OTA 升级才会被改变。

- cache: 用于暂时保存少数应用程序以及存储下载的 OTA 升级包。其他程序如使用该分区保存文件有可能会丢失。OTA 升级可能导致该分区数据被完全清除。同时 cache 分区保存 OTA 升级中的日志文件和 command 文件，保存升级文件的备份。
- data：用于存储 ota 包，在特殊情况下可更改 data 分区的内容。

2. OTA 升级流程

2.1 OTA 运行原理

Android 平台提供 Google diff arithmetic 差分机制，升级包支持完整升级以及差分升级，OTA 运行原理图如下所示：

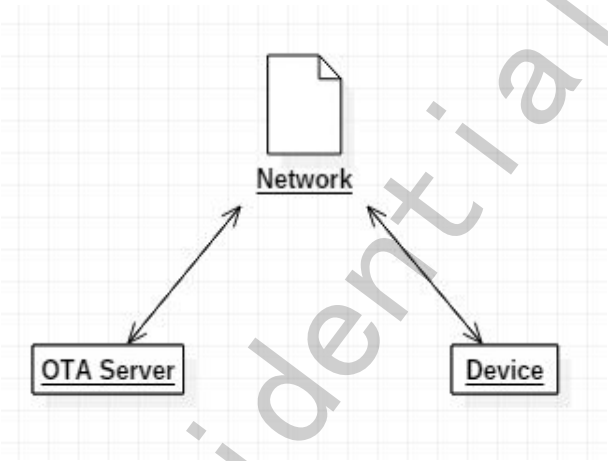

图 1: OTA 运行原理图

1. OTA Server 负责对更新包进行上传，下载以及版本的管理。
2. 开发者在修改 Android 系统后，通过差分制作工具制作出差分包，并使用客户端进行更新包上传和版本管理。
3. 设备通过 wifi 网络进行连接和下载，最后完成更新工作。

2.2 OTA 升级流程介绍

典型的 OTA 升级流程具体可以分为如下步骤：

1. 设备会周期性检查 OTA 服务器，并确认更新包的可升级性。用户也可以通过将更新包放入盒子，或者以 U 盘，SD 卡，移动硬盘的形式进行升级。

2. 更新包下载到 cache 或者 data 分区，加密签名将会利用 system/etc/security/otacerts.zip 文件进行验证，通过验证后才能进行 OTA 升级。
3. 设备重启进入到 recovery 模式，至此 recovery 分区中的内核以及 system 分区代替了 boot 分区进行启动。
4. recovery 二进制程序被 init 进程启动, 并在 cache/recovery/command 中找到已经下载的更新包。
5. recovery 利用在/res/keys 的公钥验证更新包，假如验证成功进入下一步，否则将信息保存到 cache/recovery/目录下的日志文件，并重启设备。
6. 更新包的数据被解压并用于更新 boot,system 或者有需要时更新 vendor 分区。此时新的 recovery 分区的内容被保存在 system 分区当中。boot0，uboot 也在该过程进行更新。
7. 设备重新启动，并分为以下两步进行：

    - 更新的 boot 分区被装载，它会挂载并启动 system 分区的可执行程序。
    - 在正常启动的过程，系统将会对比 recovery 分区当前的内容以及期望的内容，如果有区别，recovery 分区将会被期望内容所更新。此时系统更新完成，将升级信息保存到日志文件，系统重启。

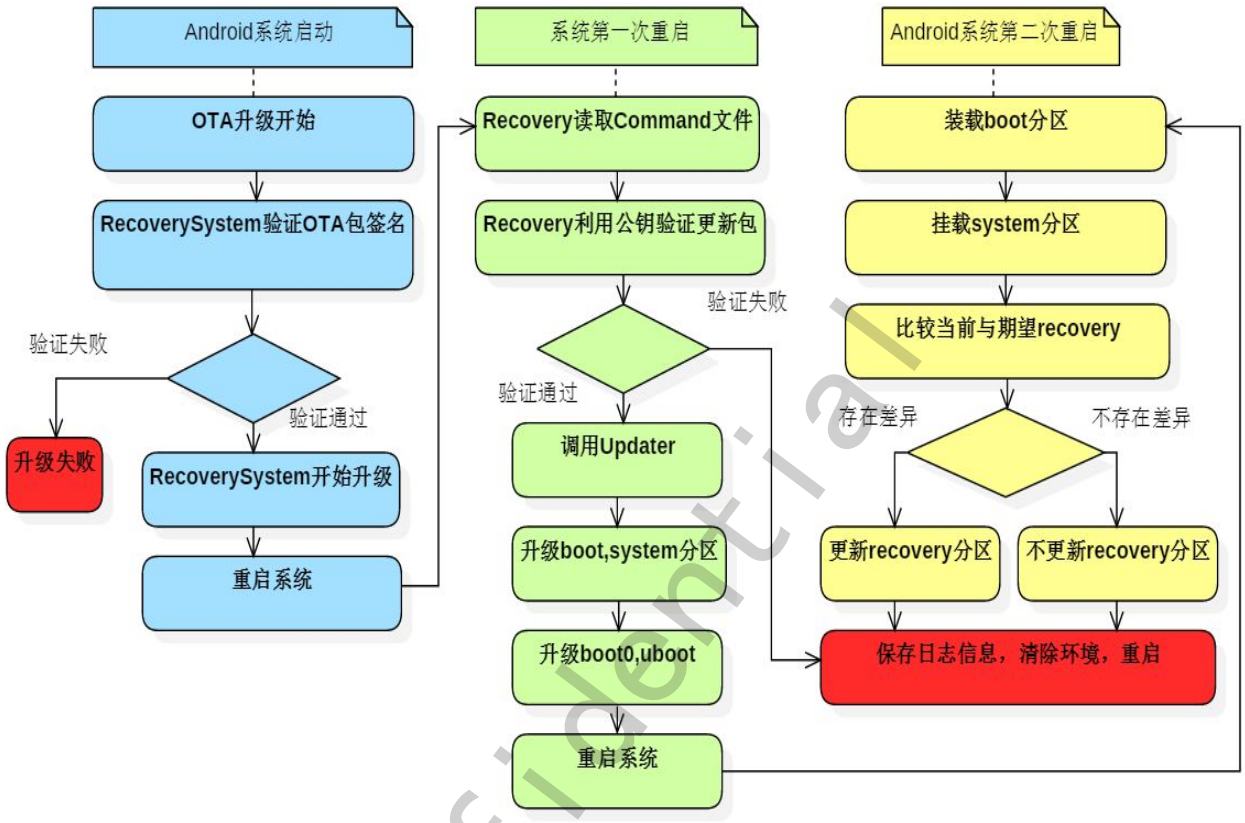

图 2: OTA 升级流程图

3. OTA 模块使用说明

3.1 OTA 的升级范围

原生 Android 提供的 Recovery 升级程序只支持更新 system 分区、recovery 分区及 boot 分区。除此之外，我们根据产品特点，给 Recovery 扩展了一些专有功能，以满足 BSP 的更新需要。

分区类型	是否支持	是否原生升级内容
Boot 分区更新	√	√
System 分区更新	√	√
Recovery 分区更新	√	√
Env 分区更新	√	×
Bootloader 分区更新	√	×
Boot0/Uboot 升级	√	×
sys_config.fex 更新	√	×
sys_partition.fex 更新	×	×

值得注意的是，BSP 中的大部分关于模块的配置都集中在 sys_config.fex 中，如果需要更新 sys_config.fex 的配置，就必须通过更新 uboot。在制作更新包时，要想新的 sys_config.fex 生效，务必记得在执行 make 命令之前先执行 get_uboot 确保当前的 sys_config.fex 配置被打到 uboot 中。如果希望自己实现制作 ota 包的脚本，可以参考以下目录的脚本文件：android/device/softwinner/common/vendorsetup.sh

3.2 制作 OTA 包步骤

使用 OTA 前首先需要区分三个包：

- TargetFile: 包含制作时当前编译版本的 system 分区，boot 分区，recovery 分区等内容，可用于制作 OTA 完整包和差分包。

- OTA 完整包：包含本次升级版本的所有内容，可以从之前各个版本直接升级到当前的版本。制作完整包需要当前版本的 TargetFile。
- OTA 差分包：包含本次升级版本和之前特定一个版本的升级内容，只适用于之前特定一个版本升级到当前版本。制作差分包需要之前特定版本的 TargetFile 和当前版本的 TargetFile。

3.2.1 制作不带签名的 OTA 包的步骤

3.2.1.1 制作不带签名的 TargetFile

在编译完 Android 后，输入以下命令：

```
$get_uboot
$make target-files-package
```

get_uboot 命令作用是从 lichee 目录中复制必要的更新文件到更新包中。不执行该命令会导致后面的 make 动作报错。最后得到 TargetFile 的路径 (device 表示编译的方案，time 表示当天的日期)：out/ target/ product/[device1]/ obj/ PACKAGING/ target_files_intermediates/[device2]-target_files-[time].zip (其中 device1 = $tdevice, device2 = $TARGET_PRODUCT，可以 echo $tdevice 可以看 device1, echo $TARGET_PRODUCT 可看 device2)

3.2.1.2 制作不带签名的 OTA 完整包

```
$get_uboot
$make otapackage -j8
```

get_uboot 命令作用是从 lichee 目录中复制必要的更新文件到更新包中。不执行该命令会导致后面的 make 动作报错。得到的 OTA 完整包的路径 (device 表示编译的方案，time 表示当天的日期)：out/target/product/[device1]/[device2]-ota-[time].zip (其中 device1 = $tdevice, device2 = $TARGET_PRODUCT，可以 echo $tdevice 可以看 device1, echo $TARGET_PRODUCT 可看 device2)

3.2.1.3 制作不带签名的 OTA 差分包

要生成差分包，必须获得前一版本的 target-file 文件，具体参考前面如何制作 TargetFile。将要升级版本 (旧版) 的 target-file 文件拷贝到 Android 的根目录下，并重命名为 old_target_files.zip。保证 Android 的根目录下只有一个 *.zip 的文件。之后执行以下命名将可以生成差分包：

```
$make otapackage_inc
```

得到的 OTA 差分包的路径 (device 表示编译的方案，time 表示当天的日期）out/target/ product/[device1]/[device2]- ota- [time]- inc.zip (其中 device1 = $tdevice, device2 = $TARGET_PRODUCT，可以 echo $tdevice 可以看 device1, echo $TARGET_PRODUCT 可看 device2)　　注意：

1. 该差分包仅对指定的前一版本固件有效。
2. 制作一个完整包，也会生成当前版本的一个 target-file 文件包。

3.2.2 制作带签名的 OTA 升级包

签名可以让 Android 带有厂家私有的签名，在 OTA 升级的时候会预先校验签名，如果签名不匹配则无法进行 OTA 升级，签名校验保证了 OTA 包的来源合法性，OTA 包的完成性（没有被第三方修改过）。制作带签名的 OTA 升级包的流程如下：

1. 生成没有签名的 TargetFile

参考制作不带签名的 OTA 包生成 TargetFile 的方法。

2.TargetFile 签名

```
$./build/tools/releasetools/sign_target_files_apks -d [key_path] -o
[unsigned_target_file.zip] [signed_target_file.zip]
```

[key_path] 为存放 key 文件夹的路径，需要包括 4 个 key 分别是 media，platform，releasekey，shared，具体包含以下文件：media.pem，media.x509.pem，platform.pk8，

releasekey.pem，releasekey.x509.pem，shared.pk8，media.pk8，platform.pem，platform.x509.pem，releasekey.pk8，shared.pem ，shared.x509.pem -o 表示替换一个公用的 ota_key，这个 key 确保第三方的 ota 包没有办法升级 [unsigned_target_file.zip] 表示上一步生成的没有签名的 TargetFile [signed_target_file.zip] 表示命令输出得到的经过签名的 TargetFile 签名的时候需要按照提示输入相应的证书的密码。

3. 从签名过的 TargetFile 得到镜像（boot.img,system.img 和 recovery.img)

```
$ ./build/tools/releasetools/img_from_target_files
  [signed_target_file.zip]  [img.zip]
```

[signed_target_file.zip] 表示经过签名的 TargetFile [img.zip] 表示命令输出得到的镜像压缩包

4. 解压 img.zip, 得到的 boot.img,system.img 和 recovery.img 复制到 target/product/[device1]/下面，重新 pack 得到可烧录的固件，是签名过的固件。

5. 生成 ota 包完整包

```
$./build/tools/releasetools/ota_from_target_files -k
  [releaseKey] [signed_target_file.zip] [ota_full.zip]
```

[releaseKey] 和 sign_target_files_apks 用来签名的 release key 相对应 [signed_target_file.zip] 表示经过签名的 TargetFile [ota_full.zip] 表示命令输出得到的 OTA 完整包

6. 生成 ota 包差分包

```
$./build/tools/releasetools/ota_from_target_files -k [releaseKey] -i
  [signed_target_file_v1.zip] [signed_target_file_v2.zip] ota_inc.zip
```

[signed_target_file_v2.zip] [ota_inc.zip] [releaseKey] 和 sign_target_files_apks 用来签名的 release key 相对应 [signed_target_file_v1.zip] 表示经过签名的版本 v1 的 TargetFile [signed_target_file_v2.zip] 表示经过签名的版本 v2 的 TargetFile [ota_inc.zip] 表示命令输出得到的 OTA 完整包

3.3 OTA 扩展功能

3.3.1 支持外部储存读取更新包

Android 原生的 OTA 升级包是放在/cache 分区的，但是随着版本的迭代，有可能出现 cache 分区不足以容纳 OTA 升级包的状况。针对这种情况，新版本的 Recovery 支持软连接形式，从 U 盘、SD 卡直接读取更新包，不用再把更新包复制到 cache 分区中，从而减少升级时间。

该功能都封装在 adnroid/frameworks/base/swextend/os/java/softwinner/os/RecoverySystemEx.java 中，调用该类的静态方法 installPackageEx() 方法，该方法需要传入更新包的路径和应用 Context 实例。

3.3.2 Usb-Recovery 功能

Usb-recovery 模式达到让用户即使不进入 Android 系统，也能够安装指定更新包的目的，让用户在系统异常无法进入系统的情况下，安装更新包恢复系统，给用户一条还原系统的通道。

Usb-recovery 模式是指将更新包改名为 update.zip，然后放到一个 u 盘的根目录上，插入 u 盘到小机中，按着小机的 Usb-recovery 按键进入 Usb-recovery。进行 Usb-recovery 模式之后，recovery 会自动搜索并安装 u 盘上的 update.zip 包。

Usb-recovery，即可通过 U 盘升级 OTA 包，有两种方法：

方法 (1) 在" 设置"-->" 备份和重置"-->" 系统恢复/升级"--> 选择需要升级的 OTA 包；
方法 (2) 按住机器"recovery 键"，上电进入 recovery 升级；

必须特别注意的是升级包必须命名为 update.zip，且只能放在该分区的根目录。具体配置如下：打开 Usb-Recovery 功能需要修改方案的 sys_config.fex 文件里相关配置，sys_config.fex 里面配置旁边也有相关说明，如下：


```
;-----------------------------------------------------------------------
; used: 模块使能端    1：开启模块  0：关闭模块
; mode: 模式选择      1：一键进入OTA升级    2：一键恢复（通过sysrecovery分区来恢复）  其他值：无效
; recovery_key：  按键配置  （例如：recovery_key= port:PH16<0><default>）
;-----------------------------------------------------------------------
[recovery_para]
used = 1
mode = 1
recovery_key = port:PH16<0><default><default><default>
```


4. FAQ

4.1 升级注意事项

4.1.1 OTA 不能改变分区数目及其大小

Recovery 只是一个运行在 Linux 上的一个普通应用程序，它并没有能力对现有分区表进行调整，所以第一次量产时就要将分区的数目和大小确定清楚，杜绝后续升级调整分区数目及其大小的想法，OTA 不能改变分区数目和分区的大小。

4.1.2 cache 分区的大小确定

原生 Recovery 机制中，因为 Recovery 内的分区挂载路径与 Android 的分区挂载路径并不完全相同，所以在 Android 上层传入更新包地址时，必须要保证这个包路径在 Recovery 和 Android 系统都是相同的。

能够读写的分区中只有 cache 分区和 data 分区会被 Recovery 和 Android 系统同时挂载，这意味着需要将包放这两个分区中，Recovery 才能识别。所以 Google 原生策略中，当在外部储存选择一个升级包时，都默认复制到 cache 分区中。所以在划分分区时需要注意要分配 cache 分区足够大的空间，否则可能出现无法容纳更新包而导致无法升级的问题。

4.1.3 misc 分区需要有足够的权限被读写

misc 分区 Recovery 与 Android 之间的桥梁，如果 misc 分区的读写权限过高，会导致上层应用无法对其写入数据，则会令 Recovery 功能异常。检验此功能存在问题时，请确保 misc 分区的设备节点/dev/block/xxx 和其软链接/dev/block/by-name/misc 有足够的权限被读写。比如，petrel-p1 方案的 nandg 的 group 权限为 system。


```
root@android:/dev/block /by-name# ls -l
lrwxrwxrwx root     root   2000-01-02 07:16 misc -> /dev/block/mmcblk0p8(misc分区软链接)
...
root@android:/dev/block # ls -l
brw-rw---- root     system    93,  48 2000-01-02 07:16 mmcblk0p8
.....
```

4.2 制作 OTA 包常见问题和注意事项

1. 执行签名命令的时候./build/tools/releasetools/sign_target_files_apks 出现类似的提示：ERROR: no key specified for:xxxx.apk Use '-e <apkname>=' to specify a key (which may be an empty string to not sign this apk). 这种情况通常是定义这个 apk 的 Android.mk 文件编写不规范导致，解决方法修改这个 apk Android.mk 文件，或者按照提示使用 -e 参数，例如 -e xxx.apk= 表示不对这个 apk 重新签名，或者 -e xxx.apk=[pathToKey]/[key]，[pathToKey] 是存放 key 的文件夹，[key] 根据实际情况选择 media，platform，releasekey 和 shared 其中一个

2. 如何生成 key? android/build/tools/mkkey.sh 脚本，打开脚本看到下面的一句定义： AUTH='/C=CN/ST=GuangDong/L=ZhuHai/O=Allwinner/OU=Allwinner/CN=China/emailAddress=allwinnertech@com' 请按照实际需求修改，然后执行./mkkey.sh media 按照提示输入这个 key 的密码然后就会在此目录下生成:media.pem media.pk8 media.x509.pem 三个文件。

3.TargetFile 和固件是否匹配的区分在 Android 设备执行 adb pull /system/build.prop 会得到这个固件的 build.prop 文件对于 TargetFile，解压出来查看 SYSTEM/build.prop，对比这两个 build.prop 如果一致，表示这个固件和 TargetFile 是匹配的.

4. 差分包升级失败在 Android 5.0 后，直接使用 make 和 pack 出来的固件直接使用差分包升级失败，原因是差分包是基于新旧 targetFile 对比生成，targetFile 里面 system 分区比直接 make 编译出来的 system 分区多了一些内容，固件生成需要从 targetFile 生成，操作流程如下：


```
$ source ./build/envsetup.sh
$ lunch [product]
$extract-bsp
$make -j8
$make_ota_target_file
# sign the target file if need be
$build/tools/releasetools/img_from_target_files [target_file.zip]  img.zip
$unzip -o img.zip -d $OUT/
$rm img.zip
$pack                              #安全方案使用pack -s -f
```

此外，在升级差分包时，会检验经过修改的 apk 或代码部分。解压缩差分包后，升级脚本的位置在 (/META-INF/com/google/android/updater-script) 目录下，对于修改过的部分，差分升级会首先检查升级前版本的 SHA 值以及升级后的 SHA 值，只有匹配后，才能够升级成功。因此客户可以判断升级前的文件或 APK 的 SHA 值是否准确，从而判断差分升级前的固件与升级差分包是否匹配。以某文件的差分升级为例子，在 updater-script 脚本中的形式如下：

```
apply_patch_check([文件名],[升级后SHA值]，[升级前SHA值]) || abort("...");
```

由此可以使用命令 shasum 对差分升级的文件进行 SHA 查询。差分升级需要使用升级前的 targetfile 以及升级后的 targetfile，如升级文件 system/bin/A.apk, 那么可以解压升级前后的 targetfile, 并查看 system/bin/A.apk 的值与升级脚本的 SHA 值是否匹配，假如匹配，则能够保证该文件差分升级成功。否则该文件不能差分升级成功，其原因归根到底是升级前的固件与生成差分包的升级前的 targetfile 不对应。

5. Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner. The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.tates nor implies warranty of any kind, including fitness for any particular application.

