



## Application Note

Application Note

Customer Support

Android OTA A/B System updates

Doc No: Android OTA DS6000-BD1A-DMT-

Version: V1.3EN update

Release date: V1.3

Classification: 2017-07-16  
Internal

© 2008 -- 2009 MediaTek Inc.

This document contains information that is proprietary to MediaTek Inc.

Unauthorized reproduction or disclosure of this information in whole or in part is strictly prohibited.

Specifications are subject to change without notice.

---

Android OTA A/B System updates  
Application Note

---

**MediaTek Inc.**

---

Postal address

No. 1, Dusing 1st Rd. , Hsinchu Science  
Park, Hsinchu City, Taiwan 30078

---

MTK support office address

No. 1, Dusing 1st Rd. , Hsinchu Science  
Park, Hsinchu City, Taiwan 30078

---

Internet

<http://www.mediatek.com/>

---



Document Revision History

Revision	Date	Author	Description
V1.0	2017-11-30	Haohsiang	Initial Release
V1.1	2018-01-03	Haohsiang	Add 6.9 A/B system application porting
V1.2	2018-01-04	Haohsiang	Modify command of generate RSA key (5.1.1)
V1.3	2018-07-16	Haohsiang	Add disable build cache and kernel-4.9 to chapter 5

MediaTek Confidential

© 2017 - 2018 MediaTek Inc.

Classification: Internal

This document contains information that is proprietary to MediaTek Inc.  
Unauthorized reproduction or disclosure of this information in whole or in part is strictly prohibited.

## Table of Contents

<b>Document Revision History.....</b>	<b>3</b>
<b>Table of Contents.....</b>	<b>4</b>
<b>Lists of Tables .....</b>	<b>6</b>
<b>Lists of Figures .....</b>	<b>7</b>
<b>1 Introduction .....</b>	<b>8</b>
1.1 Purpose .....	8
1.2 Who Should Read This Document.....	8
1.3 How to Use This Manual .....	8
1.3.1 Terms and Conventions.....	9
<b>2 References.....</b>	<b>10</b>
<b>3 OTA update Definitions .....</b>	<b>11</b>
<b>4 Abbreviations .....</b>	<b>12</b>
<b>5 Enable A/B System updates.....</b>	<b>13</b>
5.1.1 kernel-3.18 .....	13
5.1.2 kernel-4.4 .....	14
<b>6 Architecher Overview .....</b>	<b>16</b>
6.1 [Preloader boot control flow] .....	17
6.2 [LK boot control flow] .....	18
6.3 [A/B System partition layout].....	19
6.4 [Ramdisk location in A/B System updates] .....	19
6.5 [Boot Control HAL and HIDL].....	20
6.6 [DM-verity for A/B system] .....	21
<b>7 Generate OTA package Guideline .....</b>	<b>24</b>
7.1 Generate OTA full update package step by step.....	24
7.1.1 Full build project.....	24
7.1.2 Build OTA full update package for zip format .....	24
7.1.3 Full update package is in the out folder .....	24
7.2 Generate OTA incremental update package step by step .....	24
7.2.1 Build OTA incremental update package for zip format .....	24

<b>8</b>	<b>A/B System updates limitations .....</b>	<b>25</b>
<b>9</b>	<b>How to uses and debug A/B System updates.....</b>	<b>26</b>
9.1	How to use A/B updates ? .....	26
9.2	How to debug A/B updates ? .....	26
<b>10</b>	<b>Frequently Asked Questions .....</b>	<b>27</b>
10.1	FAQ - Does flashtool need to download both A and B image ? .....	27
10.2	FAQ –How to know what’s image running (A or B)? .....	27
10.3	FAQ – Could A partition loads B partition (ex:lk_a load boot_b) ? .....	27
10.4	FAQ – What’s partitions needs A/B ? .....	27
10.5	FAQ – What’s feature option for A/B system ? .....	27
10.6	FAQ – If handset ships without A/B , could handset becomes A/B through OTA update? .....	27
10.7	FAQ – What modules need porting for A/B ? .....	27
10.8	FAQ – How to add A/B partition for update ? .....	27



Lists of Tables

Table 1-2. Chapter Overview ..... 8

Table 1-3. Conventions ..... 9

Table 4-1. Abbreviations ..... 12



Lists of Figures

Figure 6.1-1 Preloader boot control flow..... 17

Figure 6.1-2 LK boot control flow..... 18

MediaTek Confidential

© 2017 - 2018 MediaTek Inc.

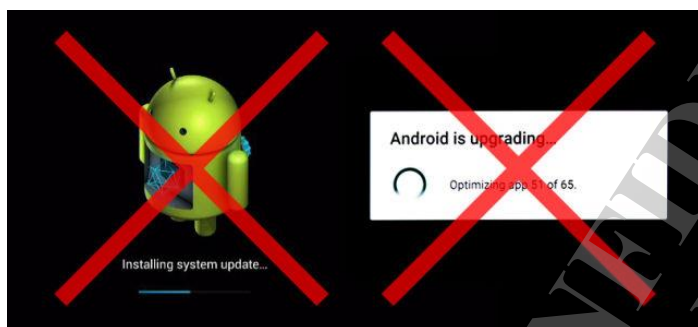
Classification: Internal

This document contains information that is proprietary to MediaTek Inc.  
Unauthorized reproduction or disclosure of this information in whole or in part is strictly prohibited.

## 1 Introduction

Introduction of A/B System updates as followings:

- It is a new OTA feature alias Seamless Updates .
- No more bricks (can fall back to previous system image).
- Ensures that a workable booting system remains on the disk.
- OTA updates applied in the normal mode background.



### 1.1 Purpose

This document provides the user guidelines for the A/B System updates. It describes how to generate the ota package on the Android platform. This manual also elaborates the mechanism required to enable A/B System updates (Seamless Updates) feature in MTK platform.

### 1.2 Who Should Read This Document

This document is primarily intended for:

- Engineers with technical knowledge of the OTA A/B System updates (Seamless Updates)

### 1.3 How to Use This Manual

This segment explains how information is distributed in this document, and presents some cues and examples to simplify finding and understanding information in this document. Table 1-1 presents an overview of the chapters and appendices in this document.

**Table 1-1. Chapter Overview**

#	Chapter	Contents
1	Introduction	Describes the scope and layout of this document.
2	References	Reference website or documents
3	OTA update definitions	OTA update definitions
4	Abbreviations	OTA Abbreviations
5	Enable A/B System updates	Enable A/B System updates in MTK platform
6	Architecher Overview	A/B System Updates Architecher Overview
7	Generate OTA package	Describe how to generate incremental and full ota package
8	A/B System limitations	A/B System updates limitations.
9	FAQ	Frequency Ask Questions






#	Chapter	Contents
10	How to uses and debug A/B	How to uses and debug A/B System updates

1.3.1 Terms and Conventions

This document uses special terms and typographical conventions to help you easily identify various information types in this document. These cues are designed to simply finding and understanding the information this document contains.

Table 1-2. Conventions

Convention	Usage	Example
[1]	Serial number of a document in the order of appearance in the References topic	Look up Chapter 2: System Architecture in [1]
	Important	

## 2 References

---

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- [1] MTK Company Profile, [http://brandclips.mediatek.inc/uploads/Company-profile-1H-2016\\_0418-Lite-final.pptx](http://brandclips.mediatek.inc/uploads/Company-profile-1H-2016_0418-Lite-final.pptx)
- [2] MTK Word Template, <http://brandclips.mediatek.inc/uploads/Microsoft-Office-Word-Oct-2014.rar>
- [3] The Android OTA Package Tools, <https://source.android.com/devices/tech/ota/tools.html>
- [4] A/B (Seamless) System updates, [https://source.android.com/devices/tech/ota/ab\\_updates](https://source.android.com/devices/tech/ota/ab_updates)
- [5] Implementing A/B Updates , [https://source.android.com/devices/tech/ota/ab\\_implement](https://source.android.com/devices/tech/ota/ab_implement)
- [6] Configuring ART, <https://source.android.com/devices/tech/dalvik/configure>

### 3 OTA update Definitions

---

For the purposes of the present document, the following terms and definitions apply:

This chapter is include from [3].

**Full updates:** A full update is one where the entire final state of the device (system, boot, and recovery partitions) is contained in the package. As long as the device is capable of receiving the package and booting the recovery system, **the package can install the desired build regardless of the current state of the device.**

**Incremental updates:** An incremental update contains a set of binary patches to be applied to the data already on the device. This can result in considerably smaller update packages:

Files that have not changed don't need to be included.

Files that have changed are often very similar to their previous versions, so the package need only contain an encoding of the differences between the two files.

☞ **You can install the incremental update package only on a device that has the old or source build used when constructing the package.**



## 4 Abbreviations

Please note the abbreviations and their explanations provided in **Error! Reference source not found..** They are sed in many fundamental definitions and explanations in this document and are specific to the information that this document contains.

**Table 4-1. Abbreviations**

Abbreviations	Explanation
MTK	MediaTek, Asia’s largest fabless IC design company.
OTA	Over-The-Air

## 5 Enable A/B System updates

☞ Enable **MTK\_AB\_OTA\_UPDATER=yes** option in sample project **k37mv1\_64\_ab** as followings

- **ProjectConfig.mk** : device/mediatek/k37mv1\_64\_ab/ProjectConfig.mk
- **Preloader:**

vendor/mediatek/proprietary/bootable/bootloader/preloader/custom/k37mv1\_64\_ab/k37mv1\_64\_ab.mk

- **LK:** vendor/mediatek/proprietary/bootable/bootloader/lk/project/k37mv1\_64\_ab.mk
- **Kernel:** arch/arm64/configs/k37mv1\_64\_ab\_defconfig

☞ Disable build cache partition

- **BoardConfig.mk:** device/mediatek/k37mv1\_64\_ab/BoardConfig.mk as followings  
ifneq (\$(strip \$(MTK\_AB\_OTA\_UPDATER)), yes)  
BOARD\_CACHEIMAGE\_FILE\_SYSTEM\_TYPE := ext4  
endif

### 5.1.1 kernel-3.18

Here is kernel configuration for dm-verity:

- CONFIG\_BLK\_DEV\_MD=y
- CONFIG\_MD\_LINEAR=y
- CONFIG\_DM\_ANDROID\_VERIFY=y
- CONFIG\_KEYS=y
- CONFIG\_ASYMMETRIC\_KEY\_TYPE=y
- CONFIG\_ASYMMETRIC\_PUBLIC\_KEY\_SUBTYPE=y
- CONFIG\_X509\_CERTIFICATE\_PARSER=y
- CONFIG\_PKCS7\_MESSAGE\_PARSER=y
- CONFIG\_PKCS7\_TEST\_KEY=y

Here is how you change key for dm-verity:

Put verity.x509 under alps/kernel-3.18. You should see alps/kernel-3.18/verity.x509, replace it with your own one.

- **Generate RSA key pair**
  - \$openssl genrsa -out prvk.pem 2048
- **Generate verity.pk8**
  - \$openssl pkcs8 -topk8 -inform PEM -outform DER -in prvk.pem -out verity.pk8 -nocrypt
- **Generate verity.x509.pem**
  - \$openssl req -new -x509 -key prvk.pem -out verity.x509.pem -sha256
- **Convert verity.x509.pem to verity.x509 (DER format)**
  - \$openssl x509 -in verity.x509.pem -outform der -out verity.x509

### 5.1.2 kernel-4.4

Here is kernel configuraiton for dm-verity:

- CONFIG\_KEYS=y
- CONFIG\_BLK\_DEV\_MD=y
- CONFIG\_MD\_LINEAR=y
- CONFIG\_DM\_ANDROID\_VERIFY=y
- CONFIG\_ASYMMETRIC\_KEY\_TYPE=y
- CONFIG\_ASYMMETRIC\_PUBLIC\_KEY\_SUBTYPE=y
- CONFIG\_X509\_CERTIFICATE\_PARSER=y
- CONFIG\_SYSTEM\_TRUSTED\_KEYRING=y
- CONFIG\_SYSTEM\_TRUSTED\_KEYS="certs/verity.x509.pem"

Here is how you change key for dm-verity

Location of dm-verity public key can be set in kernel configuration. We set it to kernel-4.4/certs/verity.x509.pem by default with "CONFIG\_SYSTEM\_TRUSTED\_KEYS="certs/verity.x509.pem".

- **Generate RSA key pair**
  - \$openssl genrsa -out prvk.pem 2048
- **Generate verity.pk8**
  - \$openssl pkcs8 -topk8 -inform PEM -outform DER -in prvk.pem -out verity.pk8 -nocrypt
- **Generate verity.x509.pem**
  - \$openssl req -new -x509 -key prvk.pem -out verity.x509.pem -sha256

After key configuration is done, rebuild kernel and boot into kernel without dm-verity. You can get public key id by "cat /proc/keys," then you will get message like this:

```
angler:/# cat /proc/keys
1c8a217e I----- 1 perm 1f010000 0 0 asymmetri
Android: 7e4333f9bba00adfe0ede979e28ed1920492b40f: X509.RSA 0492b40f []
2d454e3e I----- 1 perm 1f030000 0 0 keyring
.system_keyring: 1/4
```

### 5.1.3 kernel-4.9

☞ A/B system must uses AVB2.0 with kernel-4.9 version

Please reference "Introduction to Android Verified Boot 2.0" document for setting AVB2.0

#### 5.1.4 Flash system\_other.img to system\_b partition

☞ Set **BOARD\_USES\_SYSTEM\_OTHER\_ODEX := true** to save rom size (Only user & userdebug work)

system\_other.img is generated and download to system\_b partition at first time

Example usage (in device-common.mk): (This section is include from [6].)

```
PRODUCT_PACKAGES += \
```

```
  cppreopts.sh
```

```
PRODUCT_PROPERTY_OVERRIDES += \
```

```
  ro.cp_system_other_odex=1
```

```
BOARD_USES_SYSTEM_OTHER_ODEX := true
```

## 6 Architecher Overview

---

This chapter first gives a brief description of the modules of the system and the relationship of the modules.

1. Preloader boot control flow
2. LK boot control flow
3. A/B System partition layout
4. Ramdisk location in A/B System updates
5. Boot control HAL and HIDL
6. DM-verity for A/B System updates
7. Fastboot for A/B System updates
8. A/B system update process



### 6.1 [Preloader boot control flow]

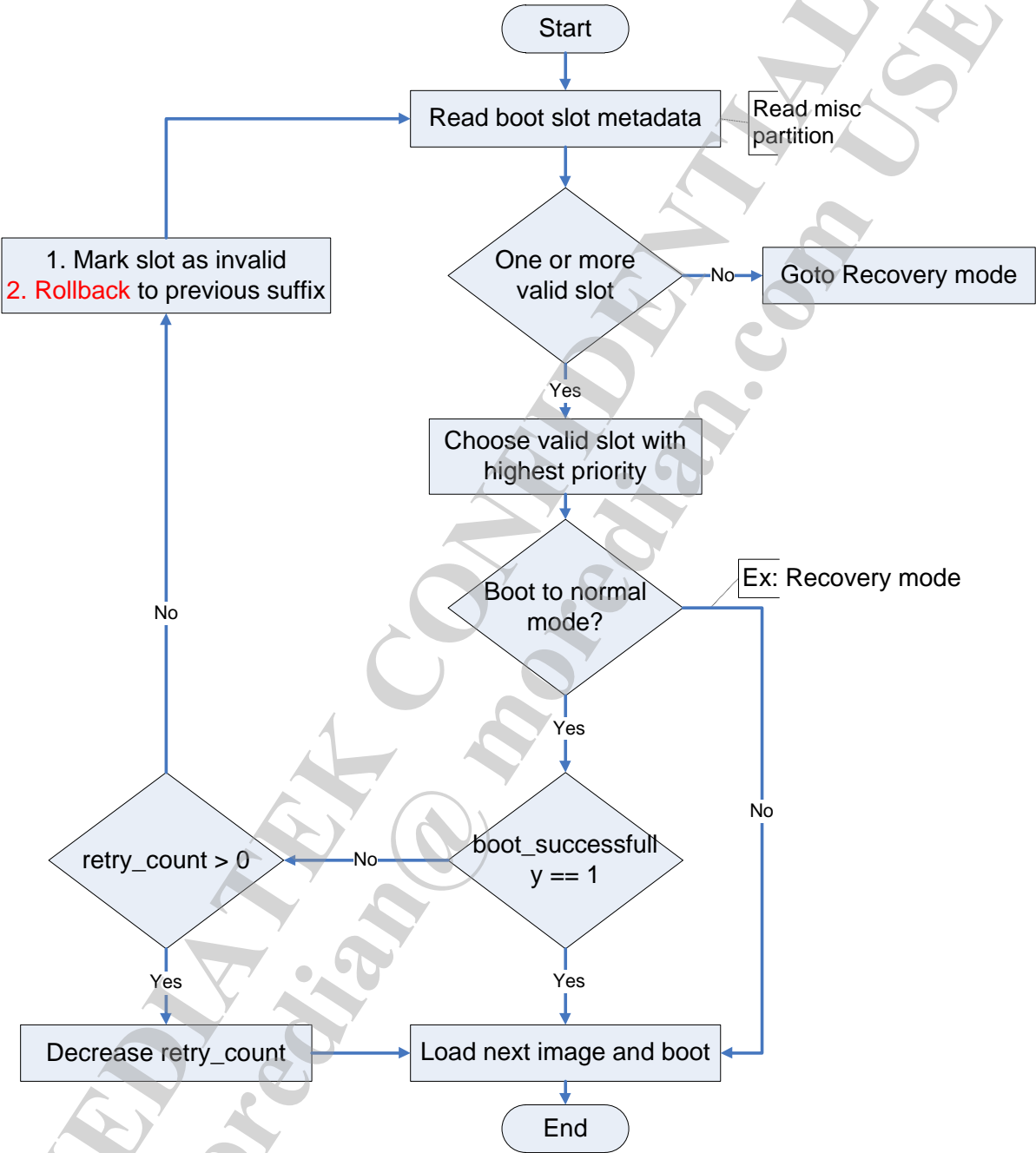


Figure 6.1-1 Preloader boot control flow.

6.2 [LK boot control flow]

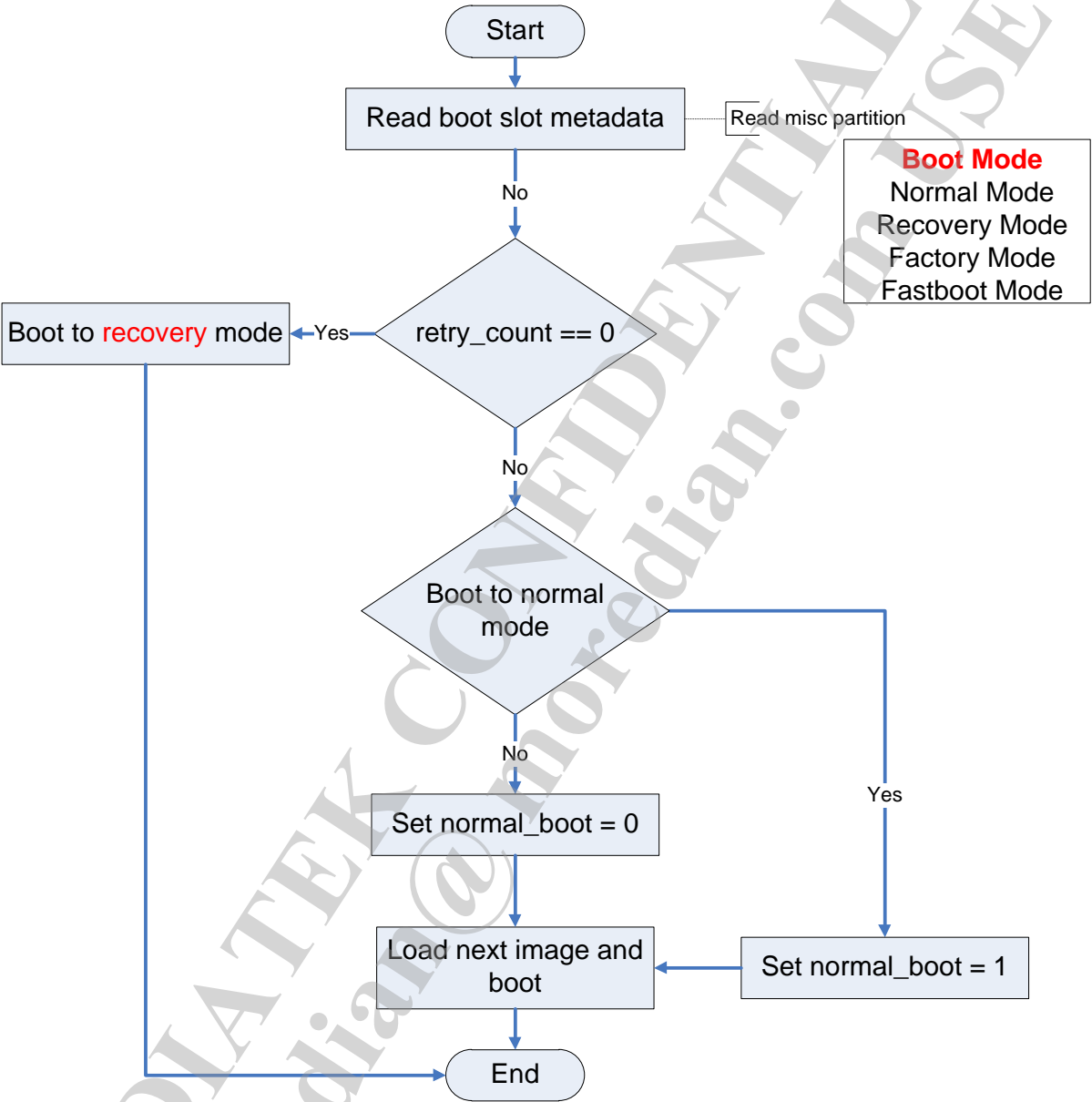


Figure 6.1-2 LK boot control flow.

## 6.3 [A/B System partition layout]

Who needs double partition ?

Partition which needs OTA update

A/B doesn't need **recovery** and **cache** partition

**Original**

Partition_Name
preloader (boot1 and boot2)
lk
lk2
md1img
md1dsp
md1arm7
md3img
nvrnm
boot
<b>recovery</b>
logo
tee1
tee2
system
<b>cache</b>
nvrnm
userdata

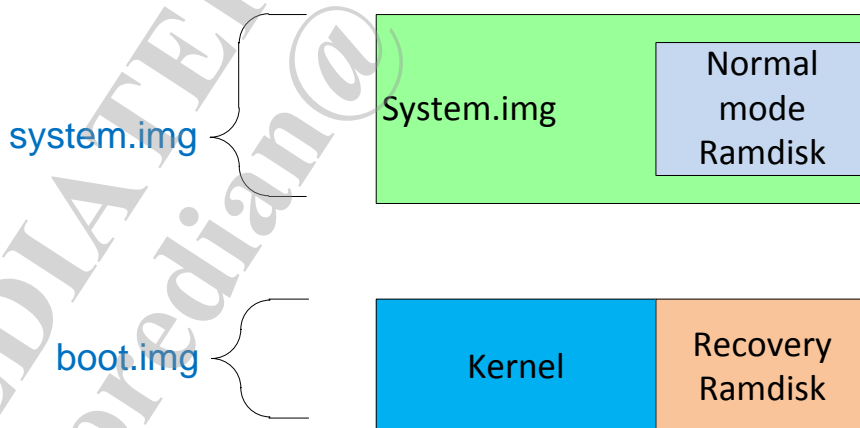
**A/B System**

A/B Partition_Name
preloader (boot1 and boot2)
lk_a
md1img_a
md1dsp_a
md1arm7_a
md3img_a
boot_a
system_a
tee_a
lk_b
md1img_b
md1dsp_b
md1arm7_b
md3img_b
boot_b
system_b
tee_b
nvrnm
userdata
logo

## 6.4 [Ramdisk location in A/B System updates]

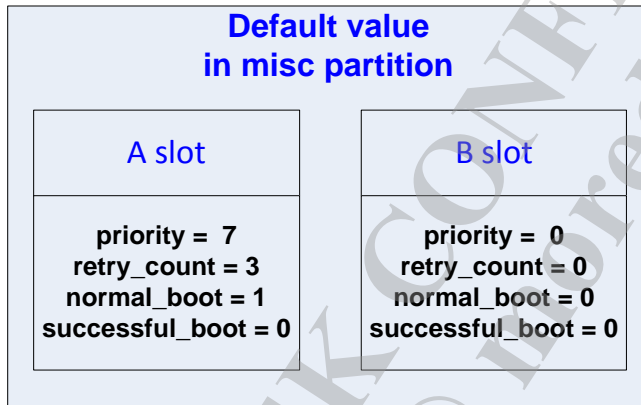
☞ Boot.img is recovery.img

☞ The ramdisk of normal mode is in system.img

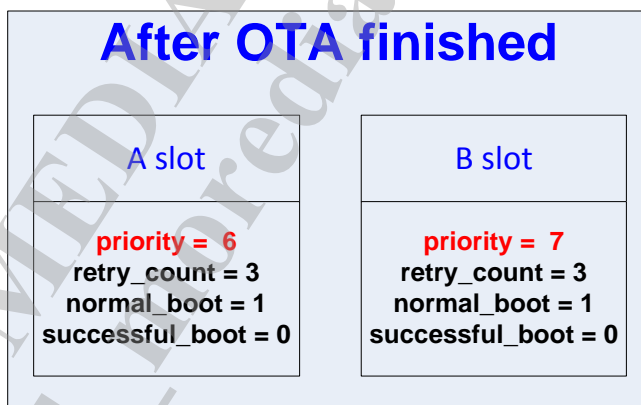


## 6.5 [Boot Control HAL and HIDL]

- ☞ MTK uses **AOSP default boot control HIDL** interface to access boot control HAL
  - Source code path is hardware/interfaces/boot/
- ☞ Boot control HAL for update\_engine
  - Source code path is vendor/mediatek/proprietary/hardware/bootctrl/
  - Library **bootctrl.platform.so** in vendor partition
- ☞ Preloader (PL) and LK also have boot control API
  - **PL:** vendor/mediatek/proprietary/bootable/bootloader/preloader/platform/common/bootctrl
  - **LK:** vendor/mediatek/proprietary/bootable/bootloader/lk/platform/common/bootctrl
- ☞ Preloader will set **default value** of boot control structure when handset boot up at first time.



- ☞ After handset OTA update firstly , it will boot from B
- ☞ Both A and B can boot up , but B has **higher priority** than A



## 6.6 [DM-verity for A/B system]

☞ DM-verity is changed when A/B system updates is enabled.

Here we assume boot image format and concept of ramdisk are known by reader. In definition of verified boot, every software/data must be verified before it's used to ensure authenticity and integrity of software/data so entire device could work as expected by device manufacturer. Previously, public key used to verify dm-verity metadata is embedded in ramdisk of boot image, which is verified by bootloader at boot time. Note that the public key for dm-verity is not embedded in recovery image ramdisk since "system" partition is mounted in recovery mode.

Story changes when A/B system comes into play. When A/B system is enabled, number of partition increases tremendously since every A/B system supported partition becomes two partitions. For example, there's only "system" partition when A/B system is not enabled, but there will be "system\_a" and "system\_b" partitions when A/B system is enabled. To reduce number of partitions, recovery partition is removed. Ramdisk inside boot image is only used in recovery mode, hence no public key inside it. The original boot ramdisk is merged into system image and the merged system image becomes the new "rootfs." Here comes the problem, if we put public key used to verify dm-verity into rootfs(the merged system image) as before, how do we avoid using content of system image before we setup dm-verity for it?

The answer is to enable kernel keyring feature and build public key into kernel image. You can imagine key ring as a database for keys in kernel, which means the keys are built into kernel, not in ramdisk any more. To let kernel know which key we're going to use, bootloader passes key id to kernel through kernel cmdline.

## 6.7 [Fastboot for A/B system]

☞ MTK implement fastboot commands and variables for A/B system

- Source code : vendor/mediatek/proprietary/bootable/bootloader/lk/app/mt\_boot/fastboot.c
  - Support command : set\_active <slot suffix> sets the current active slot to the given suffix.
  - Support variables:
    - has-slot:<partition base name without any suffix appended>** - returns "yes" if the given partition supports slots, "no" otherwise.
    - current-slot** returns the slot suffix that will be booted from next.
    - slot-count** returns an integer representing the number of available slots. Currently, two slots are supported so this value is 2.
    - slot-successful:<slot suffix>** returns "yes" if the given slot has been marked as successfully booting, "no" otherwise.
    - slot-unbootable:<slot suffix>** returns "yes" if the given slot is marked as unbootable, "no" otherwise.
    - slot-retry-count:<slot suffix>** number of retries remaining to attempt to boot the given slot.
- These variables should all appear under the following: **fastboot getvar all**

## 6.8 [A/B system updates process]

### ☞ Update\_engine:

- executed by update\_engine.rc
- Only responsible for update partition like role of recovery and updater
- Uses payload.bin which is in update.zip to update each partition
- Runs in little core and background
- Call boot\_control HAL/HIDL interface
- Run a postinstall program from the new partition after writing all the unused slot partitions

Update\_engine should **NOT** do:

- Modify the partition table (Layout changed)
- Modify the contents of partitions in the current slot

### ☞ Update\_engine\_client:

- Update\_engine\_client is for debug only , this process could get update status

### ☞ Update\_verifier:

- Update\_verifier is executed by update\_verifier.rc
- If product support verity , check verity mode is enforcing mode or not
- Mark **boot successful flag** when phone boot up

## 6.9 [A/B system application layer porting]

☞ **Make a callback in your app which is used to receive notification from update\_engine about update status and completion**

- Source code path is frameworks/base/core/java/android/os/**UpdateEngine.java**.
- Extend **UpdateEngineCallback** and overwrite **onStatusUpdate** and **onPayloadApplicationComplete** methods.
  - onStatusUpdate is used to get status of update like update percentage, or error code
  - onPayloadApplicationComplete is called when update is finished and error code is returned .
  - Complete list of error codes can be found in file /system/update\_engine/common/error\_code.h

☞ **Call applyUpdate from app**

**public void applyPayload(String url, long offset, long size, String[] headerKeyValuePairs)**

- **url:** Path of update package(Payload). AB system update need path of internal storage only. It must be at /data/ota\_package/xxx and should be start with file://. An Example will be like : file:///data/ota\_package/update.zip
  - Update Engine only accept phone internal storage path (data/ota\_package/update.zip). Any other path or sd card can't be used.
  - Calling App need to get selinux permission to access this path based on operations used on this path. An example is like:  
allow your\_app ota\_package\_file:dir { read open write create remove\_name search rename add\_name getattr };  
allow your\_app ota\_package\_file:file { read write create open rename getattr unlink };
- **offset:** It is payload offset inside update.zip. It need to be calculated from update zip file.
- **Size:** It is size of payload file and can be found from payload\_properties.txt in update package.
- **headerKeyValuePairs:** It is metadata. This can also be found from payload\_properties.txt in update package
  - The headerKeyValuePairs parameter is used to pass metadata to update\_engine. In Google's implementation, this is stored as payload\_properties.txt in the zip file. It's generated by the script { system/update\_engine/scripts/brillo\_update\_payload}.
  - Example : String[] pairs = {  
"FILE\_HASH=IURPCIKIAjtMOyB/EjQcl8zDzqtD6Ta3tJef6G/+z2k=",  
"FILE\_SIZE=871903868",  
"METADATA\_HASH=tBvj43QOB0Jn++JojcpVdbRLz0qdAuL+uTkSy7hokaw=",  
"METADATA\_SIZE=70604"  
};

## 7 Generate OTA package Guideline

This chapter gives a description of a group of generating A/B System ota package guideline.

### 7.1 Generate OTA full update package step by step

#### 7.1.1 Full build project

Command : `make -j8 -k 2>&1 | tee normal_build.log`

#### 7.1.2 Build OTA full update package for zip format

Command : `make -j8 otapackage -k 2>&1 | tee build_ota.log`

#### 7.1.3 Full update package is in the out folder

Example full update package : `full_project_name-ota-1482807938.zip`

### 7.2 Generate OTA incremental update package step by step

#### 7.2.1 Build OTA incremental update package for zip format

☞ **Example command:** `./build/tools/releasetools/ota_from_target_files`

`-k ./device/mediatek/security/releasekey`

`-s vendor/mediatek/proprietary/scripts/releasetools/mt_ota_from_target_files`

`-i V1.zip V2.zip update.zip`

Note:

1. V1.zip and V2.zip is from `obj/PACKAGING/target_files_intermediates/` in out folder.



## 8 A/B System updates limitations

---

This chapter describes A/B system updates limitations .

This chapter is include from [4].

- ☞ Modify partition table , no support partition layout changed
- ☞ Modify the contents of partitions in the current slot
- ☞ Modify the contents of nonA/B partitions that can't be wiped with a factoryreset

## 9 How to uses and debug A/B System updates

### 9.1 How to use A/B updates ?

#### ■ Normal mode:

- PC must install python
- Put update\_device.py (system/update\_engine/scripts/update\_device.py) to adb folder
- Connect phone with PC via usb cable
- python update\_device.py update.zip in adb window

```
D:\Adb>python update_device.py full_k53v1_64_ab-ota-1502713905.zip
INFO:root:Running: adb reverse tcp:1234 tcp:51501
INFO:root:Running: adb shell su 0 update_engine_client --update --follow
HASH=KyKz6JU+K3FQwYimMurD13xU0WG02n3bESPb0t02sLM=
FILE_SIZE=592100123
METADATA_HASH=ytU6gL3hIdEzi9rnkWEeULUXnTK1HSxTzAoA/fAEzD0=
METADATA_SIZE=86389
USER_AGENT=Dalvik (something, something)
NETWORK_ID=0
```

#### ■ Recovery mode:

- The behavior is the same with non-A/B
- Apply update from sideload (Update via adb)
- Apply update from SD card (Update via update.zip in sdcard)

### 9.2 How to debug A/B updates ?

#### Log location

##### ■ Normal mode:

- mtklog\mobilelog\APLogxxx\main\_log
- Search **update\_engine** key word

##### ■ Recovery mode:

- Adb pull /tmp/recovery.log

#### Common Error code

- kInstallDeviceOpenError = 7 , Open update partition error
- kDownloadStateInitializationError = 20 , ValidateSourceHash function fail
- All error code is in system/update\_engine/common/error\_code.h

## 10 Frequently Asked Questions

### 10.1 FAQ - Does flashtool need to download both A and B image ?

- No, it will waste lots of time for downloading. After flashtool downloads image, handset only runs on A images.
- A image could OTA to B empty partition through incremental update package

### 10.2 FAQ –How to know what's image running (A or B)?

- Check property ro.boot.slot\_suffix (getprop ro.boot.slot\_suffix)
- Check androidboot.slot\_suffix in command line (/proc/cmdline)

### 10.3 FAQ – Could A partition loads B partition (ex:lk\_a load boot\_b) ?

- No, suffix must be the same . It's like dual system.

### 10.4 FAQ – What's partitions needs A/B ?

- If partition needs OTA update , this partition must have both of A and B

### 10.5 FAQ – What's feature option for A/B system ?

- MTK\_AB\_OTA\_UPDATER

### 10.6 FAQ – If handset ships without A/B , could handset becomes A/B through OTA update?

- No , even handset has A/B partition , because boot flow and verity (A/B uses android-verity) are different from non-AB

### 10.7 FAQ – What modules need porting for A/B ?

- Please check [5] Implementing A/B Updates , MTK already porting all of them done.

### 10.8 FAQ – How to add A/B partition for update ?

- Add partition name to AB\_OTA\_PARTITIONS which is in device/mediatek/common/device.mk

### 10.9 FAQ – Who set boot\_sucessfully flag ?

- Update\_verfier process will set boot\_sucessfully flag  
Source code is in bootable/recovery/update\_verifier/update\_verifier.cpp

## 10.10 FAQ – How to disable dm-verity on A/B system ?

- Modify vendor/mediatek/proprietary/bootable/bootloader/lk/app/mt\_boot/mt\_boot.c  
 snprintf(tmpbuf, TMPBUF\_SIZE,  
 - "rootwait ro init=/init root=**/dev/dm-0 dm=\**"system none ro,0 1 android-verity **PARTUUID=%s \**"  
 androidboot.slot\_suffix=%s",  
 + "rootwait ro init=/init root=**PARTUUID=%s \**" androidboot.slot\_suffix=%s",