



## **SigmaStar Ota Upgrade**

---



© 2020 SigmaStar Technology Corp. All rights reserved.

SigmaStar Technology makes no representations or warranties including, for example but not limited to, warranties of merchantability, fitness for a particular purpose, non-infringement of any intellectual property right or the accuracy or completeness of this document, and reserves the right to make changes without further notice to any products herein to improve reliability, function or design. No responsibility is assumed by SigmaStar Technology arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others.

SigmaStar is a trademark of SigmaStar Technology Corp. Other trademarks or names herein are only for identification purposes only and owned by their respective owners.



## REVISION HISTORY

Revision No.	Description	Date
0.1	<ul style="list-style-type: none"><li>{Initial release}</li></ul>	{02/27/2020}
0.2	<ul style="list-style-type: none"><li>{Upgrade ui}</li></ul>	{04/14/2020}



## TABLE OF CONTENTS

REVISION HISTORY .....	i
TABLE OF CONTENTS.....	ii
1. 概述.....	1
1.1. 概述.....	1
2. 分区打包流程.....	2
2.1. 打包工具介绍 .....	2
2.2. 一般的打包流程举例: .....	2
2.3. 在 ALKAID 中打包介绍: .....	3
2.4. 不带文件系统的 RAW DATA 打包 .....	3
2.5. UBIFS/Squashfs/Jffs2 打包 .....	3
2.6. 个别文件打包 .....	4
2.7. 差分数据打包 .....	4
2.8. 新旧文件夹扫描并批量文件打包.....	4
3. 分区更新流程.....	5
3.1. 升级流程 .....	5
3.2. 分区升级前注意 .....	5
3.3. 压缩的升级包更新.....	5
3.4. 非压缩的升级包更新 .....	5
3.5. 升级 UI 显示.....	5
3.6. 差分升级 .....	6
3.7. 升级状态获取 .....	6



## 1. 概述

---

### 1.1. 概述

本文介绍的是 Sigmastar 的 IPC、NVR 平台在 OTA 升级中打包、解包、升级的流程。仅支持分区打包、分区升级，个别文件升级，支持差分升级，支持新旧文件夹对比批量打包。

升级包后台服务器维护、终端下载及管理，请结合第三方服务。

## 2. 分区打包流程

---

### 2.1. 打包工具介绍

打包工具只会根据当前要升级的文件生成所需的文件头。

打包工具的 bin 所在目录：

project/image/makefiletools/bin/otapack

otapack 工具命令参数介绍：

- c --create: 创建一个空的升级包文件头，此命令创建文件头的同时可以用-b 或-e 添加开始或结束升级的脚本。
- b --begin-script: 在服务器上指定一个文件，用于板子在升级前执行的脚本。
- e --end-script: 在服务器上指定一个文件，用于板子在升级结束后执行的脚本。
- a --append: 在已经创建的升级包中追加新的头信息，追加新的信息中，必须使用以下参数告知升级的相关信息。
- s --src-file: 需要更新的源文件路径。
- d --dst-file: 需要更新的目标文件或者分区节点的路径。
- t --dst-file-size: 需要更新的目标文件或者分区的大小，单位为 byte，可以用 16/10 进制表示。
- m --file-mode: 你要更新的文件 mode。
- o --diff-old: 差分升级的旧文件、文件夹目录。
- n --diff-new: 差分升级的新文件、文件夹目录。
- block-update: 打包的数据为一个裸数据的 block 升级。
- ubi-update: 打包的数据为 UBI 文件系统的 volume 升级。
- file-update: 打包的数据为一个文件升级。
- file-add: 用打包的数据添加一个文件。
- file-delete: 仅更新打包的数据头，删除板子上的目标文件。
- file-update-diff: 打包的数据为差分升级的 diff data。
- dir-update: 扫描新旧两个文件夹中的内容，并比较差异、增加、减少的部分，然后进行批量打包。
- dir-update-diff: 扫描新旧两个文件夹中的内容，并比较差异、增加、减少的部分，然后进行批量打包。其中差异的文件对其差分数据进行打包。
- help: 打印帮助信息。
- debug: 打开调试信息。

otaunpack 工具命令参数介绍：

- x: 解包并升级压缩的文件
- r: 解包并升级非压缩的文件
- t: 指定一个可写的文件夹路径用于差分升级。

### 2.2. 一般的打包流程举例：

- 1、创建一个空的 header：  
otapack -c SStarOta.bin
- 2、若有需要可以用-b/-e 指令添加脚本。

```
otapack -c SStarOta.bin -b start.sh -e end.sh
```

- 3、根据需要打包的文件更新 header 数据：

```
otapack -a SStarOta.bin -s ./images/kernel -d /dev/mtdblock8 -t 0x500000 --block-update
```

- 4、循环执行 3 步骤，所有的文件进行打包。

- 5、压缩升级包：

```
gzip SStarOta.bin
```

## 2.3. 在 ALKAID 中打包介绍：

OTA 打包流程目前已经整合到了 Makefile 中。

当程序编译和打包完成后，在 project 下输入指令：

```
make image-ota
```

会出现如下交互界面，指定在板子上执行的脚本路径，可以选择添加或者不添加：

Start scripts:

End scripts:

在打包分区是，列出了打包的分区数据：

```
Make ipl ?(yes/no)
```

做出相关的选择后，会在 project/image/output/images/下产生 SStarOta.bin.gz

在配置 partition 的 config 文件中会配置分区进行打包。

举例：

文件 spinand.ubifs.p2.partition.config 中有变量：OTA\_IMAGE\_LIST

在此变量后面追加需要打包的分区名称，并在分区的配置中添加字段 xxx\$(OTABLK)，配置该分区需要升级的目标节点路径。

只有在 OTA\_IMAGE\_LIST 添加了分区名，才会在 make image-ota 的时候会询问该分区是否要进行 ota 升级。

分区打包脚本的所有逻辑在 image/ota.mk 中实现，有兴趣的可以自行研究。

## 2.4. 不带文件系统的 RAW DATA 打包

分区升级步骤是大同小异的，会把需要升级的文件填到对应的 mtblock 中，因此打包的流程也大体上是一样的。稍有不同的是在 RAW DATA 分区打包的要自行实现升级包头的创建和数据填充，这里以 spinand 的 IPL 分区打包举例：

```
define makeota
@read -p "Make $(1) ?(yes/no)" select; \
if [ "$${select}" == "yes" -o "$${select}" == "y" ]; then \
    $(foreach n,$(3),$(PROJ_ROOT)/image/makefiletools/bin/otabinheader -s $(2) -d $(n) -t $(4) -p $(5) -a \
    $(IMAGEDIR)/SStarOta.bin;cat $(2) >> $(IMAGEDIR)/SStarOta.bin;) \
fi;
endef

ipl_spinand_mkota_ :
    $(call
makeota,$(patsubst %_spinand_mkota_,%,$@),$(IMAGEDIR)/ipl_s.bin,$$(patsubst %_spinand_mkota_,%,$@)$(OTABLK) ,
$$(patsubst %_spinand_mkota_,%,$@)$(PATSIZE)),0)
```

## 2.5. UBIFS/Squashfs/Jiffs2 打包

如果新创建了这些分区，并且把它们加到 OTA\_IMAGE\_LIST 中，则无需再 ota.mk 中添加特殊处理，这一类的分区升级文件会统一处理。

UBIFS 的分区升级方式与其它两种格式稍有差异，因此在打包的时候 UBIFS 打包的时候请使用--ubi-update。

## 2.6. 个别文件打包

个别文件打包的选项是--file-update，目前 ALKAID 中还没有专门针对文件更新的打包，因此使用者需要自己手动添加。

## 2.7. 差分数据打包

对个别文件进行差分数据打包的选项是--file-update-diff，除了普通文件外，也可以对 kernel、uboot 等 raw image 进行差分，也可以对只读文件系统的镜像文件进行差分，但是不可以对可写的文件系统进行差分。

差分升级需要依赖于第三方开源的 bsdiff、bspatch。

bsdiff 工具在打包的时候会使用，bspatch 在板子上使用。

bsdiff、bspatch 工具的源码在 project/tools/bsdiff

编译方式：

```
tar -vxf bzip2-1.0.6.tar.gz
```

```
cd bzip2-1.0.6
```

```
make
```

编译完成后当前目录下可以看到执行档案 bsdiff、bspatch

默认编译出来的是 pc 上使用的，编译板子上使用 bspatch 请改 Makefile。

编译完成后的 bsdiff 要放到 pc 中的/usr/bin/路径下，或者环境变量\$ PATH 所指定的位置。

同理板子上 bspatch 也要放到对应的位置。

## 2.8. 新旧文件夹扫描并批量文件打包

文件夹扫描分两种，一种是扫描出的有差异的文件进行强制更新。

--dir-update

命令举例：

```
otapack -a ./test-diff.bin -o ./old/ -n ./new/ -d /config --dir-update
```

另外一种扫描出的有差异的文件进行差分升级。

--dir-update-diff

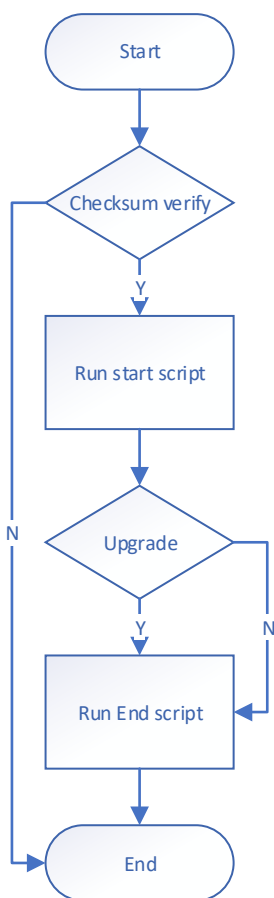
命令举例：

```
otapack -a ./test-diff.bin -o ./old/ -n ./new/ -d /config --dir-update-diff
```



## 3. 分区更新流程

### 3.1. 升级流程



### 3.2. 分区升级前注意

在分区升级之前请确保分区 **umount** 成功。可以在 **start scripts** 脚本中作相关的 **umount** 动作。若需更新文件，请确保文件所在可读写的文件系统中，并且有写权限。

otaupack 工具请找 FAE 获取。

### 3.3. 压缩的升级包更新

```
otaupack -x SStarOta.bin.gz
```

### 3.4. 非压缩的升级包更新

```
otaupack -r SStarOta.bin
```

### 3.5. 升级 UI 显示

升级时可指定一张全屏的背景图片贴到 **framebuffer** 上，图片支持 **jpg** 和 **png** 格式。同时在 **framebuffer** 中绘制进度条。



```
otaunpack -x SStarOta.bin.gz -p upgrade.jpg
```

### 3.6. 差分升级

若压缩包中存在差分升级的数据，需要指定一个临时的可以写的文件夹，命令：

```
otaupack -x SStarOta.bin.gz -t /tmp
```

### 3.7. 升级状态获取

在 END script 中利用内部变量'OTA\_STATUS'可以获取升级的状态。

若 OTA\_STATUS 为 0 则升级成功，否则为-1。

### 3.8. SPI-NAND 坏块处理

使用--block-update 命令打包升级的程序中会操作 mtdblock 节点，这种操作不会处理坏块，若需要升级同时处理坏块，请使用 mtd-utils 工具，或者在代码中自行处理坏块。