全志科技
Allwinner Technology

V40 项目

OTA 开发使用指南 V1.0

文档履历

版本号	日期	制/修订人	内容描述
V1.0	2016-06-08		正式版本

目 录

1. 概述

1.1. 编写目的

介绍 OTA 原理及客户端通用使用方法

1.2. 适用范围

V40 sdk

1.3. 相关人员

- Android 网络升级开发人员
- Android OTA 开发人员
- Android OTA 测试人员

2. OTA 简介

OTA 是 Google 用于升级 Android 设备的系统的一种技术。具体为 Android 设备通过网络或其他途径获取到 OTA 升级包（zip 压缩包），Android 设备根据 OTA 包进行系统升级。

OTA 升级包分为两种，一种叫做完整包，另外一种是差分包。完整包包含了整个系统的内容，可以忽略 Android 设备本身的系统内容进行系统升级；差分包包含了旧版本系统（版本 1）和新版本（版本 2）系统的差异信息，升级的时候需要检查 Android 设备的系统，如果系统不是版本 1，又或者被修改过，就会导致升级不成功。由于差分包是根据差异生成的，所以通常差分包会比完整包小很多。

不管是通过网络升级还是本地升级，都需要先制作 OTA 升级包，两者升级时的流程也是一致的，都会重启到 Recovery 中，然后再安装 OTA 包进行升级。

3. OTA 升级说明

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
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	√
Nand 方案 Boot0/Uboot 升级	√
TSD/EMMC 方案 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。

3.2. 升级注意事项

3.2.1. 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。

3.2.2. 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。

3.2.3. 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/misc 有足够的权限被读写。比如，magton-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
.....
```

3.3. 制作 OTA 包

使用 OTA 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。

3.3.1. 制作不带签名的 OTA 包的步骤

1.制作不带签名的 TargetFile

在编译完 Android 后，输入以下命令：

```
$make_ota_target_file
```

或者

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

make_ota_target_file 只是封装一些命令，具体可以查看 device/softwinner/common/vendorsetup.sh 里面关于 make_ota_target_file 的定义。

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)

2.制作不带签名的 OTA 完整包

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

或者

```
$make_ota_package
```

make_ota_package 只是封装一些命令，具体可以查看 device/softwinner/common/vendorsetup.sh 里面关于 make_ota_package 的定义。

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.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。

3.3.2. 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

参考制作没有待签名的 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 包完整包

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

3.3.3. 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 的 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.

3.4. OTA 扩展功能

3.4.1. 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。

4. OTA 升级应用简介

OTA 升级应用在系统中能够接收到服务器端的新版本消息推送，并能展示新版本信息，点击下载后可以开始下载升级包，用户确认升级后开始进入 Recovery 模式进行升级。同时 OTA 应用也能支持本地升级的模式。

OTA 升级应用前台由“设置”->“备份和重置”->“系统恢复/升级”选项进入，或通过 Launcher 主界面点击进入，支持本地升级和网络升级，网络升级使用艺果网络的后台系统，支持升级信息推送，CDN 下载加速。

OTA 应用主界面如下：

5. OTA 升级应用客户端配置

5.1.OTA 升级应用版本定义

OTA 升级系统主要使用到了如下的属性，这些属性都在方案的 mk 文件中定义，编译生成后位于系统的 system/build.prop 文件中

[ro.product.model]: [magton]

[ro.product.device]: [magton-p1]

[ro.product.rom.name]: [BoxRom]

[ro.product.rom.type]: [YB]

[ro.build.version.incremental]:[20160608]

[ro.product.brand]: [Allwinner]

使用到的这些属性解释如下：

字段名	是否为新增字段	说明
ro.product.model	Android 原生	机型名称，通常显示在原生设置->关于->型号
ro.product.device	Android 原生	设备名称，唯一标识一个设备的名称
ro.product.rom.name	新增	表示 ROM 的名称，服务端同时使用这个名称作为存放固件的根目录名，如果没有设置这个属性 Update APK 默认使用“BoxRom”
ro.product.rom.type	新增	说明版本类型，取值有，WD 、NB、YB 内部体验版用“NB”（内部 Beta）表示； 用户体验版本用“YB”（用户 Beta）表示； 稳定版用“WD”（稳定）表示，如果没有设置这个属性，Update APK 默认使用“NB”
ro.build.version.incremental	Android 原生	ROM 版本号，没有设置 ro.product.rom.version 的情况下会被使用到
ro.product.brand	Android 原生	品牌商名称

对于支持新的方案来说，需要修改的有主要是以下三个属性

[ro.product.rom.name]: [BoxRom]

[ro.product.rom.type]: [YB]

6. OTA 升级应用服务端配置

服务端需要与使用到推送 OTA 升级公司的后台权限，可联系艺果网络洽谈开通相应的权限，艺果网络的首页：http://pada.cc/

6.1.OTA 升级包的命名和上传路径

在制作完升级包后，将升级包重命名通过 FTP 上传服务器，升级包需要这样命名：

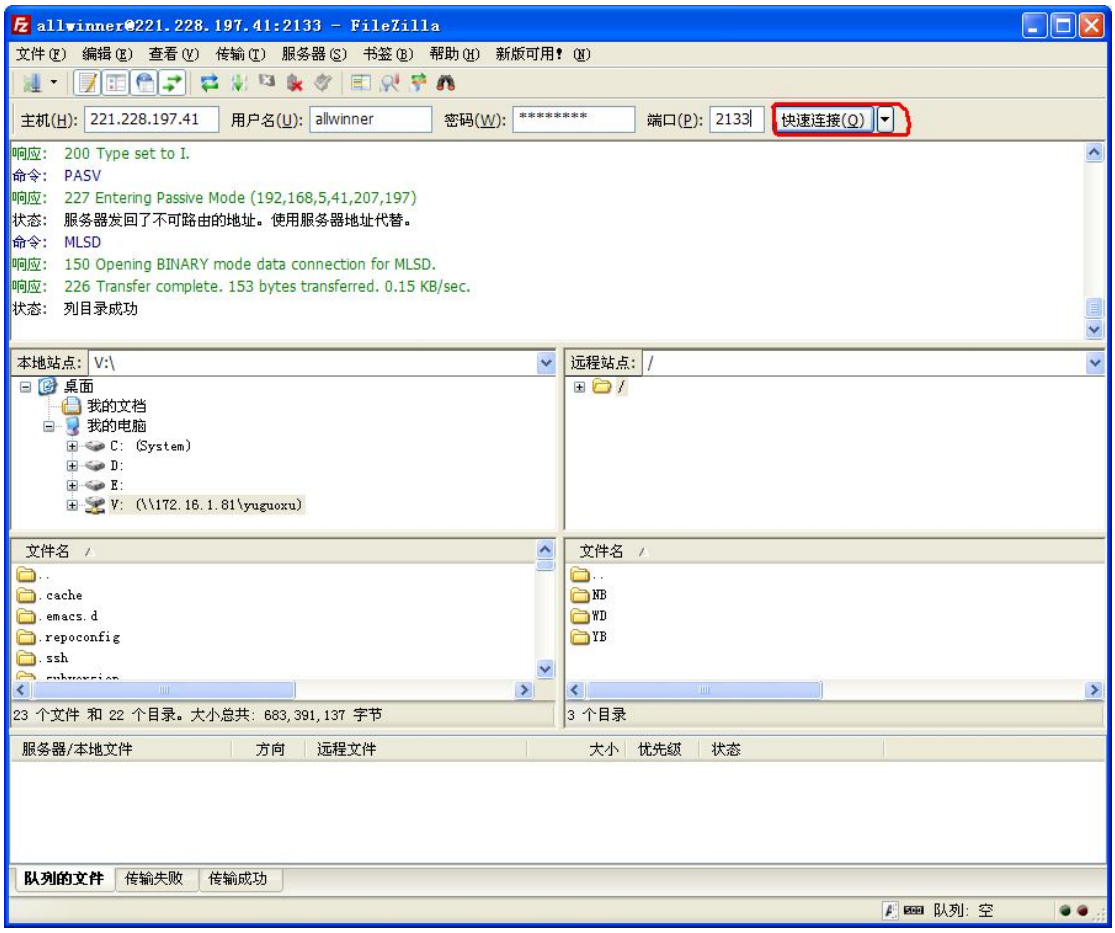
完整包的命名方式：
romName_brand_mode_device_version.release_romType_romVersion.zip
差分包的命名方式
romName_brand_mode_device_version.release_romType_old.romVersion~new.romVersion.zip

注意：
a、文件中“_”是唯一区分字段与字段的符号，如果取出来的字段中有“_”，需要转换成"-"
b、"~"是差分包的 romVersion 的区分符号，不能在文件名其它地方出现；

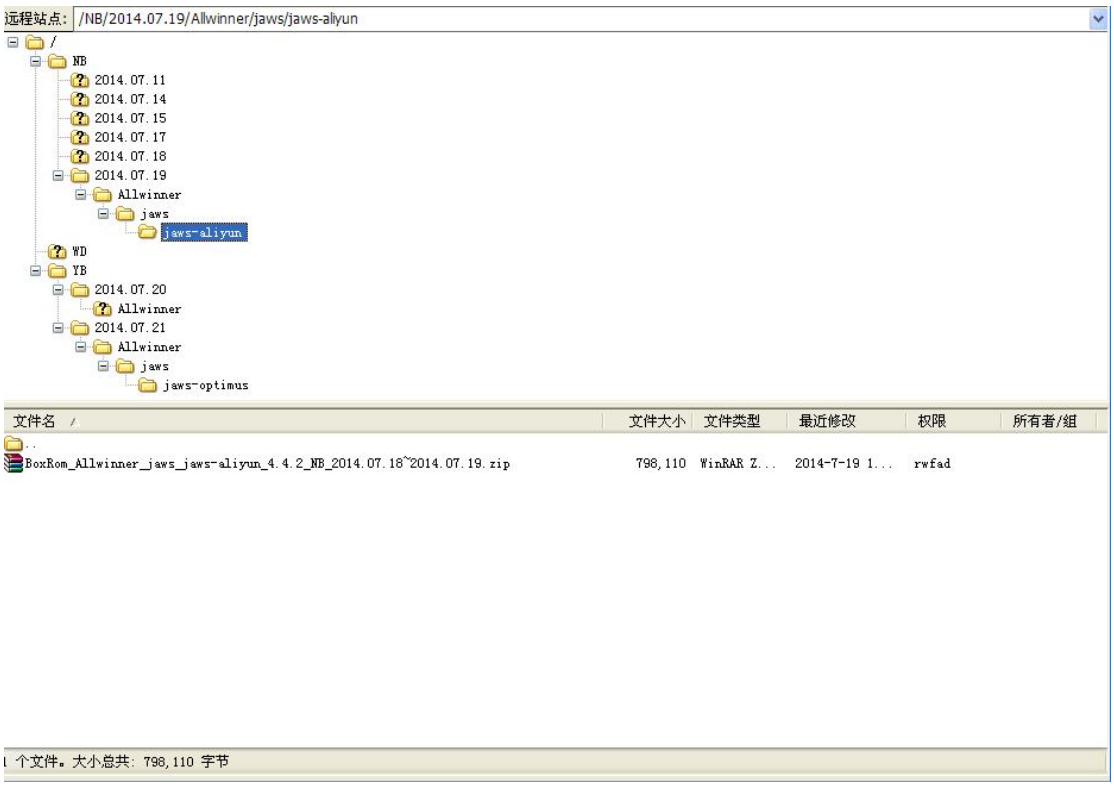
后台 rom 包文件存放目录层次结构：romName->romType->romVersion->brand->model->device
注意路径需要相同，否则后期服务器配置时会出错。

6.2. OTA 升级包的上传方法

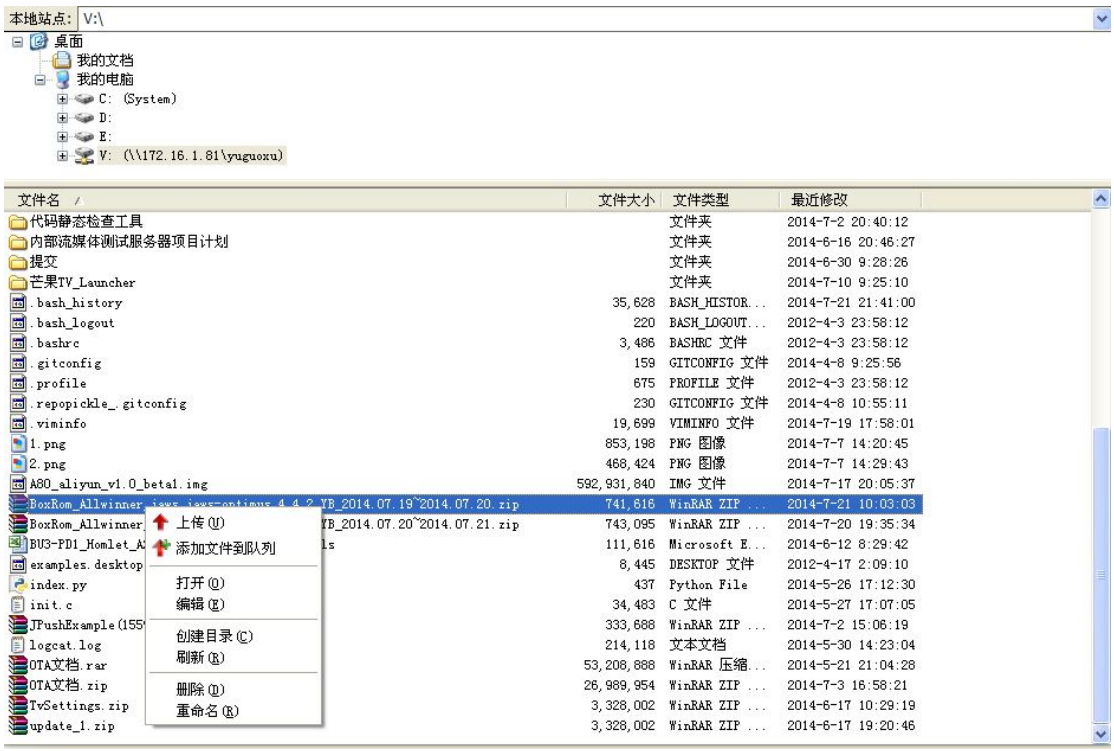
使用 FileZilla软件连接服务端上传FTP包的方法
FileZilla：软件界面如下



点击“快速连接”连接上服务器，右边为服务器上的目录，左边为本地根目录



在右侧按照 romName->romType->romVersion->brand->model->device 的结构建立目录，



软件的左边选择升级包点击右键上传升级包

6.3.升级项目及信息的配置

服务端后台使用深圳市艺果网络科技有限公司提供的后台系统，该后台对上传的 OTA 包进行了 CDN 加速处理，可以确保下载流畅，在此之前需要对服务端进行正确的配置

在使用艺果后台前请联系艺果公司开通后台权限。

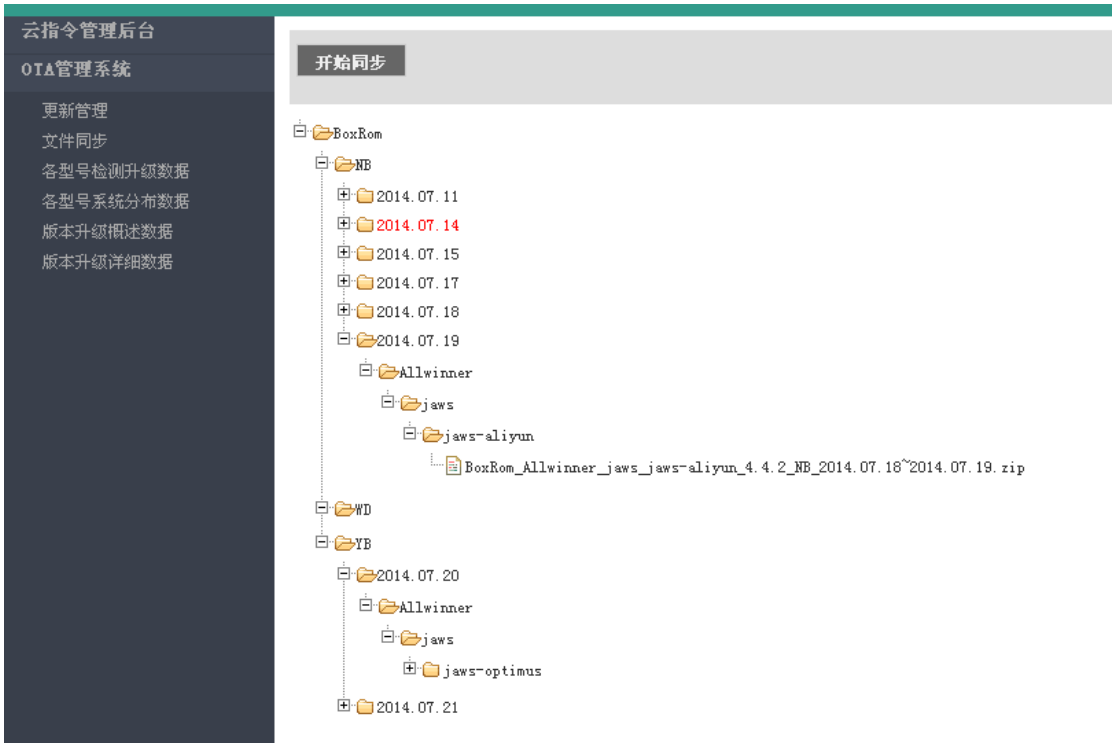
后台的地址：

http://cms.eaglenet.cn/default.aspx

页面截图

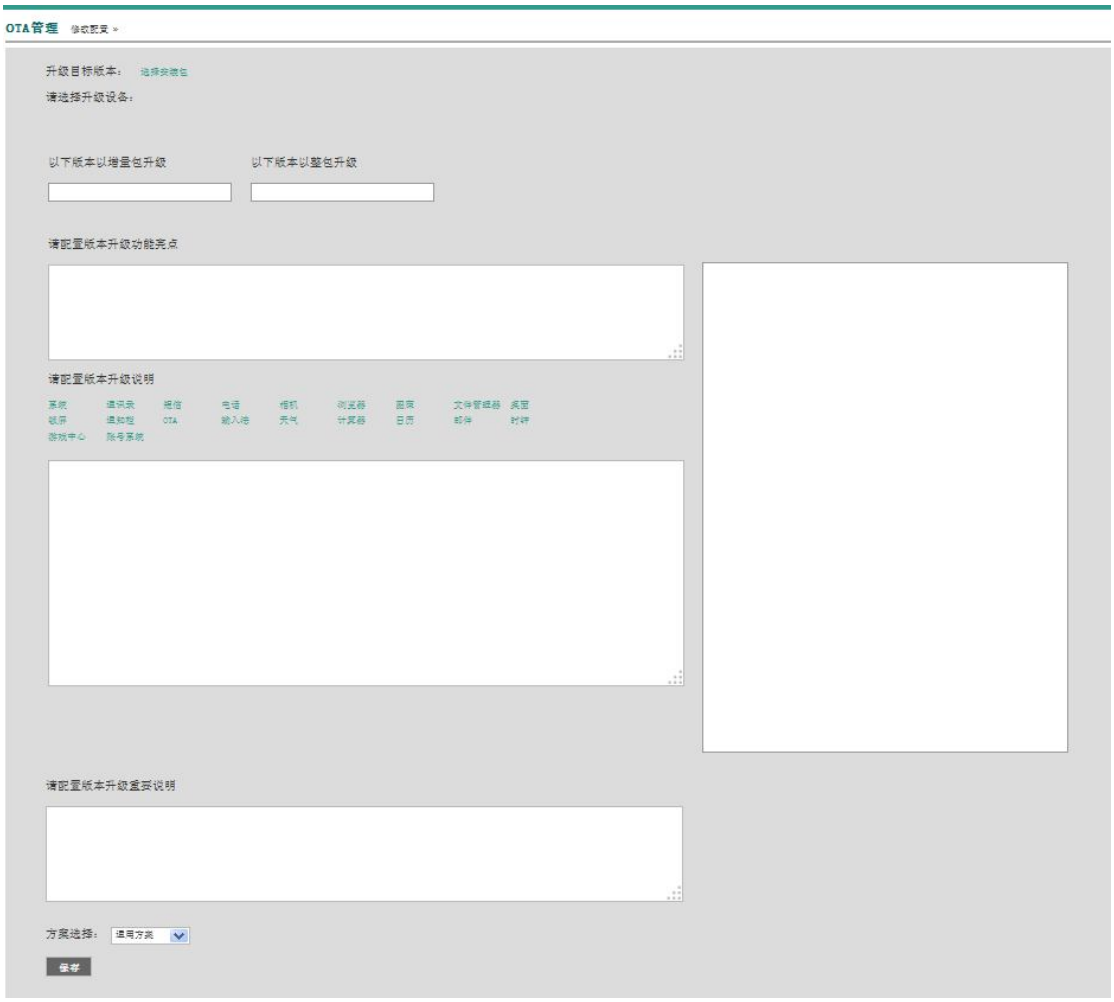
登陆后的主要选项

其中主要使用到的是更新管理何文件同步功能，在通过 FTP 上传了 OTA 文件后，点击文件同步选项，可以看见在 FTP 中建立的文件结构



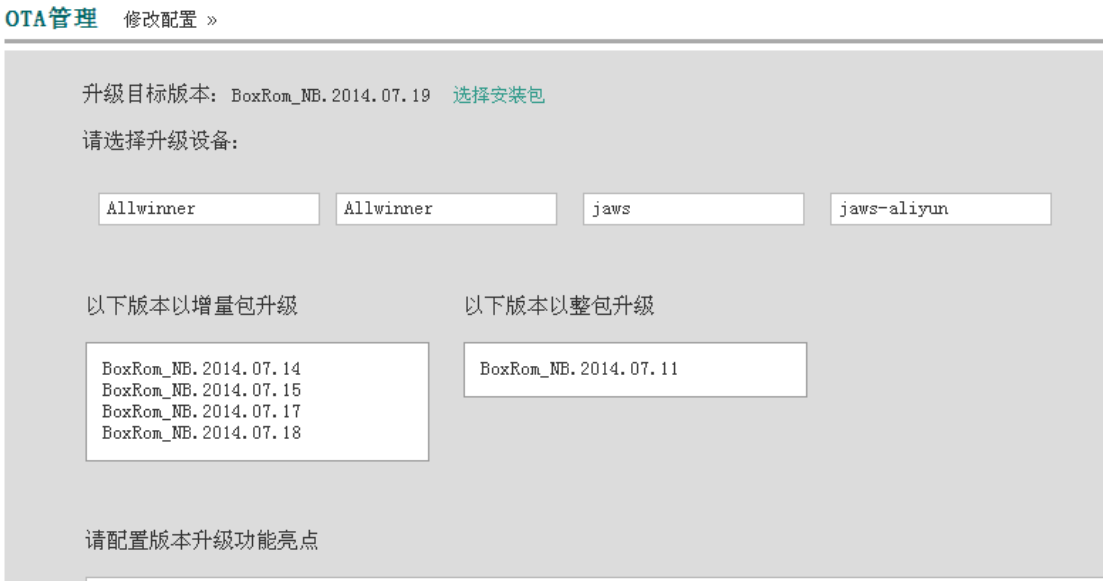
点击“开始同步”按钮，会将上传的 OTA 文件同步到对应的云存储中。等待同步完成。
点击更新管理。切换的选项卡中选择“更新管理”

点击右方的新增升级配置
页面将切换到编辑框，如下图



最上方选择升级的固件，下方为升级信息配置，升级信息汇总为 HTML 文件，发送给 OTA 客户端并显示出来。选择 OTA 升级包的方法如下：

选择后服务器会自动根据固件名和路径确定升级的版本。



系统对于相差太大的版本（超过 4 个），就会采用完整包升级的方式进行升级

在下方填写升级的信息，需要注意的是版本升级说明是用 HTML 来填写的，填写后的效果会显示在界面的右方。

版本功能亮点

xxxxx

升级日志

系统

通讯录

短信

电话

短信

相机

浏览器

图库

文件管理器

点击保存按钮，保存配置

配置完成后可以在下方看见配置信息，注意需要点击配置生效配置，配置信息才真正生效

备	升级方案	配置时间	启用时间	更新时间	操作
ner_jaws_jaws_optimus	通用方案	2014/7/11 15:53:40	2014/7/21 15:46:38	2014/7/11 15:53:40	停用 删除
ner_jaws_jaws-optimus	通用方案	2014/7/15 13:30:21	2014/7/21 15:46:38	2014/7/15 13:30:21	停用 删除
ner_jaws_jaws-aliyun	通用方案	2014/7/18 10:43:31	2014/7/21 15:46:38	2014/7/18 10:43:31	停用 删除
ner_jaws_jaws-aliyun	通用方案	2014/7/18 11:17:15	2014/7/19 11:10:40	2014/7/18 11:17:15	启用 删除
ner_jaws_jaws-asu	通用方案	2014/7/18 11:17:15	2014/7/19 11:10:40	2014/7/18 11:17:15	启用 删除
ner_jaws_jaws-aliyun	通用方案	2014/7/19 14:12:36	2014/7/21 15:46:38	2014/7/19 14:12:36	停用 删除
ner_jaws_jaws-aliyun	通用方案	2014/7/19 15:00:59	2014/7/21 15:46:38	2014/7/19 15:00:59	停用 删除
ner_jaws_jaws-optimus	通用方案	2014/7/19 17:30:09	2014/7/21 15:46:38	2014/7/19 17:30:09	停用 删除
ner_jaws_jaws-optimus	通用方案	2014/7/21 10:06:58	2014/7/21 15:46:38	2014/7/21 10:06:58	停用 删除

Declaration