T651W常用客制化

[1、项目配置 4](#_Toc396837524)

[2、客制化资源 4](#_Toc396837525)

[(1) Logo 4](#_Toc396837526)

[(2) 集成开机动画、关机动画、开机铃声、关机铃声 4](#_Toc396837527)

[(3) 开关机动画的制作 5](#_Toc396837528)

[(4) 默认铃声设置 6](#_Toc396837529)

[(5) 添加项目宏等 6](#_Toc396837530)

[(6) overlay资源控制 7](#_Toc396837531)

[(7) 情景模式配置（默认音量调节，默认震动音等开启） 7](#_Toc396837532)

[3、APN/SPN/ Voicemail 8](#_Toc396837533)

[(1) APN 8](#_Toc396837534)

[(2) SPN 8](#_Toc396837535)

[(3) Voicemail 8](#_Toc396837536)

[(4) 关于虚拟运营商: 8](#_Toc396837537)

[4、默认语言和输入法 9](#_Toc396837538)

[(1) 项目添加语言和默认语言 9](#_Toc396837539)

[(2) 根据SIM卡的MCC自动识别语言 9](#_Toc396837540)

[(3) 输入法 9](#_Toc396837541)

[5、桌面布局 11](#_Toc396837542)

[6、Browser/Email 11](#_Toc396837543)

[(1) Browser 11](#_Toc396837544)

[(2) Email 11](#_Toc396837545)

[7、时区 12](#_Toc396837546)

[(1) 默认时区配置 12](#_Toc396837547)

[(2) 需要添加的配置 12](#_Toc396837548)

[(3) NITZ默认打开设置 13](#_Toc396837549)

[(4) 默认日期格式 13](#_Toc396837550)

[(5) 24小时配置 14](#_Toc396837551)

[8、GMS 14](#_Toc396837552)

[9、内置APK的添加 15](#_Toc396837553)

[(1) 已经编译的APK 15](#_Toc396837554)

[(2) 源码（例如添加应用Test） 16](#_Toc396837555)

[10、锁网 17](#_Toc396837556)

[(1) Modem内置锁网值 19](#_Toc396837557)

[(2) 工具修改Modem配置锁网 20](#_Toc396837558)

[11、拨号 20](#_Toc396837559)

[(1) 紧急拨号 20](#_Toc396837560)

[(3) MTK平台CC相关的Log查询 23](#_Toc396837561)

[12、网络漫游配置 26](#_Toc396837562)

[(1) 何为漫游 26](#_Toc396837563)

[(2) 如何区别和取消漫游 26](#_Toc396837564)

[13、GPS相关 27](#_Toc396837565)

[(1) GPS coclock 配置 27](#_Toc396837566)

[A) 什么是GPS Coclock 27](#_Toc396837567)

[B) GPS Coclock配置 28](#_Toc396837568)

[C) GPS Log抓取 28](#_Toc396837569)

[D) GPS 校准数据的备份和恢复 29](#_Toc396837570)

[(2) AGPS 服务器配置 29](#_Toc396837571)

[14、 内置联系人 30](#_Toc396837572)

[(1)如何内置联系人 30](#_Toc396837573)

[(2)号码匹配 30](#_Toc396837574)

[15、短信相关 32](#_Toc396837575)

[(1) 短信中心号码 32](#_Toc396837576)

[(2) 内置小区广播号码 32](#_Toc396837577)

[(3) 8Bit 32](#_Toc396837578)

[(4) 7Bit 32](#_Toc396837579)

[(5) Single Shift 和 Locking Shift 32](#_Toc396837580)

[16、SIM文件的读写支持 33](#_Toc396837581)

[(1) 如何查看当前代码支持的SIM卡文件 33](#_Toc396837582)

[(2) 根据SIM卡文件显示STK名称 34](#_Toc396837583)

[17、MTK默认的显示运营商名称 34](#_Toc396837584)

[(1) MTK添加运营商名称的缘由 34](#_Toc396837585)

[(2) 如何去掉运营商名称的显示 35](#_Toc396837586)

[18、FOTA 36](#_Toc396837587)

[19、文件管理和视频播放 37](#_Toc396837588)

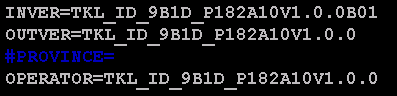
[20、特殊的号码匹配（委内瑞拉） 37](#_Toc396837589)

[21、特殊虚拟运营商的APN识别 37](#_Toc396837590)

# 1、项目配置

Makefile: mediatek/config/P182A10/ProjectConfigXXX

Version: mediatek/custom/P182A10/versionXXX



UA: mediatek/config/P182A10/custom.conf\_XXX

[quickbuild -> build\_config]

FeatrueOption: mediatek/config/common/ProjectConfig.mk

mediate/build/tools/javaoption.pm

# 2、客制化资源

## (1) Logo

开机第一屏时通过ProjectConfig.mk中的BOOT\_LOGO=XXX来定义用哪个的目录的开机第一屏图片。

我们新增一个开机第一屏定制目录：BOOT\_LOGO=T651W

在mediatek/custom/common/lk/logo目录增加t651w目录，注意图片的分辨率要和ProjectConfig.mk中的宽高值一致，不然开机第一屏不会显示。

$ cp cmcc\_fwvga t651w -r

将t651w目录中所有文件前缀修改为t651w，替换开机第一屏图片。

另需要在

文件



之后添加。

如果定制的第一屏是QHD的则需要在

后添加对应路径名称

## (2) 集成开机动画、关机动画、开机铃声、关机铃声

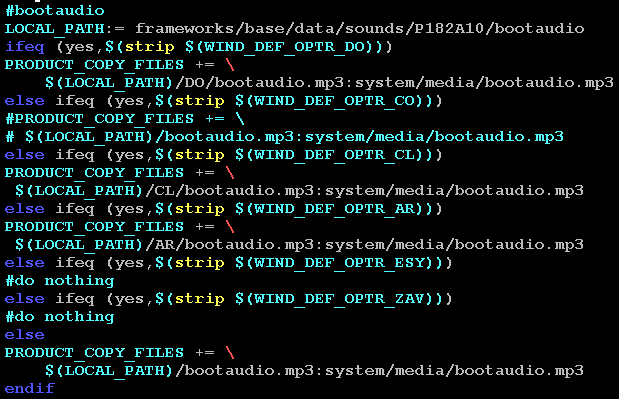
我们客制化开机动画的路径：frameworks/base/data/sounds目录

资源文件路径：frameworks/base/data/sounds/P182A10/

编译脚本：修改frameworks/base/data/sounds/P182A10Audio.mk文件

已开机铃声为例，有的项目要求不集成开机铃声，则在该项目的if判断中

#do nothing



## (3) 开关机动画的制作

分别创建名为“part0”和“part1”的文件夹以及一个名为“desc.txt”文件。“part0”中存储动画的第一阶段的资源图片，“part1”存储第二阶段的资源图片，注意图片为png格式。

  播放控制由“desc.txt”指定，内容如下：

   480 854 12

   p 1 0 part0

   p 0 0 part1

   各参数功能如下：**( 注意：desc.txt文本内容必须用单个空格隔开，且不能有多余空行。)**

|  |  |  |  |
| --- | --- | --- | --- |
| 480 | 854 | 12 |  |
| 宽 | 高 | 每秒播放帧数 |  |
| p | 1 | 0 | part0 |
| 标志符 | 循环次数 | 阶段切换间隔时间 | 对应目录名 |
| p | 0 | 0 | part1 |
| 标志符 | 循环次数 | 阶段切换间隔时间 | 对应目录名 |

  最后，将这三个组件通过**存储压缩**的方式压缩为**bootanimation.zip**文件即制作完成。

关机动画的制作也是如此，将这三个组件通过**存储压缩**的方式压缩为**shutanimation.zip**文件

关机动画及关机铃声注意事项

需要在mediatek/config/P182A10/system.prop添加

ro.operator.optr=CUST，这个添加后，没有添加关机动画的项目，关机时会显示android默认的关机动画。

考虑到项目客制化，我们新增了一个开关SHOW\_SHUTDOWN\_ANIMATION共同控制。

需要关机动画的项目则需要在mediatek/config/P182A10/ProjectConfigXX

中添加SHOW\_SHUTDOWN\_ANIMATION=yes

这个值默认为no。

## (4) 默认铃声设置

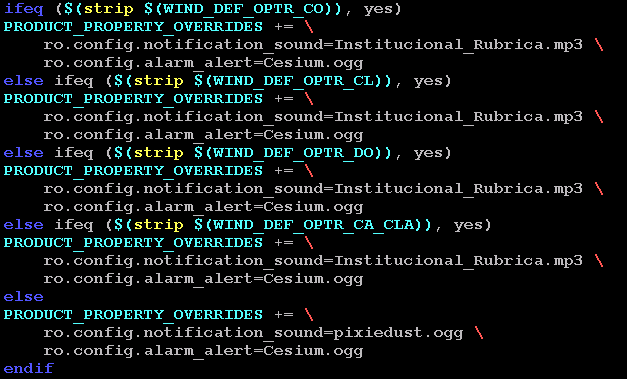
集成脚本详参frameworks/base/data/sounds/P182A10Audio.mk

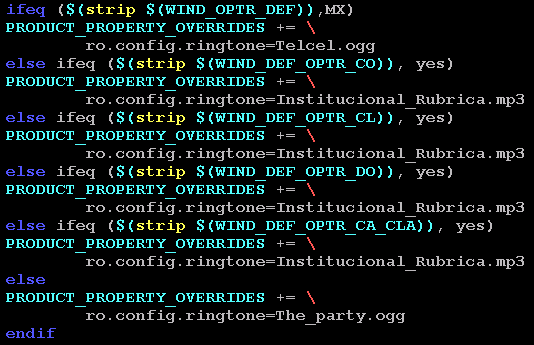
来电铃声copy到system/media/audio/ringtones

通知铃声copy到system/media/audio/notifications

闹钟铃声copy到system/media/audio/alarms

默认铃声配置：修改build/target/product/P182A10.mk





## (5)添加项目宏等

以添加添加开关WIND\_DEF\_OPTR\_XX

在mediatek/config/common/ProjectConfig.mk添加WIND\_DEF\_OPTR\_XX=no

在mediatek/config/P182A10/ProjectConfigXX添加WIND\_DEF\_OPTR\_XX=yes

在mediatek/build/tools/javaoption.pm添加WIND\_DEF\_OPTR\_XX

在上层java代码中通过FeatureOption. WIND\_DEF\_OPTR\_XX进行客制化，注意导包：

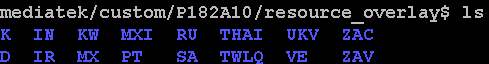


注：FeatureOption中能够添加的开关只能是boolean类型

## (6) overlay资源控制

修改对应项目mediatek/config/P182A10/ProjectConfigXX文件中的RESOURCE\_OVERLAY\_SUPPORT值，其中的generic修改为XX（项目参数）

在其目录下新建文件夹XX，建议将generic文件复制一份出来重命名为XX。。修改对应项目mediatek/config/P182A10/ProjectConfigXX文件中的RESOURCE\_OVERLAY\_SUPPORT值，其中的generic修改为XX（项目参数）

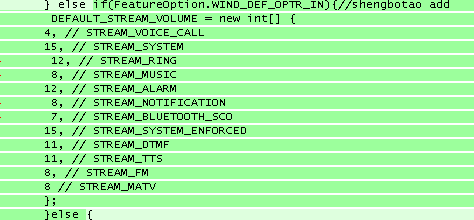


在其目录下新建文件夹XX，建议将generic文件复制一份出来重命名为XX。。

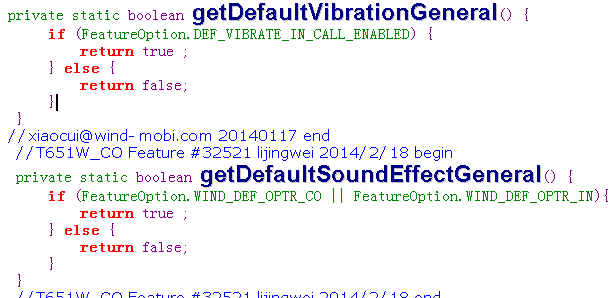
## (7) 情景模式配置（默认音量调节，默认震动音等开启）

1. 默认音量调节





（2）默认震动音等声音的开启



以上是震动和触摸屏的默认声音修改。

# 3、APN/SPN/ Voicemail

## (1) APN

SourceFile: mediatek/frameworks/base/telephony/etc/apns-conf-XXX.xml

Copymk: build/target/product/P182A10.mk

拷贝客制化的apn参加编译

注：在配置apn.xml时，Android4.2 <apns version=”7”>

Android4.4 <apns version=”8”>

## (2) SPN

（1）实体运营商APN配置文件

SourceFile: mediatek/frameworks/base/telephony/etc/spn-conf-XXX.xml

Copymk: build/target/product/common.mk

（3）虚拟运营商APN配置文件(MTK默认支持下面三种配置，配置前通过需求确认哪种)

SourceFile: mediatek/frameworks/base/telephony/etc/virtual-spn-conf-by-efspn-XXX.xml

通过spn识别的虚拟运营商

SourceFile: mediatek/frameworks/base/telephony/etc/virtual-spn-conf-by-efpnn-XXX.xml

通过pnn识别的虚拟运营商

SourceFile: mediatek/frameworks/base/telephony/etc/virtual-spn-conf-by-efimsi-XXX.xml

通过imsi识别的虚拟运营商

Copymk: frameworks/base/Android.mk

## (3) Voicemail

SourceFile: mediatek/frameworks/base/telephony/etc/voicemail-conf-XXX.xml

Copymk: frameworks/base/Android.mk

系统生成的时候会读取voicemail-conf.xml生成到对应的contact数据库中，索引是MCCMNC。

## (4) 关于虚拟运营商:

MVNO(Mobile Virtaul Network Operator)虚拟网络运营商，没有自己的实体网络，通过租用MNO(Mobile Network Operator)的网络来提供网络服务。

我们知道Spec规定operator之间是通过MCC/MNC （Mobile Country Code/Mobile Network Code）来区分的；而MVNO和对应MNO的MCC/MNC是相同的，那就需要MVNO定义额外的栏位（通常都是SIM卡中某支文件）来和对应MNO做区分；具体这个额外的栏位是什么是每个MVNO自己定义的，需要向MVNO确认。

目前MTK支持区分MVNO的方式有四种，每种区分方式对应一个xml的配置表：

1. EF\_SPN方式，对应MVNO配置到Virtual-spn-conf-by-efspn.xml中

2. EF\_IMSI方式，对应MVNO配置到Virtual-spn-conf-by-imsi.xml中

3. EF\_PNN方式，对应MVNO配置到Virtual-spn-conf-by-efpnn.xml中

4. EF\_GID1方式，对应MVNO配置到Virtual-spn-conf-by-efgid1.xml中

**详情请参见：[FAQ09811] 如何区分MNO和MVNO**

# 4、默认语言和输入法

## (1) 项目添加语言和默认语言

修改mediatek/config/P182A10/ProjectConfigXX中的MTK\_PRODUCT\_LOCALES=，

添加需要集成的语言，默认语言放在第一个。

## (2) 根据SIM卡的MCC自动识别语言

Android 默认支持通过SIM识别默认国家（通过MCC）来设置默认语言。

SRC：

framework/opt/telephony/src/java/com/android/internal/telephony/MccTable.java

MccTable.java中sTable数组的添加，

MccEntry(MCC, ISO, smallestDigitsMCC [, Language] ).

如果遇到不能通过SIM卡来切换默认语言的问题，请确认对应的MCC列表中，是否有Language的设置。



## (3) 输入法

修改文件 mediatek/config/P182A10/ProjectConfigXX

将宏 DEFAULT\_INPUT\_METHOD 的值设置为所需的输入法；

比如 PinyinIME，则设为

DEFAULT\_INPUT\_METHOD = com.android.inputmethod.pinyin.PinyinIME

这个值是输入法源码中，包含InputMethodService子类的那个包名.实现InputMethodService的子类的类名。这里比log中打印的完整类名少一个"/",请注意

补充说明：上面的修改方法只适用于默认输入法为

com.android.inputmethod.pinyin/.PinyinIME这种格式（即反斜杠后面就是类名的情况），

对于com.google.android.inputmethod.latin/com.android.inputmethod.latin.LatinIME

（即反斜杠后面是完整类名的情况），

这种情况是不支持修改mediatek/config/${Project}/ProjectConfig.mk

里面DEFAULT\_INPUT\_METHOD的方法来设置默认输入法的。

以com.google.android.inputmethod.latin/com.android.inputmethod.latin.LatinIME 为例

请改用下面的方法修改，在SettingsProvider 的DatabaseHelper.java中有

            Log.i(TAG, "defaultIME : " + defaultIME);

           if (defaultIME != null) {

                StringBuffer strBuffer = new StringBuffer(defaultIME);

               int lastDotPos = defaultIME.lastIndexOf(".");

                String strInsert = "/";

               strBuffer.insert(lastDotPos, strInsert);

               loadSetting(stmt, Settings.Secure.DEFAULT\_INPUT\_METHOD,

                       strBuffer.toString());

            } else {

               Log.i(TAG, "default IME from IMEFeatureOption is null!!");

            }

请将默认的初始化部分code删除，即将下面的code都删除掉。

 Log.i(TAG, "defaultIME : " + defaultIME);

           if (defaultIME != null) {

                StringBuffer strBuffer = new StringBuffer(defaultIME);

               int lastDotPos = defaultIME.lastIndexOf(".");

                String strInsert = "/";

               strBuffer.insert(lastDotPos, strInsert);

               loadSetting(stmt, Settings.Secure.DEFAULT\_INPUT\_METHOD,

                       strBuffer.toString());

            } else {

               Log.i(TAG, "default IME from IMEFeatureOption is null!!");

            }

然后在删除的后面添加

  loadSetting(stmt, Settings.Secure.ENABLED\_INPUT\_METHODS,

   "com.google.android.inputmethod.latin/com.android.inputmethod.latin.LatinIME");

//这里修该默认enable的输入法

 loadSetting(stmt, Settings.Secure.DEFAULT\_INPUT\_METHOD,

   "com.google.android.inputmethod.latin/com.android.inputmethod.latin.LatinIME");

//这里修改默认选中的输入法

# 5、桌面布局

主桌面布局，集成壁纸均需要在overlay中修改。

SourceFile: packages/apps/Launcher2/res/xml/default\_workspace.xml 【4.2，注意overlay】

SourceFile: packages/apps/Launcher3/res/xml/default\_workspace.xml 【4.4，注意overlay】

壁纸资源放置在xx/resource\_overlay/xx/packages/apps/Launcher3/res/drawable-hdpi

修改xx/resource\_overlay/xx/packages/apps/Launcher3/res/values-nodpi/wallpapers.xml

默认壁纸放置在xx/resource\_overlay/xx/frameworks/base/core/res/res/drawable-nodpi目录，

命名为default\_wallpaper.jpg。【Android4.4默认壁纸不需要放到壁纸资源目录，也不需要在wallpapers.xml

中集成，但是Android4.2需要】

# 6、Browser/Email

## (1) Browser

修改homepage和bookmarks

Sourcefile:

mediatek/custom/P182A10/resource\_overlay/XXX/packages/apps/Browser/res/values/**strings.xml**

(修改**homepage\_base**和紧接着下面的bookmarks)

Sourcefile:

mediatek/custom/P182A10/resource\_overlay/XXX/packages/apps/Browser/res/values/**mtk\_strings.xml**

[此文件需要观察**homepage\_base\_site\_navigation**是否符合要求以作修改]

如果需要根据MNC和MNN来自适应homepage，请参考我的AR定制的Browser提交changeid:i084c5。

## (2) Email

**(1) 默认自动数据添加**

修改邮箱的APN（即设定自动登陆邮箱的服务器信息）

Sourcefile：

mediatek/custom/P182A10/resource\_overlay/XXX/packages/apps/Email/res/xml/providers.xml

注意，此处修改后，也无法显示在手动配置界面，google默认没有此显示功能。

配置信息时，注意端口号和协议的编写，**如果是不包含在providers.xml中的域，则需要再拷贝增加的信息到product\_provider.xml中**，否则信息无法追加。

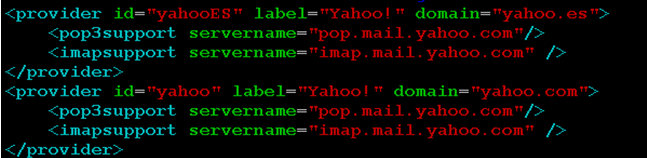
**(2) 手动选择添加账户**

一般来说，手动配置账户是需要用户更改的，系统默认会根据选择的账户类型填充server地址、端口号以及协议类型。

目前遇到某些定制要求选择手动配置后，server地址、端口号以及协议类型符合定制要求的值。

接收邮箱server配置：修改packages/apps/Email/res/xml/manual\_setup\_helper.xml

同一个域名可能会要求配置pop3、imap、exchange类型的账户，按照如下格式将servername设置为需求提供的即可。

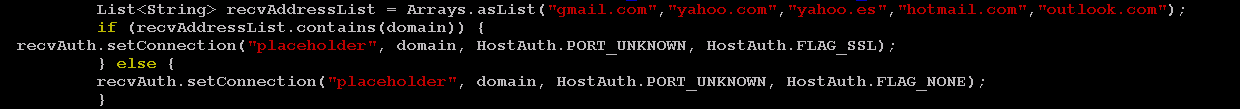


接收邮箱端口以及协议配置：一种协议类型会对应一个端口，所以设置好协议类型就可以满足这个两个需求。

修改文件packages/apps/Email/src/com/android/email/activity/setup/AccountSetupBasics.java中的onManualSetup(Boolean allowAutoDiscover) 方法。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 协议 | POP3端口(接收) | IMAP端口(接收) | SMTP端口(发送) | 配置协议类型 |
| NONE | 110 | 143 | 25 | HostAuth.FLAG\_NONE |
| STARTTLS | 110 | 143 | 25 | HostAuth.FLAG\_SSL |
| SSL/TLS | 995 | 993 | 465 | HostAuth.FLAG\_TLS |

我们可以通过判断域名来设置需求要求的设置发送服务器或接收服务器的协议类型。



# 7、时区

## (1) 默认时区配置

Android4.2 和 Android4.4

1. 首先确定时区ID 例如：时区为GMT+1,Madrid

时区ID 为：Europe/Madrid

备注：时区ID 可以从time\_zones\_by\_country.xml文件中查询

frameworks/base/core/res/res/xml/time\_zones\_by\_country.xml

1. 在build/target/product/P182A10.mk文件中配置项目的默认时区persist.sys.timezone

例如：ifeq ($(strip $(WIND\_DEF\_OPTR\_ESQ)), yes)

PRODUCT\_PROPERTY\_OVERRIDES += \

persist.sys.timezone=Europe/Madrid

## (2) 需要添加的配置

Android4.2 和 Android4.4

* 1. 如果设置—日期和时间—时区列表中没有要设置的默认时区，需要在packages/apps/Settings/res/xml-xx-xx/timezones.xml 文件中添加，也可将此文件移到覆盖资源目录下去修改：

mediatek/custom/P182A10/resource\_overlay/XXX/packages/apps/Settings/res/xml-xx-xx /timezones.xml

参考：FAQ06455 [Zone]如何添加一个新时区

* 1. 如果设置—日期和时间—时区列表里选择的时区名称和界面上显示的时区名称不一致时，

可以参考FAQ08718 进行修改

FAQ08718 [TimeZone]如何在ICU 中自定义某个时区的显示名称

Android4.4

如果一个国家有多个时区（查看time\_zones\_by\_country.xml文件）

例如：<timezone code="pt">Europe/Lisbon</timezone>

<timezone code="pt">Atlantic/Madeira</timezone>

<timezone code="pt">Atlantic/Azores</timezone>

需要修改以下文件

frameworks/opt/telephony/src/java/com/android/internal/telephony/gsm/GsmServiceStateTracker

.java

修改private String[][] mTimeZoneIdOfCapitalCity数组的定义，给那个拥有多个时区的国家指定一个默认时区：

例如：{"pt","Europe/Lisbon"}

修改此处的目的是：避免出现空指针异常。

修改此处的大概原因如下：有些海外运营商不支持从网络获取时区，这个时候，就会根据MCC/MNC来更新时区，但有些国家配置了多个时区，代码中会根据国家代码从mTimeZoneIdOfCapitalCity数组中获取国家对应的时区，如果没有再此数组中设置，获取不到时区信息，执行以下语句时就会报空指针异常。

zone = getTimeZonesWithCapitalCity(iso);

setAndBroadcastNetworkSetTimeZone(zone.getID());

## (3) NITZ默认打开设置

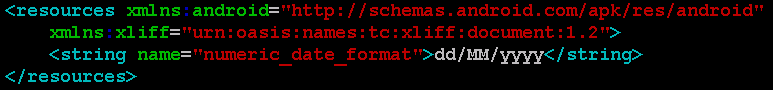




以上两项设置为true即可。

## (4)默认日期格式





需要注意，因为此处修改是在overlay里面的，这个文件有可能也在某些语言下也有配置的。所以修改的时候记得将其他语言下的此文件也要overlay一下。如下：英语和西班牙语下的都要覆盖。



## (5) 24小时配置





如果是12小时，则配置



# 8、GMS

1. GMS包里的信息添加
2. 项目MakeFile中配置编译宏
3. 如何检查

一般集成哪些GMS应用不会出错，主要问题出现在GMS需求表中一些属性的配置。

Client ID一般会有两种类型

|  |  |
| --- | --- |
| **Client ID** | **Value** |
| ro.com.google.clientidbase | android-zte |
| ro.com.google.clientidbase.ms | android-americamovil-{country} |
| ro.com.google.clientidbase.am | android-americamovil-{country} |
| ro.com.google.clientidbase.gmm | android-zte |
| ro.com.google.clientidbase.yt | android-zte |

|  |  |
| --- | --- |
| **Client ID** | **Value** |
| ro.com.google.clientidbase | android-zte |
| ro.com.google.clientidbase.ms | andorid-zte |
| ro.com.google.clientidbase.am | andorid-zte |
| ro.com.google.clientidbase.gmm | andorid-zte |
| ro.com.google.clientidbase.yt | andorid-zte |

代码中宏控已经加好，Client\_ID默认配置为下面表格中的值，如果为上面一种，则需要在对应项目的ProjectConfigXX中添加GMS\_CLIENTID\_ZTE\_DEFAULT=no

需要注意ro.product.xxx(model、brand、board)值，编译之后查看一下out目录下system/build.prop，检查属性值是否正确，有差异，一般修改build/target/product/P182A10.mk文件和build/tools/buildinfo.sh文件。

# 9、内置APK的添加

## (1) 已经编译的APK

1、集成apk不可卸载

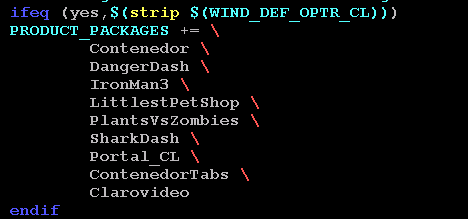
在alps/mediate/binary/package/P182A10创建运营商目录XX，将需要集成的apk放在该目录下，添加Android.mk

注意集成的APK，不可更改其签名，否则会造成该APK应用升级时出现签名冲突的问题

即配置LOCAL\_CERTIFICATE := PRESIGNED

Android.mk最好用项目开关控制好。

修改alps/build/target/product/common.mk文件



2、集成apk需要可以卸载，并且恢复出厂设置需要能够恢复

首先确认ProjectConfigXX文件中MTK\_SPECIAL\_FACTORY\_RESET=yes

之前的项目在verdor目录下直接添加对应目录及文件即可，但是如果在vendor目录增加，T651W的不同项目将无法区分。

我们目前还是放在alps/mediate/binary/package/P182A10这个目录下。

以西班牙Yoigo为例，在上述目录添加ESY目录。

在ESY目录下创建目录data/app，system/app，system/appbackup,一般还需要创建system/lib

在data/app目录下放置需要集成的apk，以及隐藏文件.keep\_list该文件为空。

在system/appbackup目录下放置需要集成的apk

在system/lib目录放置apk需要的库文件

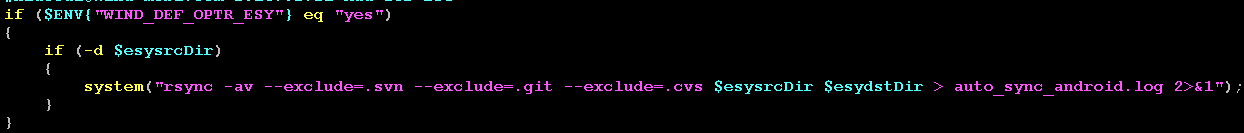
在system/app放置隐藏文件.restore\_list，该文件列出需要回复的apk名称。



然后修改mediatek/build/tools/mtkBegin.pl文件

添加//大家可以查看代码





这个主要是将ESY中的内容拷贝到vendor目录下。

## (2) 源码（例如添加应用Test）

添加路径：mediatek/packages/app/Test

编译脚本：

1、在 Test 目录下创建一个名为 Android.mk的文件，内容如下：

Ifeq ($(strip $(WIND\_DEF\_OPTR\_XX),yes) //或者直接添加一个控制该apk是否集成的宏控

LOCAL\_PATH:= $(call my-dir)

include $(CLEAR\_VARS)

LOCAL\_MODULE\_TAGS := optional

LOCAL\_SRC\_FILES := $(call all-subdir-java-files)

LOCAL\_PACKAGE\_NAME := Test

LOCAL\_CERTIFICATE := platform

include $(BUILD\_PACKAGE)

endif

打开文件 build/target/product/${Project}.mk （其中 ${Project} 表示工程名）

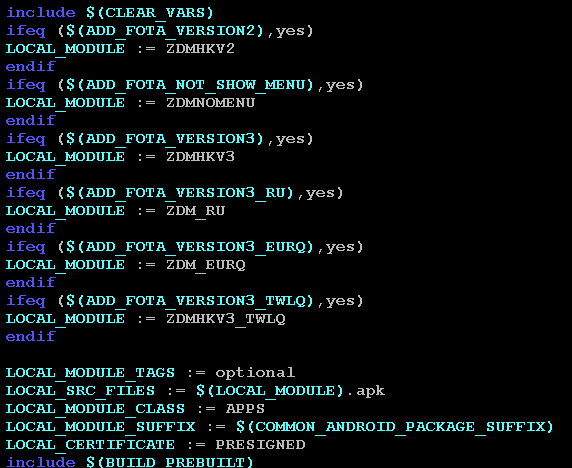
将 Test 添加到 PRODUCT\_PACKAGES 里面。

2、FOTA

添加路径：mediatek/binary/packages/${Project}/Fota

编译脚本：编译脚本同集成第3方apk。

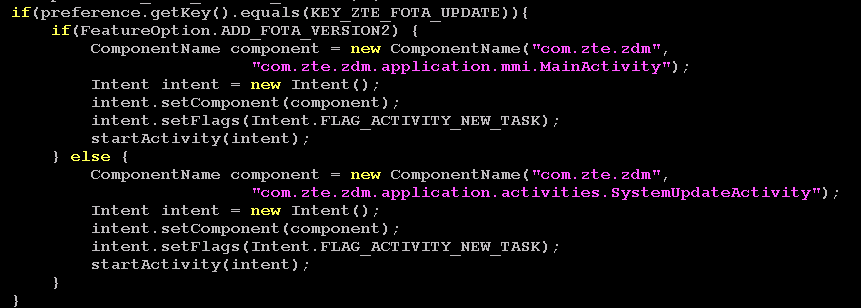
P182A10定制的版本较多，要求集成的fota应用有差异，可以添加宏开关控制集成那个Fota应用。



将Fota应用添加到PRODUCT\_PACKAGES 里时同样用宏控制区别。

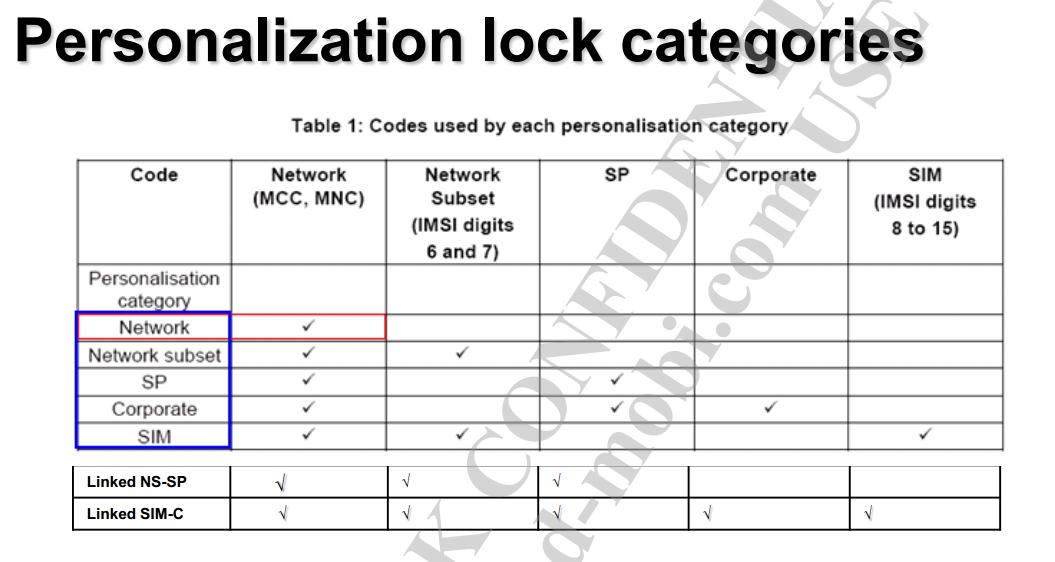
※Setting中菜单的调用---包名

在Settings目录的SoftwareUpdates.java的onPreferenceTreeClick方法中增加启动Fota的代码，不同的fota版本启动的Activity可能会不一样。所以集成Fota应用之后，一定要查看一下Setting中的Fota升级菜单能否启动该应用。

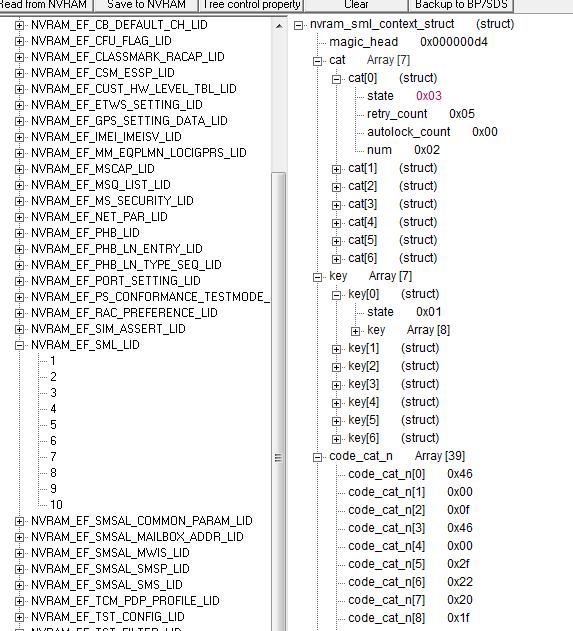


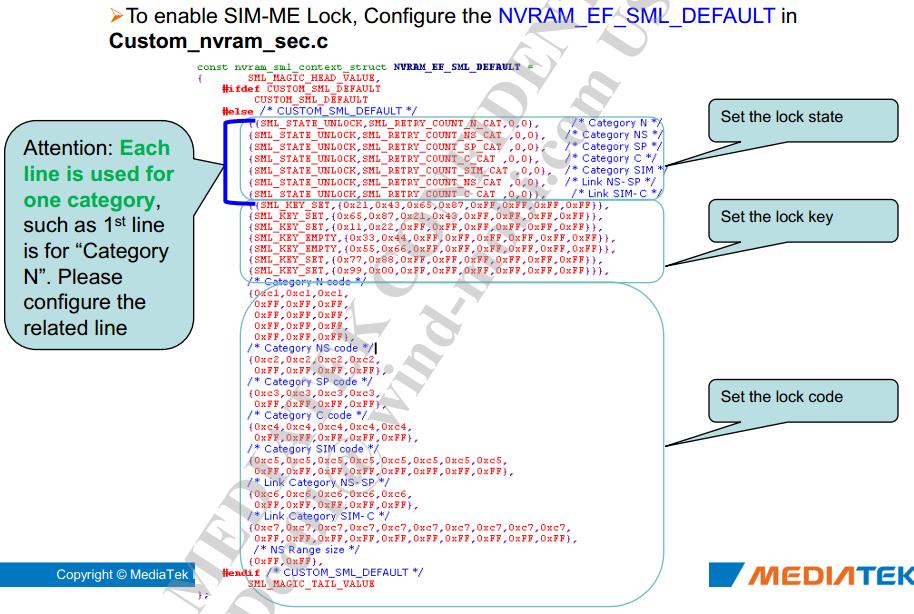
# 10、锁网

某些运营商出于利益保护，会禁止其他运营商的SIM卡通过其发布的终端入网，获取电信服务。MTK平台上常用的锁网类型如下图。前五项是3GPP协议（22.022）定义了上张表格中的5中锁网种类，最后两项是MTK自己扩展的。

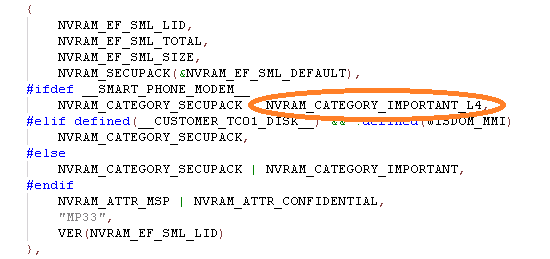


MTK平台对锁网的配置主要是Modem部分的NV的配置。如下图





SML锁网的NV参数配置如下：



主要要注意参数NVRAM\_CATEGORY\_IMPORTANT\_L4；若NVRAM具有NVRAM\_CATEGORY\_IMPORTANT属性或者有NVRAM\_CATEGORY\_IMPORTANT\_L4属性，会被自动添加NVRAM\_ATTR\_AUTO\_RECOVER属性。这样这些NVRAM在modem端读取失败之后，就会触发自BIN Region中的recovery。即，此类参数的NVRAM在下载的时候不会被清除掉，除非格式化。如果要去掉此属性，这样的修改之后，每次factory rese(恢复出厂设置)都会让锁卡数据恢复成默认值。

目前仅仅有IMEI, CALIBRATION, SIM-ME Lock相关的NVRAM会被back up到BIN Region中。

## (1) Modem内置锁网值

之前T651W所有的Android 4.2版本上的锁网参数都是直接配置在NVRAM\_EF\_SML\_DEFAULT数组里，直接编译Modem，配置对应的客制化项目。

这样配置Modem直接锁网有利也有弊。

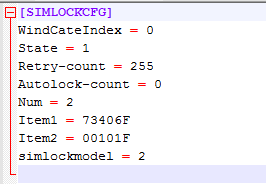
好处：版本单一，前方、工厂对版本状态比较明确，测试简单。

不便：不同版本之前升级必须格式化，如果后续调板升级必须格式化。对于兼容很多国家的渠道版本不能做到有的支持锁网，有的不锁网。

## (2) 工具修改Modem配置锁网

Android 4.4版本之后更新了锁网配置机制。改成了通过工具读取写入NVRAM\_EF\_SML\_DEFAULT字段。这样通过一个配置的锁网文件SIMLOCK.ini来锁不同的运营商。

如下是一个委内瑞拉的锁网文件



这样处理也有利也有弊

好处：渠道版本可以兼容锁网或者不锁网。AP端可以使用一套Modem兼容很多客制化。

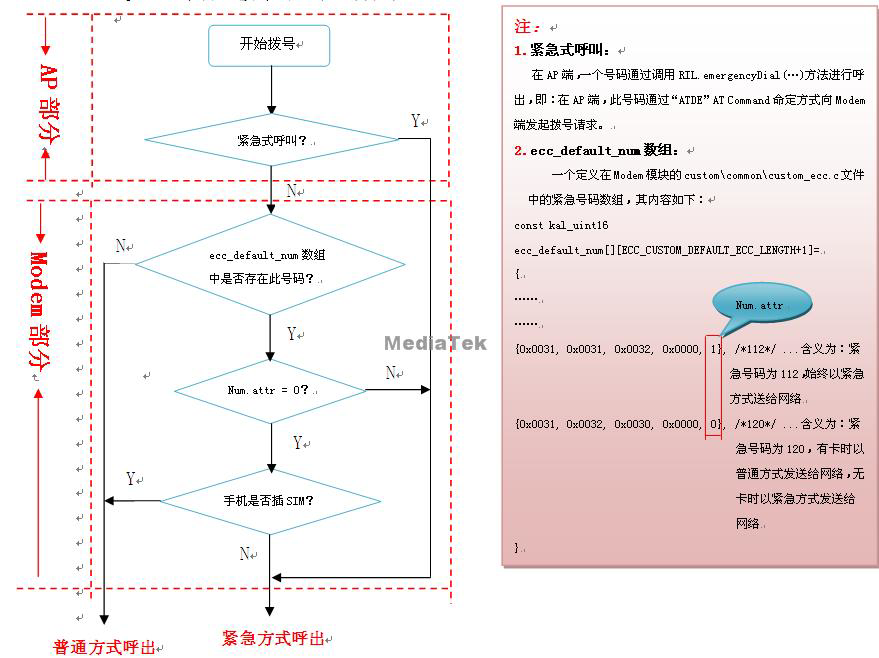
弊端：锁网文件独立于版本，如果产线漏了锁网则无法补救。要使用此方法则需要放开META读取NVRAM\_EF\_SML\_DEFAULT的条件（默认不能查看读取），则有一定风险被破解。

详细资料参考

# 11、拨号

## (1) 紧急拨号

紧急号码的客制化分为AP(android)端和modem端两部分。以下是Smartphohe平台上拨号的一个逻辑流程.



[SOLUTION]

根据上述拨号逻辑流程，可知：

【Ⅰ】如果一个紧急号码要求不管在什么情况下，都以“紧急方式呼出”，则可采取如下两种修改方法：

a.只修改AP部分，即：AP端采取“紧急式呼叫”便可。

b.修改AP和Modem两部分。AP端采用“非紧急式呼叫”进行客制化，而Modem端的客制化请参考如下Modem部分的修改。

【Ⅱ】如果一个紧急号码要求在手机有卡时，以“普通方式呼出”，在无卡时，以“紧急方式呼出”，则可采取如下修改方法：

a. 修改AP和Modem两部分。AP端采用“非紧急式呼叫”进行客制化，而Modem端的客制化请参考Modem部分的修改。

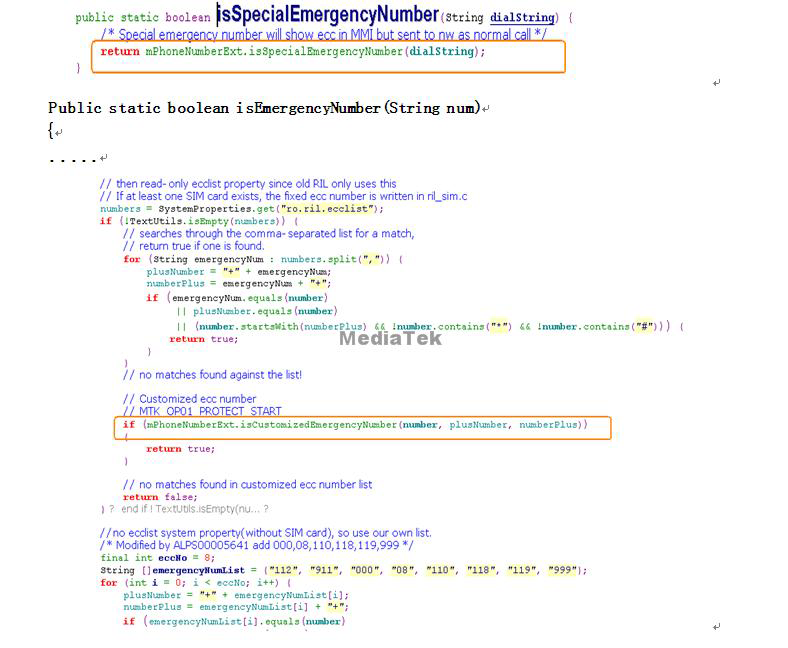
※注意事项

1.客制化紧急拨号功能时，注意不同软件版本的区别：

A.对于GB, GB2, ICS, ICS2这四个版本，请直接参考FAQ中修改

B．对于GB3,JB,JB3及以上版本，由于在phoneNumberUtils.java文件中，对isSpecialEmergencyNumber(..)和isEmergencyNumber(…)方法定义有所改变，如下(红色框标识差异的地方)：

……



……

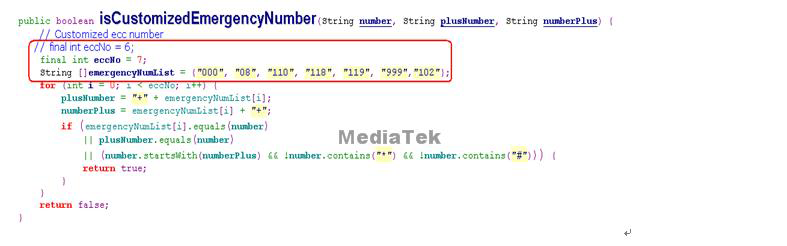
}

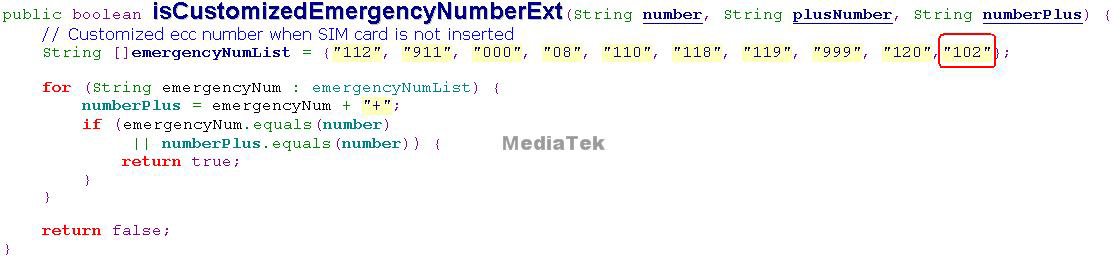
针对以上不同点，在客制化紧急号码时，请采取如下方式去修改以上两方法的不同点：

（1）先查看mediatek\operator\Opxxx(OP01、OP02、。。。)\frameworks\java\com

\mediatek\op\telephony目录下是否有PhoneNumberExtOPxxx.java文件，如有，就直接修改此文件就行，否则就修改mdiatek\frameworks\base\op\java\com\mediatek\op\telephony目录下的PhoneNumberExt.java文件。

（2）针对PhoneNumberExtOPxx.java文件或PhoneNumberExt.java文件，如有如下方法，请按下述方式修改（红色框标识修改的地方）：





※附录

〓.解读framework\base\telephony\java\android\telephony\

PhoneNumberUtils.java文件中的isEmergencyNumber()方法：

§.在拨号时,屏幕上是否应该显示为紧急拨号界面。它的唯一判断标准就是此方法。该方法返回为true,则拨号界面会显示成紧急拨号界面。否则显示为正常拨号界面。

§.在此方法中,会从3个地方来读取紧急号码：

1) SIM卡(ril.ecclist, ril.ecclist2) 。读取的是存储在SIM卡中的紧急号码，仅当插了SIM卡的情况下才可能读出数据。无法向此property中写

入数据，因为每次开机都会重新改写此property。

2) RILD(ro.ril.ecclist)。这是一个只读的property，里面写入的紧急号码为112和911，此property内容无法更改。

3) 数组(emergencyNumList)。如果需要添加或删除紧急号码，则在此数组中修改

详细参考**[FAQ02892] 紧急号码的管理及客制化方法**

## (3) MTK平台CC相关的Log查询

在外场问题中，经常会出现通话相关的故障。这里简单总结一下通话相关log的分析点：

**主叫方：**

主叫方，是指主动发起通话的一方。

1. 初略定位问题，

用户发起通话时，AP端的拨号指令最终会通过AT到达modem，所以可以通过查看radio\_log中相关的拨号AT指令来判断问题出现在AP还是BP。

11-04 11:06:06.397 484 487 D AT : AT> ATD13711349140;

11-04 11:06:06.397 484 487 D AT : ATD13711349140;

如果出现问题的时间点没有对应的ATD指令，可以推断AP端没有向Modem发送拨号指令，基本可以断定是AP的问题；如果ATD已经下发，可以将问题归属到modem端。

1. AP端，

定位AP端的问题主要根据main\_log来跟踪代码的执行流程。

在MTK Androd平台上大致的执行流程如下：

OutgoingCallBroadcast.java

PhoneGlobals.getInstance().callController.placeCall(intent);

CallController.java

CallStatusCode status = placeCallInternal(intent);

PhoneUtils.placeCall (\*\*\*);

PhoneUtils.java

Connection = app.mCM.dial(phone, numberToDial);

CallManager.java

Result = basePhone.dial(dialString);

GsmPhone.java

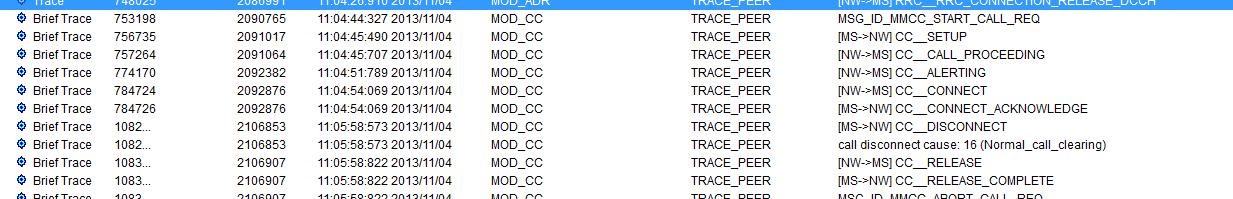
Return mCT.dial(newDialString, uusInfo);

cm.dial(\*\*\*\*\*, obtainCompleteMessage(EVENT\_DIAL\_CALL\_RESULT));

此后，AP端通过RIL.java将拨号发送至libril，最终到达modem。

1. Modem端，

Modem这边的问题主要通过modem log来排查：



图表 A MT端正常Modem log示例

PS：可以通过AT+ECPI来查看当前的CC的状态。其中，各字段的值可以参考MTK AT文档。

掉话：

**被叫方：**

1. 粗略定位：

Modem接收到寻呼消息时，会通过检测TMSI来判断是否是对于自身的寻呼。如果是，对其进行响应，并向AP上报RING。

11-04 10:54:09.154 482 497 D AT :

11-04 10:54:09.154 482 497 D AT : RING

11-04 10:54:09.154 482 497 D AT : AT< RING

11-04 10:54:09.154 482 497 D AT : RIL\_URC\_READER:RING

11-04 10:54:09.154 482 497 D AT : RIL\_URC\_READER Enter processLine

11-04 10:54:09.154 482 497 D RIL : Nw URC:RING

11-04 10:54:09.154 482 497 D RIL : receiving RING!!!!!!

11-04 10:54:09.154 482 497 D RIL : receiving first RING!!!!!!

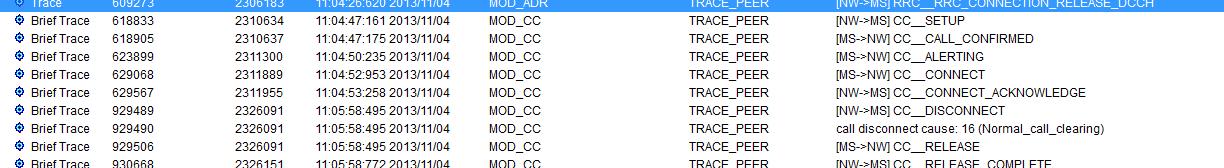
1. AP端，

RIL.java中接收到下层上报的RING信息之后，会通知CallTracker.java。此中，会将来电信息通知到在其中注册的Handler。

一般而言，这期间一般没有什么问题。

1. Modem端，

正常MT端的modem log如下：



图表 B MO端正常Modem log示例

通常的问题有如下：

* 没有接收到Paging消息：

[NW->MS] RRC\_\_PAGING\_TYPE1

[3G SMART PAGING] Smart Paging Enable = 1,Level2 Enable = 0,Learning CN Domain = 2,SmartPaging\_Available = 0,is\_Risk\_RNC = 1

PGD\_TMSI: 4B 54 B 1E

Paging record 2 is for this UE.

CN Paging:CS\_DOMAIN,TMSI\_TYPE, RRC\_PagingCause\_terminatingConversationalCall

RRCE: Starts the timer RRCE\_PendingPagingTmr\_TIMER\_ID for 3 seconds and 0 milliseconds

MSG\_ID\_RATCM\_RRCE\_PAGE\_IND

MSG\_ID\_MM\_RATCM\_PAGE\_IND

MS is paged with TMSI\_TYPE

[MS->NW] MM\_\_PAGING\_RESPONSE

以上是UE对Paging的正常响应，可以通过参看MM\_\_PAGING\_RESPONSE消息来检测UE是否对问题时间点的寻呼进行响应。

PS： TMSI是MT与网络交互的过程中，网络给MT发送的一个临时的识别码。modem log中可以查看到：

[RRM] TMSI value 4b54b1e

# 12、网络漫游配置

## (1) 何为漫游

**1. 关于漫游**

A．Roaming：

Spec定义的roaming是指Operator A的sim卡注册到了Operator B的网络plmn上-------如果这部分需要特殊处理成非roaming，请参考：

ID: FAQ06637 [NW]网络漫游客制化 (手机显示漫游图标)

B. International Roaming：

Operator A的sim卡注册到了和Operator A不同国家的Operator的plmn上；

operator用MCC/MNC来区分；MCC: mobile country code；MNC：mobile network code

operator的plmn的构成也是MCC/MNC的结构

所以International Roaming<=>SIM卡所属operator的MCC ！=此卡当前所注册plmn的MCC

**2.如何知手机发生了international roaming**

1. AP NW FW的主体逻辑在GsmServiceStateTracker.java中；其中的key method是 pollState，当网络发生变化时就会call此方法，对应处理方法是handlePollStateResult

2. 在handlePollStateResult中有 roaming这个变数如果漫游就会被置为true；

3. 在handlePollStateResult中的case EVENT\_POLL\_STATE\_OPERATOR:下opNames[2]记录的就是当前网络的plmn，其中前三位就是此plmn对应的MCC

4. 在SIMRecords.java中有方法getSIMOperatorNumeric可以获取卡的MCCMNC，然后其前三位就是MCC；在GsmServiceStateTracker.java中的updateSpnDisplay中有使用此方法的demo code，您可以参考

至此，您需要的info都已经有了，您只需要在handlePollStateResult方法中的roaming变数后面判断在roaming为true时当前网络的plmn对应MCC和卡的MCC如果不一样您就发出一个broadcast来通知或者做弹框之类的提示

## (2) 如何区别和取消漫游

**1. 漫游判定的flow：**

在GsmServiceStateTracker.java中如果网络发生变化就会call pollState方法去获取最新的网络状况，response用handlePollStateResult处理，其中roaming的判定modem通过+CREG上报：



**2. 漫游客制化方法：**

对于JB\*, KK\*及以后的版本:

在regCodeIsRoaming中有留了客制化接口，只需在customEhplmn二位数组的尾部加入一组item即可；

比如韩国operator KCT 45011的卡注册了operator SKT 45005的网络显示漫游，要求作为home plmn，不显示漫游:

Private String[][] customEhplmn = {{“46000”,“46002”, “46007”},

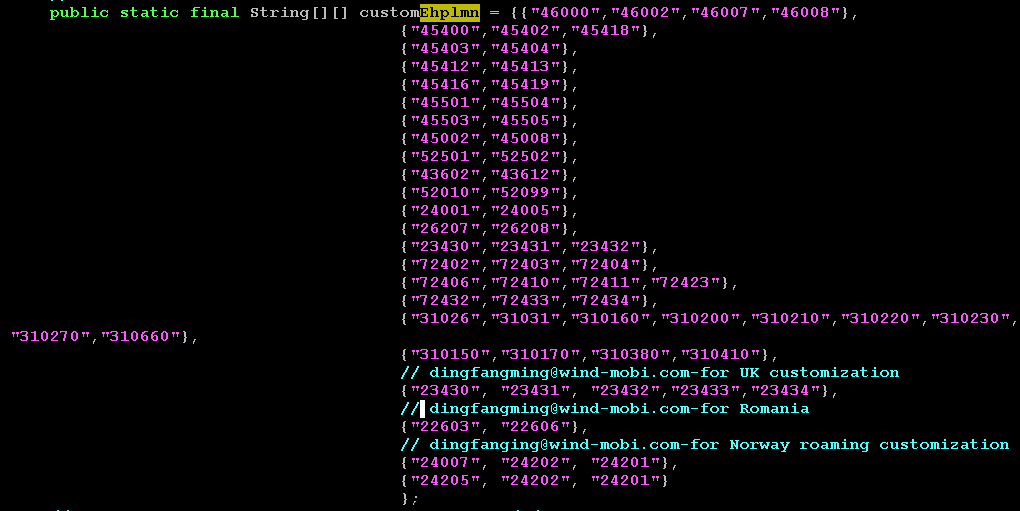
{“45400”,“45402”,“45418”},

…,

{“310150”,“310170”,“310380”,“310410”},

{“45011”,“45005”}}

例如T651W



# 13、GPS相关

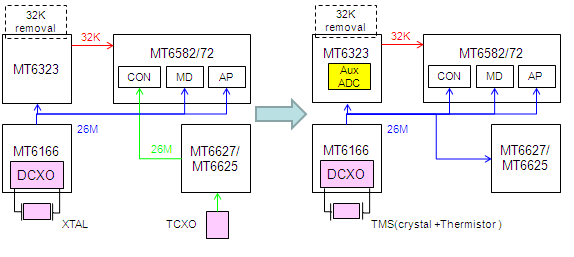
## (1) GPS coclock 配置

### A) 什么是GPS Coclock

**GPS co-clock(TMS)**

MT6627/MT6625 GPS co-clock(TMS)方案是：MT6166使用TMS(crystal+thermistor)，MT6627/MT6625省去TCXO，参考时钟由MT6166提供。

TCXO方案与TMS方案示意图



### B) GPS Coclock配置

配置文档：密码：5795959177

### C) GPS Log抓取

对于GPS问题，请提供mobile log和GPS debug log。

 log 抓取办法如下：

【gps debug log抓取方法】

1. 进到工程模式，打开YGPS。

2. 点击“NMEA Log->Enable dbg2file [Need Restart]”按钮和“NMEA Log->Enable dbg2ddms [Need Restart]”按钮。

”。

3. 退出YGPS，间隔至少3s后，再进入GPS应用程序（如导航软件，YGPS）开始测试，这样才能确保操作生效。退出YGPS的时候一定要按Home键退出，不能按返回键，否则log会停止。

4. log最后会存放在手机的/data/misc/gpsdebug.log.xxx，测试完后请将log提供给我们分析。

5. 附上log的同时也请告知复现问题的操作步骤以及复现问题的时间点，如hh:mm:ss出现什么现象。

note:

1.测试时请选择GPS信号良好的空地（视野范围内无建筑物遮挡）进行测试，以排除环境因素导致无法定位，定位慢，定位漂移等问题。

2.测试前请确保测试机有root 权限，可以访问/data/misc/路径，可以将gps debug log用adb拉出来。

3.测试完成后请注意以下事项

  a.请进入YGPS，点击“NMEA Log->disable dbg2file [Need Restart]”按钮，然后退出YGPS，这样才能确保操作生效。否则，只要打开GPS，就会录制debug log，这样会占用更多的存储空间，导致不可预测的问题发生。

  b.将/data/misc/gpsdebug.log.xxx删除，因为即使disable dbg2file以后，这笔log仍然会在/data/misc/。

【Mobile log抓取方法】

1. 进入工程模式，找到system logger进行相应设置（即选择mobile,然后按start按钮）

2. 成功后可以看到状态栏会出现“Mobile log service starts” 的五角星。

3. 测试完成后，请将SD卡中mtklog文件夹中的所有log提供给我们。

补充说明一下：

我们测试GPS的版本必须使用ENG版本，或者是USER版本把它root一下，\*983\*7668#。

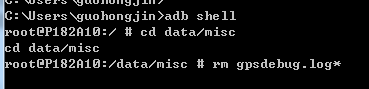
我们测试GPS的时候需要三种log，可以按照下面的方法抓取，退出YGPS的时候一定要按Home键退出，不能按返回键，否则log会停止。

1. NMEA Log 存在手机存储的根目录下面，直接拷出来就好了。
2. gps debug log在data/misc路径下，需要用adb pull出来。在电脑的D盘建一个文件夹：gps\_test。把手机的USB调试打开，然后连接adb，用adb pull /data/misc d:\gps\_test

这样就会把misc文件夹全部pull了出来，我们只需要其中的gpsdebug.log.20140505091416 这样的log。

Pull出这些log之后，要把手机中这些gpsdebug.log 全部删除掉，便于下次测试。

删除命令：



按Enter就ok了。可以输入ls指令，确认是否log已经删除。

1. mtklog就在SD卡的根目录，拷出来就好了。

### D) GPS 校准数据的备份和恢复



## (2) AGPS 服务器配置

手机上查看AGPS服务器：Settings🡪Location access🡪GPS satellites🡪A-GPS settings🡪Profile Selector

目前遇到关于AGPS的修改只有： SR-F-AGPS-0010 AGPS需求开发原则，由于我司并未和Nokia签署合作合约，因此，版本中不可以使用Nokia AGPS服务器信息

修改文件mediatek/frameworks/base/agps/etc/agps\_profiles\_conf.xml

在此文件中找到NOKIA的服务器配置，将show\_type的值修改为1。

# 内置联系人

## (1)如何内置联系人

Android 4.2

文件 AbstractStartSimService.java

onStart()函数中增加函数，例子651W上增加了importDefaultReadOnly()函数具体查看相关代码

Android 4.4

文件 SimimportProcess.java

doWork()函数中增加相应添加内置联系人函数

详细参考**[FAQ07470] 在手机中预置联系人/Service Number**

## (2)号码匹配

号码匹配位数涉及两个地方：

1.PhoneNumberUtils.java

这里有两个常量MIN\_MATCH为7，MIN\_MATCH\_CTA为11，

如果是模糊匹配，会在compareLoosely中根据MTK\_CTA\_SUPPORT\_FLAG去选择获取MIN\_MATCH还是MIN\_MATCH\_CTA

2.alps\external\sqlite\android\OldPhoneNumberUtils.cpp

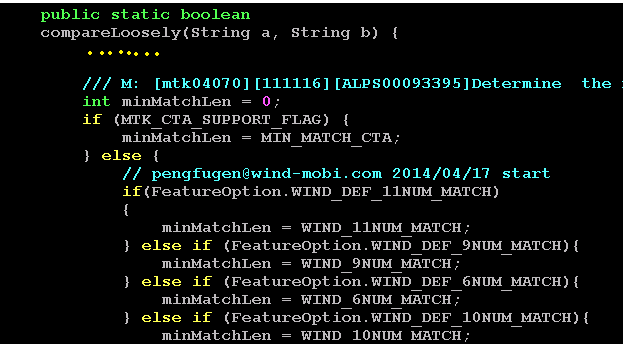
这里有个函数是getMinMatch()，同样这里有根据宏去获取是7位匹配还是11位匹配。

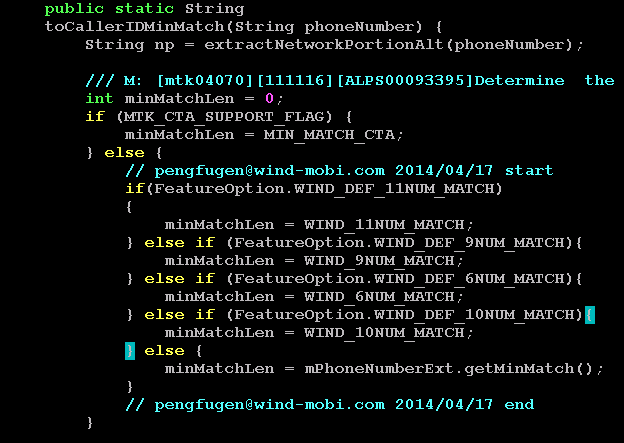
如果贵司需要客制化匹配位数，需修改以上两个地方。

下面是T651W中号码匹配的客制化，下图是ProjectConfig.mk

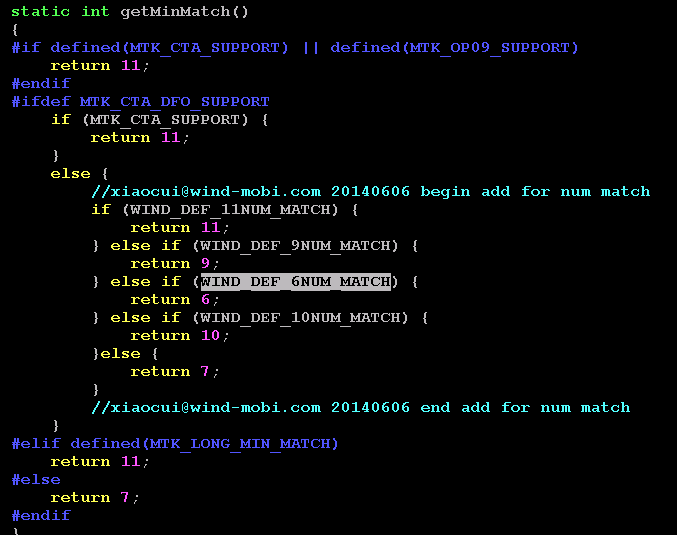


PhoneNumberUtils.java





OldPhoneNumberUtils.cpp



详细资料参考

**[FAQ11487] 信息部分号码匹配位数的说明**

**[FAQ05195] 根据mccmnc实现号码匹配自适应**

# 15、短信相关

## (1) 短信中心号码

文件 SmsPreferenceActivity.java

OnPreferenceTreeClick函数，变量gotScNumber进行赋值，

mNumberText.setInputType(EditorInfo.TYPE\_NULL);可以设置短信中心号码的只读

常见问题：短信可以接受，但是无法发送。基本就是短信中心号码错误。

## (2) 内置小区广播号码

具体要参考FAQ06820，增加database操作函数，增加chanle函数，以及UI函数

注相关设置不同频道不同声音

CBMessageNotification.java

getCBNewMessageNotificationInfoBase获取相关频道

updateNotification进行不同频道不同铃声，具体请参考T651W，

## (3) 8Bit

android默认不支持8bit编码，8bit只是数据传输的一种方式，8bit编码并没有定义唯一的字符编码表来指明某个编码表示某个字符，所以没办法唯一去解析它

SmsMessage.java函数parseUserData中增加8bit处理方法，原函数8bit处理直接为null处理，具体请查看T651W

## (4) 7Bit

frameworks/base/core/res/res/value/config.xml

然后只要将相应的编号填入single shift table 或者 locking shift table

一般配置single shift table就可以

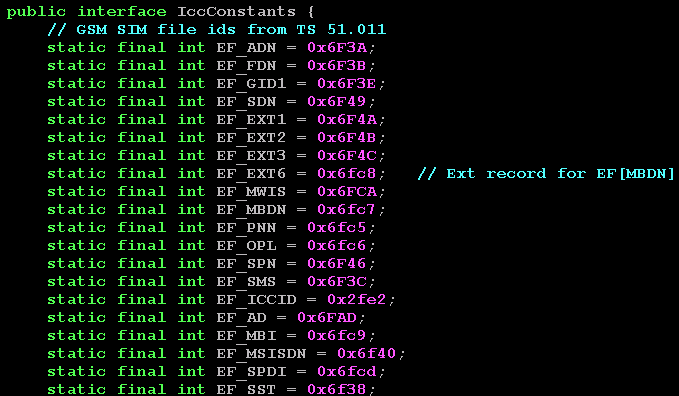
## (5) Single Shift 和 Locking Shift

在实现R8 features的情况下，土耳其/西班牙/葡萄牙等国家，对本国的字符做了类似GSM 7 bit Default Alphabet和GSM 7 bit default alphabet extension table的两个表，分别叫做locking\_shift 和 single\_shift ，其中locking\_shift 和GSM 7 bit Default Alphabet类似，single\_shift和GSM 7 bit default alphabet extension table类似

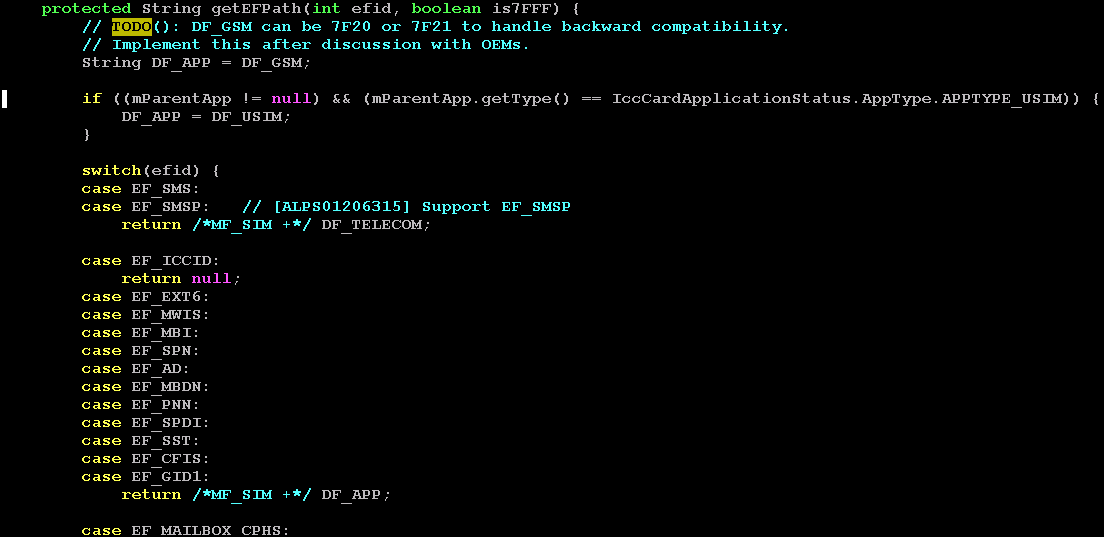
# 16、SIM文件的读写支持

## (1) 如何查看当前代码支持的SIM卡文件

SIM卡相关文件定义在IccConstants.java 如下图。

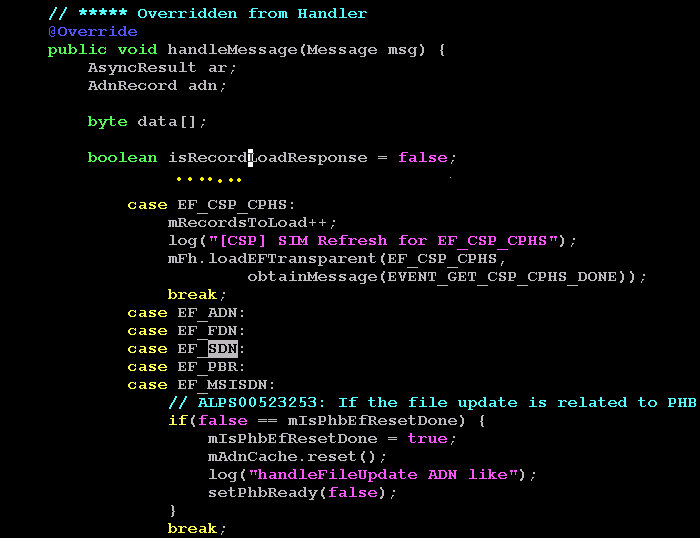


对SIM文件的处理在SimFileHandler.java / UsimFileHandler.java



通过查询getEFPath(int, Boolean) 可以查看哪些SIM文件被读取处理了。

在SIMRecords.java中的handleMessage对读取到的SIM文件进行处理。



详细参考

**[FAQ08976] [SIM]如何读取ICC Card中的文件**

## (2) 根据SIM卡文件显示STK名称

A)、参考**[FAQ05657] [SAT]如何动态修改SIM卡应用名称**

修改文件：ApplicationPackageManagerl.java

B)、修改应用程序的快捷方式名称：

修改文件：

packages\apps\launcher2\src\com\android\launcher2\iconCache.java

具体修改：

将cacheLocked()方法中, 通过特殊判断将要显示的名字赋值给entry.title。

同理这里也要区分上面两种情况。

# 17、MTK默认的显示运营商名称

## (1) MTK添加运营商名称的缘由

MTK目前的design，现在的命名规则是这样的，如果是第一次插入的卡，会根据手机里的网络名称来设置sim卡的名字为中国联通或其他运营商的名字，但同一个卡槽如果再插入另外一张中国联通的卡，为了跟前一张卡区别，就会自动加上01，02等用来跟之前插入的卡区别,并且运营商名称的背景图片的颜色也会发生改变。

有些项目是单卡定制,要求运营商名称和背景颜色不发生改变,需要修改一下文件：

[mediatek/frameworks](http://10.0.30.6:8081/#/q/status:merged+project:MT6582_4_2/mediatek/frameworks,n,z)/[base/telephony/java/com/mediatek/telephony/SimInfoManager.java](http://10.0.30.6:8081/#/c/28056/1/base/telephony/java/com/mediatek/telephony/SimInfoManager.java)

修改下面的三个处理函数

public static int setDefaultName(Context ctx, long simInfoId, String name)

public static int setDefaultNameEx(Context ctx, long simInfoId, String name, long nameSource)

private static SimInfoRecord fromCursor(Cursor cursor)

## (2) 如何去掉运营商名称的显示

SMS 、 Calllog、 Contact、 Settings

**Android4.2**

1. 去掉：信息列表和收到新信息时的提示框中的运营商名称

[packages/apps/Mms](http://10.0.30.6:8081/#/q/status:merged+project:MT6582_4_2/platform/packages/apps/Mms,n,z)/[src/com/android/mms/ui/MessageListItem.java](http://10.0.30.6:8081/#/c/28352/1/src/com/android/mms/ui/MessageListItem.java)

[packages/apps/Mms](http://10.0.30.6:8081/#/q/status:merged+project:MT6582_4_2/platform/packages/apps/Mms,n,z)/[src/com/android/mms/ui/DialogModeActivity.java](http://10.0.30.6:8081/#/c/28352/1/src/com/android/mms/ui/DialogModeActivity.java)

1. 去掉：拨号盘界面中的运营商名称

[packages/apps/Contacts](http://10.0.30.6:8081/#/q/status:merged+project:MT6582_4_2/platform/packages/apps/Contacts,n,z)/[src/com/mediatek/contacts/ContactsFeatureConstants.java](http://10.0.30.6:8081/#/c/28404/1/src/com/mediatek/contacts/ContactsFeatureConstants.java)

[packages/apps/Contacts](http://10.0.30.6:8081/#/q/status:merged+project:MT6582_4_2/platform/packages/apps/Contacts,n,z)/[src/com/mediatek/contacts/dialpad/DialerSearchAdapter.java](http://10.0.30.6:8081/#/c/28404/1/src/com/mediatek/contacts/dialpad/DialerSearchAdapter.java)

1. 去掉：通话界面(去电和来电)中的运营商名称

[packages/apps/Phone](http://10.0.30.6:8081/#/q/status:merged+project:MT6582_4_2/platform/packages/apps/Phone,n,z)/[src/com/android/phone/CallCard.java](http://10.0.30.6:8081/#/c/28562/1/src/com/android/phone/CallCard.java)

1. 去掉：通话记录中的运营商名称

[packages/apps/Contacts](http://10.0.30.6:8081/#/q/status:merged+project:MT6582_4_2/platform/packages/apps/Contacts,n,z)/[src/com/android/contacts/PhoneCallDetailsHelper.java](http://10.0.30.6:8081/#/c/28545/1/src/com/android/contacts/PhoneCallDetailsHelper.java)

**Android4.4**

1. 去掉：信息列表和收到新信息时的提示框中的运营商名称

[packages/apps/Mms](http://10.0.30.6:8081/#/q/status:merged+project:MT6582_4_2/platform/packages/apps/Mms,n,z)/[src/com/android/mms/ui/MessageListItem.java](http://10.0.30.6:8081/#/c/28352/1/src/com/android/mms/ui/MessageListItem.java)

[packages/apps/Mms](http://10.0.30.6:8081/#/q/status:merged+project:MT6582_4_2/platform/packages/apps/Mms,n,z)/[src/com/android/mms/ui/DialogModeActivity.java](http://10.0.30.6:8081/#/c/28352/1/src/com/android/mms/ui/DialogModeActivity.java)

1. 去掉：拨号盘列表中的运营商名称

去掉：通话记录中的运营商名称

[packages/apps/Dialer](http://10.0.30.6:8081/#/q/status:merged+project:MT6592_KK/platform/packages/apps/Dialer,n,z)/[src/com/android/dialer/CallDetailActivity.java](http://10.0.30.6:8081/#/c/35776/1/src/com/android/dialer/CallDetailActivity.java)

[packages/apps/Dialer](http://10.0.30.6:8081/#/q/status:merged+project:MT6592_KK/platform/packages/apps/Dialer,n,z)/[src/com/android/dialer/list/DialerPhoneNumberListAdapter.java](http://10.0.30.6:8081/#/c/35776/1/src/com/android/dialer/list/DialerPhoneNumberListAdapter.java)

[packages/apps/Dialer](http://10.0.30.6:8081/#/q/status:merged+project:MT6592_KK/platform/packages/apps/Dialer,n,z)/[src/com/mediatek/dialer/PhoneCallDetailsHelperEx.java](http://10.0.30.6:8081/#/c/35776/1/src/com/mediatek/dialer/PhoneCallDetailsHelperEx.java)

[packages/apps/Dialer](http://10.0.30.6:8081/#/q/status:merged+project:MT6592_KK/platform/packages/apps/Dialer,n,z)/[src/com/mediatek/dialer/dialpad/DialerSearchAdapter.java](http://10.0.30.6:8081/#/c/35776/1/src/com/mediatek/dialer/dialpad/DialerSearchAdapter.java)

1. 去掉：通话界面(去电和来电)中的运营商名称

[packages/apps/InCallUI](http://10.0.30.6:8081/#/q/status:merged+project:MT6592_KK/platform/packages/apps/InCallUI,n,z)/[src/com/android/incallui/CallCardFragment.java](http://10.0.30.6:8081/#/c/35777/1/src/com/android/incallui/CallCardFragment.java)

[packages/apps/ContactsCommon](http://10.0.30.6:8081/#/q/status:merged+project:MT6592_KK/platform/packages/apps/ContactsCommon,n,z)/[src/com/mediatek/contacts/ContactsFeatureConstants.java](http://10.0.30.6:8081/#/c/35778/1/src/com/mediatek/contacts/ContactsFeatureConstants.java)

# 18、FOTA

**(1) FOTA 检查版本**

ZTE使用的FOTA apk 是有中兴业软提供的，如果需要修改请让需求协助。

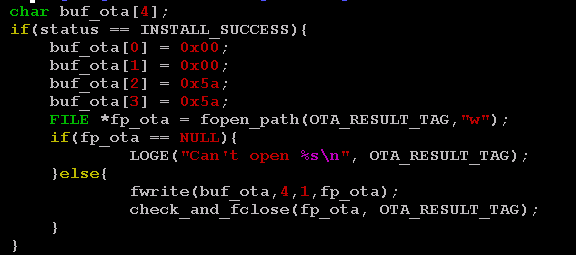
注意事项：

FOTA 最新的APK增加了Model校验功能，即：每个FOTA apk都内置限定了Model。如果检测到当前手机的Model和APK内置的Model不一致话，会提示无法升级。请联系需求提供对应的apk.

**(2) Recovry修改**

***A)、FOTA升级成功后，提示升级失败。***

FOTA升级状态成功与否，是从QQ截图20140801164055.png文件中的标志位读取出来的，所以在recovery.cpp文件中要加入标志位的写入：



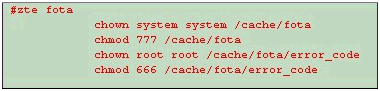
***B)、T卡升级后，再进行FOTA升级，会提示升级失败的问题***

问题的大概原因：

如第一次先进行SD卡升级，在进入recovery模式后，SD卡升级完成后如果不存在error\_code，则会创建该文件，创建的文件是有root账户创建的，且采用的默认方式创建，所以导致该文件只能由root权限的用户才能访问。则会导致system/shell用户无法读取。之后进行fota手机，由于fota应用的用户权限是system，无法访问error\_code，就会在升级成功后提示升级失败。

修改方法：

由recovery在创建error\_code的时候强制修改该文件权限为666或者777，或者在系统初始化的时候修改：



**(3) 服务器配置**

ZTE海外项目，都放在dms.ztems.com服务器上。   
[http://dms.ztems.com:81/portal/uiloader/index.html](http://dmcn.ztems.com:81/portal/uiloader/index.html)

和FOTA相对应，服务器上的项目名称也是靠Model名称来识别的。如果修改了Model名称需要项目经理重新在服务器上配置对应的项目。

# 19、文件管理和视频播放

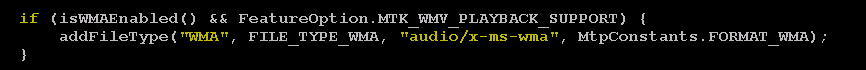
如何屏蔽指定的文件格式

修改文件：



只要能保证不运行addFileType就可以屏蔽掉次文件格式。

例如把MTK\_WMV\_PLAYBACK\_SUPPORT宏设置为false，那么WMA格式就会被屏蔽。



# 20、特殊的号码匹配（委内瑞拉）

丁方明/李经纬

# 21、特殊虚拟运营商的APN识别

孙安国