Application Notes

# Amlogic MBOX API

Revision 0.1



Amlogic, Inc.
2518 Mission College Blvd
Santa Clara, CA 95054
U.S.A.
www.amlogic.com

# 目录

# 修改记录

| 版本 | 日期 | 作者 | 修改 |
|---|---|---|---|
| 0.1 | Oct 24th, 2013 | Lei Qian | 初稿 |
| | | | |
| | | | |

# 1. 开机流程

## 1.1. 开机 logo 和动画

### 1.1.1. 实现原理

**（1） A9 logo，M8 logo:**

将一张 bmp 图片通过 framebuffer 显示, 代码具体参见 common/drivers/amlogic/display/aml_logo 目录

**（2） android 机器人：**

init 进程加载的启动画面，具体参考 system/core/init/init.c 中 load_565rle_image 的实现

**（3） 开机动画：**

bootanimation 动画，init.rc 中启动 service：

```
service bootanim /system/bin/bootanimation
    user graphics
    group graphics
    disabled
    oneshot
```

bootanimation 的实现参考 frameworks/base/cmds/bootanimation/目录，这个进程会显示一段动画

### 1.1.2. 怎么更换开机 logo 和动画

**（1） A9 logo, M8 logo:**

M8和 M6都是一样的方法。首先，制作一张分辨率为1920 x 1080的未压缩的 bmp 图片，把它的名字改成“bootup”。然后，将 device 下对应项目中 res_pack 目录的 bootup 文件（device/amlogic/g18/res_pack/bootup）替换成前面自己做的 bootup 文件。

**（2） Android 机器人：**

M6只使用了一张720P 的 logo；M8使用了两张 logo(720P 和1080P)。

首先 M6制作一张1280 x 720大小的 png 图片，M8制作1920x1080和1280x720大小的两张 png 图片,保存为 logo.png。然后，在 ubuntu 系统中安装 imagemagick 工具，执行

```
sudo apt-get install imagemagick
```

再使用 imagemagick 自带的 convert 命令，进行 raw 格式转换，执行

```
convert -depth 8 logo.png rgb:logo.raw
```

再使用 Android 自带的 rgb2565工具,对 raw 文件进行 rle565格式转换，到源码下执行

```
Android_rootfs/out/host/linux-x86/bin/rgb2565 -rle <logo.raw >initlogo.rle
```

如果是由720P 图片做成的，就把“initlogo.rle”改名为“initlogo-robot-1280x720.rle”；

如果是由1080P 图片做成的，就把“initlogo.rle”改名为“initlogo-robot-1920x1080.rle”；

最后用得到的 rle 文件，将 device 对应项目（如 device/amlogic/g18ref）下同名的文件替换掉就好了。

**（3） 开机动画：**

开机动画文件分成三部分，首次出现的动画，loading 动画和配置文件 desc.txt。

首先制作多张连续的 png 图片（建议分辨率不小于454x200，且必须以阿拉伯数字按顺序命名),将首次出现的动画图片（amlogic 自带的是31张）放到名字为“android”的文件夹，将 loading 动画图片（amlogic 自带的是 49张）放到名字为“loading”的文件夹。

然后，把这两个文件夹和 decs.txt 以“文件打包（仅存储）”的方式打包成一个 zip（注意删除掉 zip 包中，打包生成的多余文件，如 Thumbs.db）。最后替换掉 device 对应项目的 bootanimation.zip。

desc.txt 格式解析，以 Amlogic 自带 bootanimation.zip 为例：

454 200 20

p 1 0 android

p 0 10 loading

第一行： 454代表动画图片宽度；200代表图片高度；20代表每20毫秒就播放下一张图片

第二行： p 为标志符，不用改；1代表只播放1次；0代表播放这部分动画前延迟0毫秒；android 代表第一部分 动画所在的目录名字；

第三行：p 为标志符，不用改；0代表循环播放，知道开机起来为止；10代表播放一次这部分动画前延迟10毫秒； loading 代表第二部分动画所在的目录名字

## 1.2. HDMI 和 CVBS 切换

### 1.2.1. 简介

Amlogic MX 公版默认的输出方式是单输出（HDMI 输出或 CVBS 输出），通过下面的输出规则确定 MBX 的输出模式：

① 检测到有 HDMI 接入时，自动关闭 CVBS 输出，打开 HDMI 输出；

② 检测到 HDMI 没有接入时，自动打开 CVBS 输出。

根据上面的输出规则，在拔插 HDMI 线时，MBX 会自动切换输出。

通过读文件 sys/class/amhdmitx/amhdmitx0/hpd_state 的值可以确定 HDMI 是否接入。值为 1 时，表示 HDMI 接入，为 0 时，HDMI 没有接入。

打开或关闭 CVBS 输出的方法如下：

① 打开 CVBS 输出：写字符串”vdac”到文件 sys/class/aml_mod/mod_on；

② 关闭 CVBS 输出：写字符串”vdac”到文件 sys/class/aml_mod/mod_off。

### 1.2.2. Frameworks 相关广播

文件 frameworks/base/policy/src/com/android/internal/policy/impl/PhoneWindowManager.java 中，与 HDMI 接入状态相关的广播为 WindowManagerPolicy.ACTION_HDMI_HW_PLUGGED。

在文件 PhoneWindowManager.java 中，当获取到 uevent 事件”DEVPATH=/devices/virtual/switch/hdmi”时，就会调用 setHdmiHwPlugged 函数发送 WindowManagerPolicy.ACTION_HDMI_HW_PLUGGED 广播。得到 HDMI 是否接入的代码示例：

```
public void onReceive(Context context, Intent intent) {
.....
    if (WindowManagerPolicy.ACTION_HDMI_HW_PLUGGED.equals(intent.getAction())) {
        boolean plugged =
intent.getBooleanExtra(WindowManagerPolicy.EXTRA_HDMI_HW_PLUGGED_STATE, false);
.....
```

### 1.2.3. HDMI/CVBS 切换

接收到 WindowManagerPolicy.ACTION_HDMI_HW_PLUGGED 广播并得到 HDMI 接入状态后，就可以设置新的输出分辨率并切换输出。关于设置输出分辨率的方法，可以参考“多分辨率切换及重显示率”的部分。

**（1）切换到 CVBS 输出**

在 HDMI 输出状态下，拔下 HDMI 线，按如下步骤切换到 CVBS 输出。

① 通过系统属性 ubootenv.var.cvbsmode 得到 CVBS 输出的分辨率，并设置该分辨率；

② 写字符串”vdac”到文件/sys/class/aml_mod/mod_on，打开 VDAC；

③ 写字符串”0”到文件/sys/class/graphics/fb0/blank，打开 OSD。

**（2） 切换到 HDMI 输出**

在 CVBS 输出状态下，插上 HDMI 线，按如下步骤切换到 HDMI 输出。

① 写字符串”vdac”到文件/sys/class/aml_mod/mod_off，关闭 VDAC；

② 通过系统属性 ubootenv.var.hdmimode 得到 HDMI 输出的分辨率，并设置该分辨率；

③ 写字符串”0”到文件/sys/class/graphics/fb0/blank，打开 OSD。

## 1.3. HDMI 多分辨率原理

- MX 平台

MX 平台只有一种基础分辨率 ---1280 x 720。720制式（包括720p50hz,720p6hz）是原始的720p native 显示，其他分辨率都是以720P 为基础，通过 scale 来达到显示目的的，pixel density 为160。

# 2. 开机启动部分

## 2.1. Mbox Launcher 添加默认 APP 快捷方式

Mbox launcher 共有5个界面可更换默认 APP 快捷方式（首页快捷栏，在线视频，我的推荐，音乐，本地播放）。它们全都是通过 device/amlogic/g18ref/default_shortcut.cfg 这个文件配置的。

### 2.1.1. default_shortcut.cfg 内容解析

以下面的内容为例：

```
Home_Shortcut:com.farcore.videoplayer;com.android.music;app.android.applicationxc;
Video_Shortcut:com.youku.tv;com.qipo;com.togic.livevideo;com.moretv.tvapp;
Recommend_Shortcut:com.amlogic.mediacenter;com.example.airplay;com.amlogic.miracast;ol;
Music_shortcut:st.com.xiami;com.android.music;com.hycstv.android;
Local_Shortcut:com.android.gallery3d;com.farcore.videoplayer;com.fb.FileBrower;
```

“Home_Shortcut:”代表 首页快捷栏界面
“Video_Shortcut:” 代表 在线视频界面
“Recommend_Shortcut:”代表 我的推荐界面
“Music_shortcut:”代表 音乐界面
“Local_Shortcut:” 代表 本地播放界面

在这些标题的后加需要添加的 APK 的包名，就代表在对应的界面下显示这个 APK 的快捷方式。例如“Video_Shortcut:com.youku.tv;”的意思是， 将优酷显示到在线视频界面。

### 2.1.2. 注意事项：

（1）分隔符： default_shortcut.cfg 里面标题项后面不能缺少“:”,包名的后面不能缺少“;”
（2）标题项和包名的必须准确，不能有错误
（3）default_shortcut.cfg 的行数必须只有5行，不能因为某个界面的包名太长而给它换行一下。

## 2.2. Mbox Launcher 怎么显示天气地点

### 2.2.1. 原理：

如果系统内有 APK 发送名字为“android.amlogic.settings.WEATHER_INFO”的广播，launcher 就会响应到， 然后检索这个广播包含的数据，并将其显示在 Home 界面上。

现在默认是由 setting 发送天气数据。客户如果有自己的天气服务，那么可以按照下面的方式发送广播。

### 2.2.2. 发送广播的格式：

```
Intent i = new Intent();
i.setAction("android.amlogic.settings.WEATHER_INFO");
i.putExtra("weather_today", “上海,20℃,0.gif”);
sendBroadcast(i);
```

其中“0.gif”代表天气“晴”，天气对照表如下：

| | |
|---|---|
| 晴 | 0.gif |
| 多云 | 1.gif |
| 阴 | 2.gif |
| 阵雨 | 3.gif |

| | |
|---|---|
| 雷阵雨 | 4.gif |
| 雷阵雨并伴有冰雹 | 5.gif |
| 雨夹雪 | 6.gif |
| 小雨 | 7.gif |
| 中雨 | 8.gif |
| 大雨 | 9.gif |
| 暴雨 | 10.gif |
| 大暴雨 | 11.gif |
| 特大暴雨 | 12.gif |
| 阵雪 | 13.gif |
| 小雪 | 14.gif |
| 中雪 | 15.gif |
| 大雪 | 16.gif |
| 暴雪 | 17.gif |
| 雾 | 18.gif |
| 冻雨 | 19.gif |
| 沙尘暴 | 20.gif |
| 小雨-中雨 | 21.gif |
| 中雨-大雨 | 22.gif |
| 大雨-暴雨 | 23.gif |
| 暴雨-大暴雨 | 24.gif |
| 大暴雨-特大暴雨 | 25.gif |
| 小雪-中雪 | 26.gif |
| 中雪-大雪 | 27.gif |
| 大雪-暴雪 | 28.gif |
| 浮尘 | 29.gif |
| 扬沙 | 30.gif |
| 强沙尘暴 | 31.gif |
| 小到中雨 | 21.gif |
| 中到大雨 | 22.gif |
| 大到暴雨 | 23.gif |
| 小到中雪 | 26.gif |
| 中到大雪 | 27.gif |
| 大到暴雪 | 28.gif |

## 2.3. 启动过程中，调用 scaler 的详细说明

### 2.3.1. 简介

MBX 支持 HDMI 和 CVBS 输出，分辨率有720x480, 720x576, 1280x720, 1920x1080, 3840x2160 等多种，在开机启动阶段，有3个 logo 显示，如果每个 logo 每种分辨率都分别配置资源的话，那不但很占空间，而且替换 logo 也会很麻烦。因此，使用了 scaler， 用一套资源通过 scaler 实现在多种分辨率下的显示。

### 2.3.2. scaler 设置相关节点和属性

#### 2.3.2.1. 节点

/sys/class/display/mode

/sys/class/display/axis

/sys/class/graphics/fb0/free_scale

/sys/class/graphics/fb0/freescale_mode

/sys/class/graphics/fb0/free_scale_axis

/sys/class/graphics/fb0/window_axis

/sys/class/ppmgr/ppscaler

/sys/class/ppmgr/ppscaler_rect

/sys/class/graphics/fb0/blank

#### 2.3.2.2. 节点访问接口

在.c .cpp 文件中使用 open/read/write 标准接口可以直接访问节点
在 java 中，要使用 SystemWriteManager 提供的接口：
sw=(SystemWriteManager)mContext.getSystemService(Context.SYSTEM_WRITE_SERVICE );

#### 2.3.2.3. ppscaler

(1) /sys/class/ppmgr/ppscaler： 0： disable; 1: enable;
echo 0 > /sys/class/ppmgr/ppscaler

echo 1 > /sys/class/ppmgr/ppscaler

(2) /sys/class/ppmgr/ppscaler_rect start_x, start_y, end_x, end_y,
echo 0 0 1919 1079 0 〉/sys/class/ppmgr/ppscaler_rect

#### 2.3.2.4. freescaler

freescaler 有 2 种模式，default 和 independent 。

/sys/class/graphics/fb0/freescale_mode 0：default； 1：independent ；

/sys/class/graphics/fb0/free_scale

default 模式： 0： disable； 1：enable

independent 模式： 0：disable both of horizontal and vertical direction freescale

0x1: enable vertical direction freescale

0x10000: enable horizontal direction freescale

0x10001: enable both of horizontal and vertical direction freescale

/sys/class/graphics/fb0/free_scale_axis

Scaler 源窗口： start_x, start_y, end_x, end_y,

/sys/class/graphics/fb0/window_axis

Scaler 目标窗口： start_x, start_y, end_x, end_y,

## 2.3.3. code 实现

### 2.3.3.1. ppscaler

- system/core/init/

开机绿色机器人 logo 是用 ppscaler 来显示的，在 load_565rle_image_mbx()函数中实现。

首先在 import_kernel_nv()中获取 uboot 传递过来的保存的制式分辨率（hdmimode 和 cvbsmode），然后读取 /sys/class/amhdmitx/amhdmitx0/hpd_state 节点，判断是 HDMI 还是 CVBS 输出，获得当前分辨率对 M8，OSD size 为1920x1080， M6 OSD size 为1280x720，Logo 资源图片也要和 OSD size 保持一致。

下面 scaler 的设置也要和 OSD size 一致：

```
write(fd_blank, "1", strlen("1"));
write(fd_daxis, "0 0 1920 1080 0 0 18 18", strlen("0 0 1920 1080 0 0 18 18"));
write(fd_faxis, "0 0 1919 1079", strlen("0 0 1919 1079"));
write(fd_vaxis, "0 0 1919 1079", strlen("0 0 1919 1079"));
```

然后根据当前分辨率设置/sys/class/ppmgr/ppscaler_rect 节点：

```
if((!strncmp(resolution, "480i", 4)) || (!strncmp(resolution, "480p", 4)) || (!strncmp(resolution, "480cvbs", 7)))
{
    write(fd_ppscale_rect, "0 0 719 479 0", strlen("0 0 719 479 0"));
}
else if(!strncmp(resolution, "720p", 4))
{
        write(fd_ppscale_rect, "0 0 1279 719 0", strlen("0 0 1279 719 0"));
    }
.....
```

最后 enable scaler：

```
write(fd_freescale, "1", strlen("1"));
write(fd_ppscale, "1", strlen("1"));
write(fd_blank, "0", strlen("0"));
```

- device/amlogic/k200/set_display_mode.sh

对于 M6，一直是使用 ppscaler， 在前面 system/core/init 中已经设置好了，不过考虑到重显率的问题，需要重新设置下：

```
busybox echo $outputx $outputy $(($outputwidth + $outputx - 1)) $(($outputheight + $outputy - 1)) 0 >
        /sys/class/ppmgr/ppscaler_rect
```

```
echo 2 > /sys/class/graphics/fb0/request2XScale
echo 1 > /sys/class/graphics/fb0/free_scale
echo 1 > /sys/class/graphics/fb1/free_scale
```

对于 M8， MBOX 动画 logo，也是用 ppscaler，但要注意设置下/sys/class/display/axis 节点，对 720 及以下的分辨率用 1280x720，对 1080 及以上的分辨率用 1920x1080:

```
if [ "$scaledir" = "down" ]; then
  echo 0 0 1280 720 $outputx $outputy 18 18 > /sys/class/display/axis
else
  echo 0 0 1920 1080 $outputx $outputy 18 18 > /sys/class/display/axis
Fi
```

#### 2.3.3.2. freescaler

- frameworks/base/

在 android 启动完成后进入 launcher，对于 M8 ，720/1080 分辨率是不用 scaler，而 480/576/4k2k 则是用 freescaler 来实现显示的， M6 则不需要再设置。在进入 launcher 前要设置好 scaler，具体实现 M8 是在 frameworks\base\services\java\com\android\server\wm\WindowManagerService.java 的 performEnableScreen()中

首先关闭 scaler：

```
sw.writeSysfs("/sys/class/ppmgr/ppscaler","0");
sw.writeSysfs("/sys/class/graphics/fb0/freescale_mode","0");
sw.writeSysfs("/sys/class/graphics/fb0/free_scale","0");
```

然后根据分辨率设置相关节点， 要注意重显率的设置：

```
curPosition = getPosition(old_mode);
sw.writeSysfs("/sys/class/graphics/fb0/freescale_mode","1");
sw.writeSysfs("/sys/class/graphics/fb0/free_scale_axis","0 0 "+curPosition[6]+" "+curPosition[7]);
sw.writeSysfs("/sys/class/graphics/fb0/window_axis", curPosition[0]+" "+curPosition[1]+"
                "+(curPosition[0]+curPosition[2]-1)+" "+(curPosition[1]+curPosition[3]-1));
sw.writeSysfs("/sys/class/display/axis",curPosition[0]+" "+curPosition[1]+" "+(curPosition[6]+1)+"
            "+(curPosition[7]+1)+" "+curPosition[0]+" "+curPosition[1]+" 18 18");
```

最后 enable/disable freescaler：

```
if(curPosition[5] != 0) {
   sw.writeSysfs("/sys/class/graphics/fb0/free_scale","0x10001");
   sw.writeSysfs("/sys/class/graphics/fb1/free_scale","1");
} else {   // 720  or  1080   no need scaler
   sw.writeSysfs("/sys/class/graphics/fb0/free_scale","0");
   sw.writeSysfs("/sys/class/graphics/fb1/free_scale","0");
}
```

## 2.4. 对于不同分辨率，配置 framebuffer, kernel config

### 2.4.1. 简介

MBX 支持 HDMI 和 CVBS 输出，分辨率有 720x480, 720x576, 1280x720, 1920x1080, 3840x2160 等多种。一般是基于一种分辨率为主，其余的分辨率由这种分辨率做 scaler 实现显示。M6 上是以 1280x720 为基准分辨率，M8 由于性能提高，可以支持 1920x1080 分辨率，以 1280x720 和 1920x1080 为基准，480/576 由 720 做 scaler，3840x2160 由 1080 做 scaler。 修改基准分辨率需要 framebuffer, kernel config 等配置。

### 2.4.2. framebuffer 配置

（1） M6 在 common/customer/configs/meson6_g18_jbmr1_defconfig 和 common\customer\boards\board-m6g18.c

```
CONFIG_FB_OSD1_DEFAULT_BITS_PER_PIXEL=32
CONFIG_FB_OSD1_DEFAULT_WIDTH=1280
CONFIG_FB_OSD1_DEFAULT_HEIGHT=1080
CONFIG_FB_OSD1_DEFAULT_WIDTH_VIRTUAL=1280
CONFIG_FB_OSD1_DEFAULT_HEIGHT_VIRTUAL=2160

#define OSD_480_PIX        (640*480)
#define OSD_576_PIX        (768*576)
#define OSD_720_PIX        (1280*720)
#define OSD_1080_PIX       (1920*1080)
#define OSD_PANEL_PIX      (800*480)
#define B16BpP  (2)
#define B32BpP  (4)
#define DOUBLE_BUFFER   (2)

#define OSD1_MAX_MEM       U_ALIGN(OSD_1080_PIX*B32BpP*DOUBLE_BUFFER)
#define OSD2_MAX_MEM       U_ALIGN(32*32*B32BpP)

/******** Reserved memory configuration ***************/
#define OSD1_ADDR_START    U_ALIGN(AUDIODSP_ADDR_END )
#define OSD1_ADDR_END      (OSD1_ADDR_START+OSD1_MAX_MEM - 1)
#define OSD2_ADDR_START    U_ALIGN(OSD1_ADDR_END)
#define OSD2_ADDR_END      (OSD2_ADDR_START +OSD2_MAX_MEM -1)
```

（2） M8 在 common\arch\arm\boot\dts\amlogic\meson8_k200_v1.dtd 中

```
mesonfb{
            compatible = "amlogic,mesonfb";
            dev_name = "mesonfb";
            status = "okay";
            reg = <0x05100000 0x01000000
                        0x06100000 0x00100000>;
                  vmode = <3>; /**0:VMODE_720P 1:VMODE_LCD  2:VMODE_LVDS_1080P
3:VMODE_1080P*/
            display_size_default = <1920 1080 1920 2160 32>;
      };
```

reg： osd0 memory 分配地址， osd0 memory 大小， osd1 memory 分配地址， osd1 memory 大小，
osd memory size = osd width * osd height * double buffer * bytesperpixel (1920*1080*2*4)
display_size_default ： 1920 osd width, 1080 osd height, 2160: double buffer, 1080*2, 32: bitsperpixel

vmode： 分辨率，在 common\drivers\amlogic\display\osd\osd_hw.c 中实现：

```
prop = of_get_property(pdev->dev.of_node, "vmode", NULL);
if(prop)
        prop_idx = of_read_ulong(prop,1);
        if(prop_idx == 3)
                vmode = VMODE_1080P;
        else if(prop_idx == 1)
                vmode = VMODE_LCD;
        else if(prop_idx == 2)
                vmode = VMODE_LVDS_1080P;
        else
                vmode = VMODE_720P;
```

### 2.4.3. device\amlogic

logo 图片资源要改成相应分辨率的
device\amlogic\k200\res_pack\bootup
device\amlogic\k200\initlogo-robot-1920x1080.rle
device\amlogic\k200\bootanimation.zip

### 2.4.4. frameworks\base

对于 M8， 不同的分辨率，android display size 和 density 要相应设置, 而 M6 固定 720 就不需要了
在 frameworks\base\services\java\com\android\server\wm\WindowManagerService.java 里
readForcedDisplaySizeAndDensityLocked()函数中根据分辨率设置，720 及以下的分辨率 size=1280x720，
density=160，1080 及以上的分辨率 size=1920x1080, density=240，注意 size 可能要根据重显率做相应调整

### 2.4.5. apk

apk 也要设置好相应分辨率的资源和布局，这个按 android 标准就可以了

# 3. Setting 部分

## 3.1. 多分辨率切换及重显示率

### 3.1.1. 分辨率设置

仅针对单输出情况，即 HDMI 和 CVBS 只有一种生效

#### 3.1.1.1. 分辨率设置说明

- Amlogic 支持的分辨率模式有
  1）HDMI 模式：
  "480i","480p","576i","576p","720p50hz","720p","1080i","1080p","1080i50hz","1080p50hz","4k2k24hz"
  （对于 p 制式，带有 50hz 的即对应频率为 50hz，不带的即对应频率为 60hz）
  2）CVBS 模式:
  “480cvbs”,”576cvbs”

- 分辨率设置中用到的 sysfs 节点文件与 property 的说明
  1）用到的节点有：
  /sys/class/amhdmitx/amhdmitx0/disp_cap (读取电视所支持的分辨率列表，ps:电视有作假情况)
  /sys/class/aml_mod/mod_off （关 cvbs）
  /sys/class/aml_mod/mod_on （开 cvbs）
  /sys/class/display/mode （切换分辨率）
  /sys/class/display/axis
  /sys/class/graphics/fb0/blank （开关 osd）
  /sys/class/graphics/fb0/free_scale （开关 freescale）
  /sys/class/graphics/fb0/request2XScale
  /sys/class/graphics/fb0/scale_axis
  /sys/class/graphics/fb0/update_freescale （更新 freescale，新接口）
  /sys/class/graphics/fb1/free_scale
  /sys/class/graphics/fb1/scale
  /sys/class/graphics/fb1/scale_axis
  /sys/class/ppmgr/ppscaler
  /sys/class/ppmgr/ppscaler_rect
  /sys/class/video/axis （设置视频的位置）
  /sys/class/display2/mode

  2）用到的 property 有：
  ubootenv.var.480ioutputx
  ubootenv.var.480ioutputy
  ubootenv.var.480ioutputwidth
  ubootenv.var.480ioutputheight
  ubootenv.var.480poutputx
  ubootenv.var.480poutputy
  ubootenv.var.480poutputwidth
  ubootenv.var.480poutputheight
  ubootenv.var.576ioutputx
  ubootenv.var.576ioutputy
  ubootenv.var.576ioutputwidth
  ubootenv.var.576ioutputheight
  ubootenv.var.576poutputx

ubootenv.var.576poutputy
ubootenv.var.576poutputwidth
ubootenv.var.576poutputheight
ubootenv.var.720poutputx
ubootenv.var.720poutputy
ubootenv.var.720poutputwidth
ubootenv.var.720poutputheight
ubootenv.var.1080ioutputx
ubootenv.var.1080ioutputy
ubootenv.var.1080ioutputwidth
ubootenv.var.1080ioutputheight
ubootenv.var.1080poutputx
ubootenv.var.1080poutputy
ubootenv.var.1080poutputwidth
ubootenv.var.1080poutputheight
ubootenv.var.4k2koutputx
ubootenv.var.4k2koutputy
ubootenv.var.s4k2koutputwidth
ubootenv.var.4k2koutputheight
ubootenv.var.cvbsmode （保存 cvbs 分辨率值）
ubootenv.var.hdmimode （保存 hdmi 分辨率值）
ubootenv.var.outputmode（保存分辨率值）
ro.platform.hdmionly （是否为单输出）
ro.platform.has.realoutputmode （是否支持 1080 源）

#### 3.1.1.2. M6， g18 平台上的分辨率设置 , ro.platform.has.realoutputmode 不等于 true 的情况

- 变量定义：

1）String mNewMode ：要切换到的目标分辨率，如(“1080i”)
2） Int x = getPropertyInt(sel_XXXXoutput_x)
Int y = getPropertyInt(sel_XXXXoutput_y)
Int width = getPropertyInt(sel_XXXXoutputwidth)
Int height = getPropertyInt(sel_XXXXoutputheight)
（取对应分辨率的 property 值，不考虑 50hz 还是 60hz）
3）String mValuePpscalerRect :
String mValuePpscalerRect = x + " " + y+ " " + (width + x - 1)+ " " + ( height + y - 1)+ " " + 0;
4）String mValueVideoAxis :
String mValueVideoAxis = x+” ”+y + “ ” + (width + x+-1) + “ ” + ( height + y-1);
5）String mDisplayAxis_1080 :
String mDisplayAxis_1080 = (x/2)*2+” ” + (y/2)*2 + “ ” + “1280 720”+ (x/2)*2+” ”+ (y/2)*2 ;
6）String mFb0ScaleAxis_1080:
String mFb0ScaleAxis_1080 = "0 0 " + (960 - (x/2) - 1) + " " + (1080 - (y/2) - 1);
7）String mFb0Request2XScale_1080 :
String mFb0Request2XScale_1080 = "7" + (width /2) + " " + (height /2)*2;
8）String mFb1ScaleAxis_1080 :
String mFb1ScaleAxis_1080 = "1280 720 " + (width /2)*2 + " " + (height /2)*2);
9）String mDisplayAxis :
String mDisplayAxis = x+ " " + y + " 1280 720 "+ x + " "+ y + " 18 18";
10) String mFb0Request2XScale:
String mFb0Request2XScale = "16 " + width + " " + height ;
11）String mFb1ScaleAxis :
String mFb1ScaleAxis = "1280 720 " + width + " " + height ;

12）String mFb0WindowAxis :

```
String mFb0WindowAxis =  x + " " + y+ " " +  (width  + x -1)  + " " + (height  + y -1) ;
```

- 切换到 mNewMode

1）在 freescale 打开的情况下( 即 /sys/class/graphics/fb0/free_scale 为 "0x1")

```
If ( mNewMode .contains("cvbs"))
        echo vdac > /sys/class/aml_mod/mod_on
else
        echo vdac > /sys/class/aml_mod/mod_off
echo  mNewMode >  /sys/class/display/mode
echo  mValuePpscalerRect  > /sys/class/ppmgr/ppscaler_rect
echo 1 > /sys/class/graphics/fb0/update_freescale
setprop  ubootenv.var.outputmode mNewMode
If( mNewMode .contains("cvbs" ))
        setprop   ubootenv.var.cvbsmode mNewMode
else
        setprop  ubootenv.var.hdmimode mNewMode
```

2）在 freescale 关闭的情况下（即 /sys/class/graphics/fb0/free_scale 为 "0x0"）

```
If ( mNewMode .contains("cvbs"))
        echo vdac > /sys/class/aml_mod/mod_on
else
        echo vdac > /sys/class/aml_mod/mod_off
echo  1 >  /sys/class/graphics/fb0/blank
echo 0 >  /sys/class/ppmgr/ppscaler
echo 0 >  /sys/class/graphics/fb0/free_scale
echo 0 >  /sys/class/graphics/fb1/free_scale
echo  mNewMode >  /sys/class/display/mode
echo mValueVideoAxis >  /sys/class/video/axis
setprop  ubootenv.var.outputmode mNewMode
setprop  ubootenv.var.hdmimode mNewMode
if  （ mNewMode 为 1080 制式）
        echo  mDisplayAxis_1080 > /sys/class/display/axis
        echo mFb0Request2XScale_1080 > /sys/class/graphics/fb0/request2XScale
        echo mFb1ScaleAxis_1080 > /sys/class/graphics/fb1/scale_axis
        echo 0x10001 > /sys/class/graphics/fb1/scale
else      （NEW_MODE 为非 1080 制式）
        echo mDisplayAxis  > /sys/class/display/axis
        echo mFb0Request2XScale  > /sys/class/graphics/fb0/request2XScale
        echo  mFb1ScaleAxis  > /sys/class/graphics/fb1/scale_axis
        echo 0x10001 > /sys/class/graphics/fb1/scale
setprop  ubootenv.var.outputmode mNewMode
If( mNewMode .contains("cvbs" ))
        setprop   ubootenv.var.cvbsmode mNewMode
else
        setprop  ubootenv.var.hdmimode mNewMode
```

#### 3.1.1.3. M8， k200 平台上的分辨率设置，ro.platform.has.realoutputmode 等于 ture 的情况

1）480 分辨率 （包括 480p,480i,480cvbs）

```
If ( mNewMode .contains("cvbs"))
        echo vdac > /sys/class/aml_mod/mod_on
else
echo vdac > /sys/class/aml_mod/mod_off
echo mNewMode  >  /sys/class/display/mode
```

```
echo 0 >  /sys/class/graphics/fb0/free_scale
setDisplaySize(1280,720);
setDensity("160");
echo 1 >   /sys/class/graphics/fb0/freescale_mode
echo 0 0 1279 719 > /sys/class/graphics/fb0/free_scale_axis
echo  mFb0WindowAxis  >  /sys/class/graphics/fb0/window_axis
echo  0x10001 > /sys/class/graphics/fb0/free_scale
echo  mValueVideoAxis   >  /sys/class/video/axis
String  mDisplayAxis =  x + " " + y +  " 720  480 " + x + " " + y +  18 + " " + 18;
echo  mDisplayAxis  > /sys/class/display/axis ;
setprop  ubootenv.var.outputmode mNewMode
If( mNewMode .equals("480cvbs" ))
        setprop   ubootenv.var.cvbsmode mNewMode
else
        setprop  ubootenv.var.hdmimode mNewMode
```

2）576 分辨率 （包括 576p,576 i,576 cvbs）

```
If ( mNewMode .contains("cvbs"))
        echo vdac > /sys/class/aml_mod/mod_on
else
echo vdac > /sys/class/aml_mod/mod_off
echo mNewMode  >  /sys/class/display/mode
echo 0 >  /sys/class/graphics/fb0/free_scale
setDisplaySize(1280,720);
setDensity("160");
echo 1 >   /sys/class/graphics/fb0/freescale_mode
echo 0 0 1279 719 > /sys/class/graphics/fb0/free_scale_axis
echo  mFb0WindowAxis  >  /sys/class/graphics/fb0/window_axis
echo  0x10001 > /sys/class/graphics/fb0/free_scale
echo  mValueVideoAxis   >  /sys/class/video/axis
String  mDisplayAxis =  x + " " + y +  " 720  576 " + x + " " + y +  18 + " " + 18;
echo  mDisplayAxis  > /sys/class/display/axis ;
setprop  ubootenv.var.outputmode mNewMode
If( mNewMode .equals("576cvbs" ))
        setprop   ubootenv.var.cvbsmode mNewMode
else
        setprop  ubootenv.var.hdmimode mNewMode
```

3）720 分辨率 （包括 720 p,720p50hz,720 i）

```
If ( mNewMode .contains("cvbs"))
        echo vdac > /sys/class/aml_mod/mod_on
else
echo vdac > /sys/class/aml_mod/mod_off
echo mNewMode  >  /sys/class/display/mode
echo 0 >  /sys/class/graphics/fb0/free_scale
setDisplaySize(width ,height);
setDensity("160");
echo 1 >   /sys/class/graphics/fb0/freescale_mode
echo 0 0 1280 720 > /sys/class/graphics/fb0/free_scale_axis
echo  mFb0WindowAxis  >  /sys/class/graphics/fb0/window_axis
echo  0 > /sys/class/graphics/fb0/free_scale
echo  mValueVideoAxis   >  /sys/class/video/axis
String  mDisplayAxis =  x + " " + y +  " 1280  720 " + x + " " + y +  18 + " " + 18;
echo  mDisplayAxis  > /sys/class/display/axis ;
```

```
setprop  ubootenv.var.outputmode mNewMode
setprop  ubootenv.var.hdmimode mNewMode
```

4）1080 分辨率 （包括 1080p,1080p50hz,1080 i）

```
If ( mNewMode .contains(“cvbs”))
        echo vdac > /sys/class/aml_mod/mod_on
else
echo vdac > /sys/class/aml_mod/mod_off
echo mNewMode  >  /sys/class/display/mode
echo 0 >  /sys/class/graphics/fb0/free_scale
setDisplaySize(width ,height);
setDensity(“240”);
echo 1 >   /sys/class/graphics/fb0/freescale_mode
echo 0 0 1920 1080 > /sys/class/graphics/fb0/free_scale_axis
echo  mFb0WindowAxis  >  /sys/class/graphics/fb0/window_axis
echo  0 > /sys/class/graphics/fb0/free_scale
echo  mValueVideoAxis   >  /sys/class/video/axis
String  mDisplayAxis =  x + “ ” + y +  " 1920 1080 " + x + “ ” + y +  18 + “ ” + 18;
echo  mDisplayAxis  > /sys/class/display/axis ;
setprop  ubootenv.var.outputmode mNewMode
setprop  ubootenv.var.hdmimode mNewMode
```

5）4k2k 分辨率 （包括 4k2k24hz）

```
If ( mNewMode .contains(“cvbs”))
        echo vdac > /sys/class/aml_mod/mod_on
else
echo vdac > /sys/class/aml_mod/mod_off
echo mNewMode  >  /sys/class/display/mode
echo 0 >  /sys/class/graphics/fb0/free_scale
setDisplaySize(1920,1080);
setDensity(“240”);
echo 1 >   /sys/class/graphics/fb0/freescale_mode
echo 0 0 1920 1080 > /sys/class/graphics/fb0/free_scale_axis
echo  mFb0WindowAxis  >  /sys/class/graphics/fb0/window_axis
echo  0x10001 > /sys/class/graphics/fb0/free_scale
echo  mValueVideoAxis   >  /sys/class/video/axis
String  mDisplayAxis =  x + “ ” + y +  " 3840 2160 " + x + “ ” + y +  18 + “ ” + 18;
echo  mDisplayAxis  > /sys/class/display/axis ;
setprop  ubootenv.var.outputmode mNewMode
setprop  ubootenv.var.hdmimode mNewMode
```



#### 3.1.1.4. 注意事项

1）在分辨率切换过程中，界面会有伸缩之类的动作，为了不让用户看到这个过程，可以通过节点关掉 osd。

2）在 M8 平台上，大于等于 1080 分辨率 与 小于等于 720 分辨率 之间切换，同时会切换 density，由此会引起前台应用重新运行。

3）在 M6 平台上，通过检测 HDMI 是否插上的互斥条件，来判定当前是否为 CVBS 模式。而在 M8 上由于 2）所描述的现象存在。故 M8 现在不支持，动态的 HDMI 与 CVBS 模式切换。

### 3.1.2. 重显率设置

#### 3.1.2.1. M6， g18 平台上的分辨率设置，ro.platform.has.realoutputmode 不等于 true 的情况

- 变量定义：

```
1)   int  x  = getPropertyInt(sel_XXXXoutput_x)
     int  y =  getPropertyInt(sel_XXXXoutput_y)
int width = getPropertyInt(sel_XXXXoutputwidth)
     int  height = getPropertyInt(sel_XXXXoutputheight)
          2)  int left : 左上角横轴坐标
     int left = x;
          3)int top : 左上角纵轴坐标
     int top= y;
          4)int right : 右下角横轴坐标
     int right = left  + width  +1
 5)int bottom: 右下角纵轴坐标
     int bottom = top + bottom + 1
 6)int newLeft : 调整后的 left 值
 7)int newTop : 调整后的 top 值
          8)int newRight : 调整后的 right 值
          9)int newBottom: 调整后的 bottom 值
         10)int newWidth:
         int newWidth = newRight  - newLeft  -1;
         11)int newHeight :
          int newHeight =    newBottom - newTop -1 ;
```

- 设置重现率

```
（int mPpscalerRect = newLeft+ " " + newTop+ " " + newRight+ " " + newBottom + " " + 0）
echo mPpscalerRect>  /sys/class/ppmgr/ppscaler_rect
```

- 保存重现率值

```
serprop sel_XXXXoutput_x  newLeft
serprop sel_XXXXoutput_y  newTop
setprop sel_XXXXoutputwidth  newWidth
        setprop sel_XXXXoutputheight  newHeight
```

#### 3.1.2.2. M8， k200 平台上的分辨率设置，ro.platform.has.realoutputmode 等于 true 的情况

- 变量定义：

```
 a) String mCurrentMode : 当前分辨率
 b) String  mValueDisplay  :
If(mCurrentMode .contains("4k2k"))
        mValueDisplay = “ 3840 2160 ”
else if(mCurrentMode .contains("1080") )
        mValueDisplay = “ 1920  1080 ”
         else if(mCurrentMode .contains("720"))
            mValueDisplay = “ 1280 720 ”
         else if(mCurrentMode .contains("576") )
                  mValueDisplay = “ 720  576 ”
else  if(mCurrentMode .contains("480"))
        mValueDisplay = “ 720  480 ”
```

- 设置重现率

```
if(mCurrentMode.contains("720")||mCurrentMode.contains("1080")){
        echo 1 > /sys/class/graphics/fb0/freescale_mode
    (String mFreeScaleAxis = “0 0” +width + " "+height)
    echo mFreeScaleAxis  > /sys/class/graphics/fb0/free_scale_axis
 }
(String str = newLeft + " " + newTop + " " + newRight + " " + newBottom )
echo  str > /sys/class/graphics/fb0/window_axis
echo 0x10001 > /sys/class/graphics/fb0/free_scale
```

- 保存重现率值

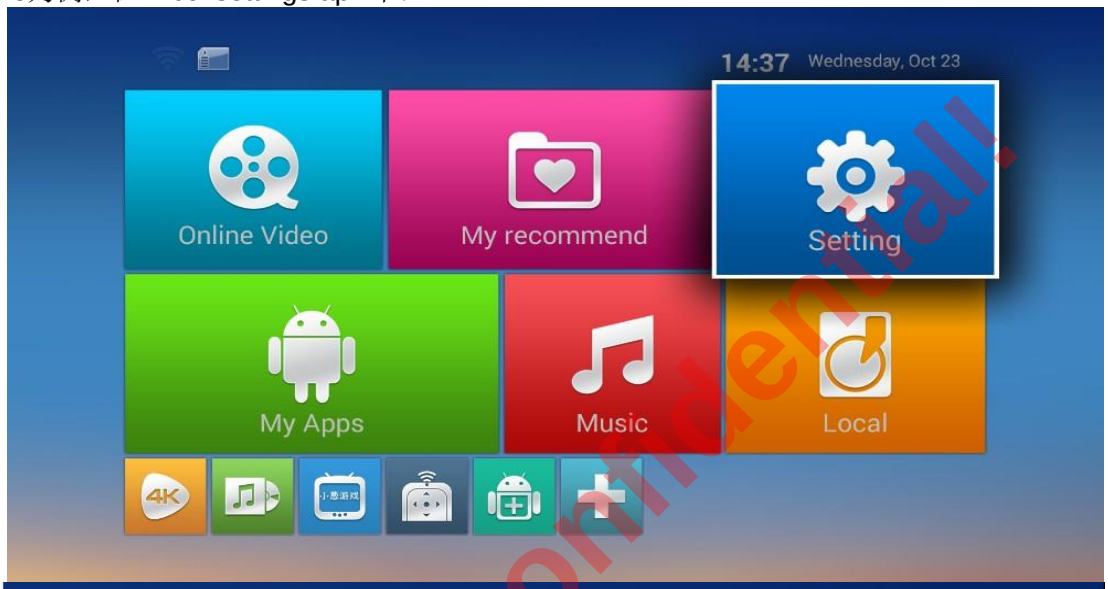
```
if( mCurrentMode .contains("720")||mCurrentMode.contains("1080")){
echo 0x0 > /sys/class/graphics/fb0/free_scale
 setDisplaySize(newWidth,newHeight );
}
(String mDisplayAxis= newLeft +” ” + newTop  + “ ” + mValueDisplay  +“ ” + newLeft +” ”+newTop  + “ ”+
18 + “ ” + 18 )
echo  mDisplayAxis >  /sys/class/display/axis

(String mVideoAxis  = newLeft +" "+newTop  +" "+newWidth+" "+newHeight );
echo  mVideoAxis  > /sys/class/video/axis
```

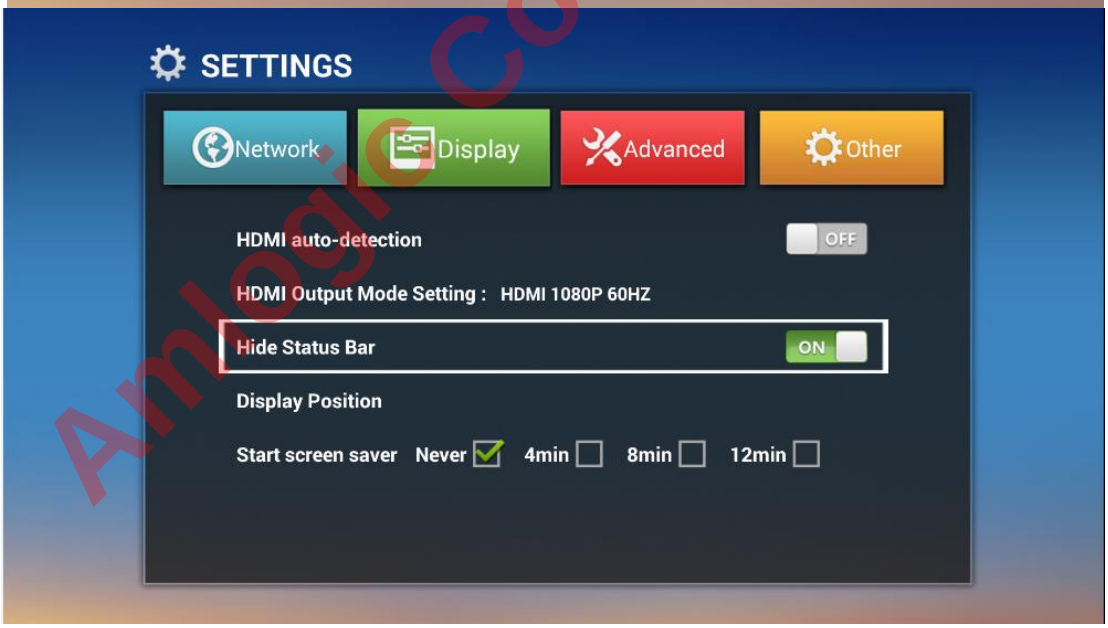
## 3.2. 隐藏状态栏

对于 mbox 产品来说，状态栏可能是不需要的。所以 amlogic 提供给用户一个操作接口，用户可以自己选择是否隐藏掉状态栏。

### 3.2.1. 用户操作界面

以 android4.3为例，在 mboxsettings.apk 中：




### 3.2.2. 代码实现

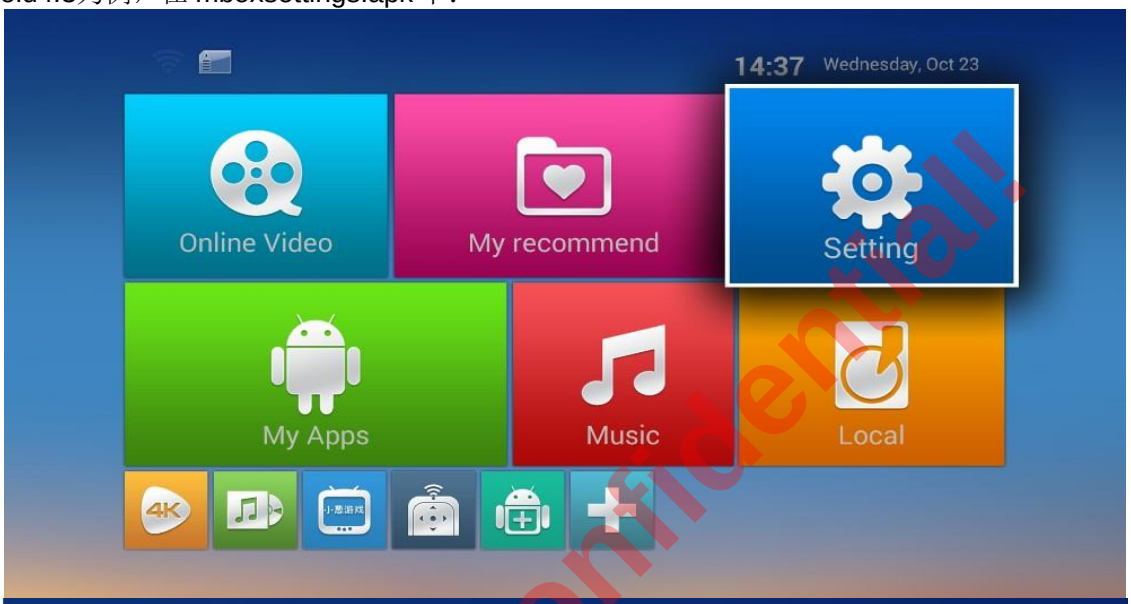
在 mboxsetting.apk 的源码中，对是否显示 statusbar 是通过写一个 property 来控制的，这个 property 是：persist.sys.hideStatusBar 。如果是 true，则隐藏，如果是 false，则显示。至于这个 property 如何起作用，大家则可以参看 framework/base 中，与”persist.sys.hideStatusBar”相关的代码。

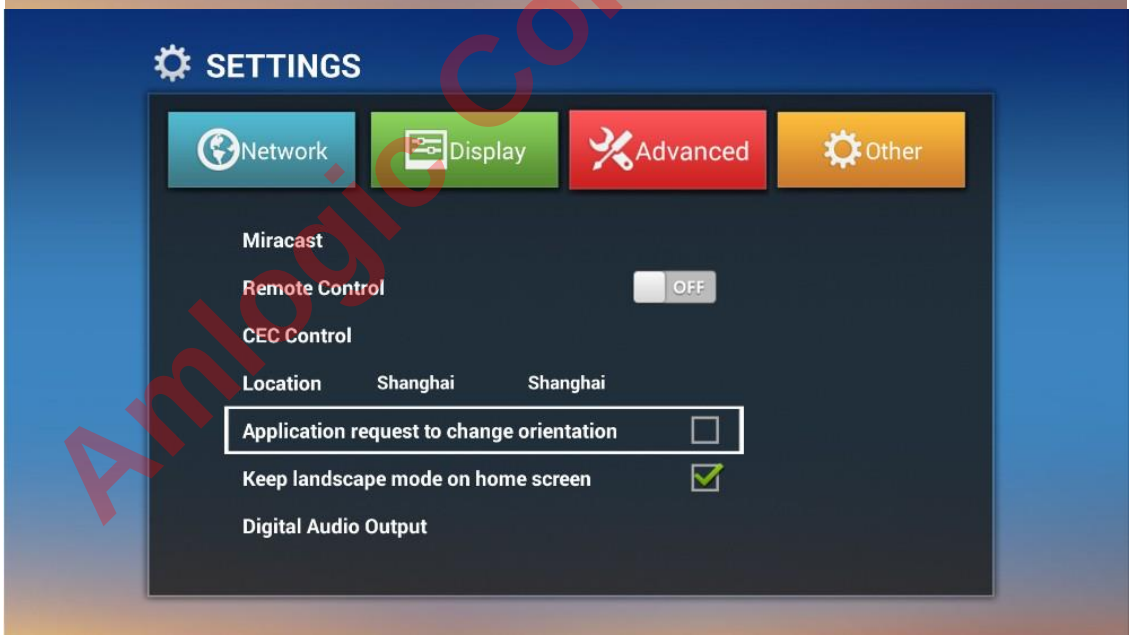
## 3.3. 横屏设置

对 mbx 这个产品定位来说，一般是固定放置，没有 sensor，屏幕不会旋转。所以在 android4.3中是默认不旋转屏幕；但考虑到兼容性，还是允许应用程序请求旋转屏幕的。

### 3.3.1. 应用程序中设置

以 android4.3为例，在 mboxsettings.apk 中：




### 3.3.2. 代码实现

在 mboxsetting 中，是否勾选这个选项，实际上是设置了一个 property：ubootenv.var.has.accelerometer 。

如果为 true，则应用程序可以旋转屏幕；如果为 false，则不可以。至于这个 property 如果控制屏幕能否旋转，可以参看 framework/base 中跟此 property 相关的代码。

## 3.4. 以太或者 WIFI 优先的设置

网络优先的定义，在 android 中有默认配置的地方。

以 android4.2 为例， frameworks\base\packages\settingsprovider\res\values\defaults.xml 中，有一个 property：def_network_preference 。

这个 property 用来控制网络通路的优先级。

默认是1，即 wifi 优先；在 mbox 的相关 build 中，overlay 中将其定义为9，即 ethernet 优先。

大家感兴趣的话，可以参看 DatabaseHelper.java 。在这个里面，会读取 def_network_preference 的值，并把它存入 database 中。在网络连接时，就会从 database 中读取这个值，进行优先级设置。

该文件的 path：

frameworks\base\packages\settingsprovider\src\com\android\providers\settings\DatabaseHelper.java

## 3.5. 声音节点介绍

- getprop media.amplayer.enable-acodecs
  查看音频格式 audiodsp 解码支持

- /sys/class/audiodsp/digital_raw
  0：PCM
  1：SPDIF passthrough
  2：HDMI passthrough
- /sys/class/audiodsp/ac3_drc_control
  1）DRC Mode： {"CUSTOM_0","CUSTOM_1","LINE","RF"}
  Echo drcmode 2 > /sys/class/audiodsp/ac3_drc_control
  2）DRC Enable/Disable
  DRC high cut scale : 0~100 （默认 100） (0 为关闭，100 为打开后默认值）
  DRC low boost scale:0~100（默认 100） (0 为关闭，100 为打开后默认值）
  关闭 DRC ：
  Echo drchighcutscale 0> /sys/class/audiodsp/ac3_drc_control
  Echo drclowboostscale 0> /sys/class/audiodsp/ac3_drc_control
  打开 DRC ：
  Echo drchighcutscale 64 > /sys/class/audiodsp/ac3_drc_control (十六进制）
  Echo drclowboostscale 64 > /sys/class/audiodsp/ac3_drc_control （十六进制）

- /sys/class/audiodsp/dec_option
  "0 - mute dts and ac3 ",
  "1 - mute dts.ac3 with noise ",
  "2 - mute ac3.dts with noise",
  "3 - both ac3 and dts with noise",

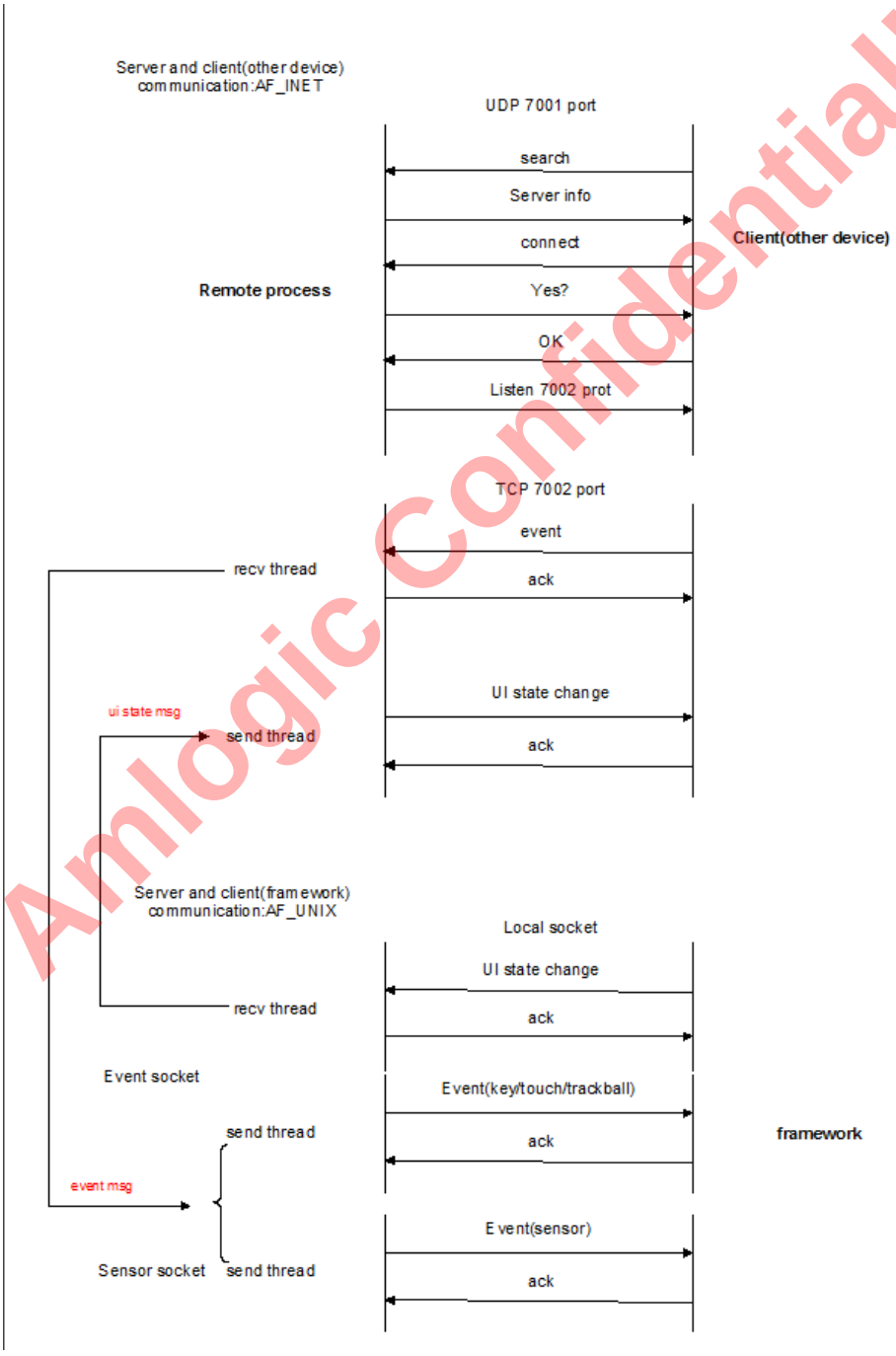
- /sys/class/audiodsp/dts_dec_control
  1） downmix mode： 0：Lo/Ro ； 1：Lt/Rt
  echo dtsdmxmode 0 > /sys/class/audiodsp/dts_dec_control
  echo dtsdmxmode 1 > /sys/class/audiodsp/dts_dec_control
  2) DRC scale control
  范围 0~100
  默认 0 就是关闭 DRC
  echo dtsdrcscale 0~0x64 > /sys/class/audiodsp/dts_dec_control
  3) Dial Norm control
  enable

disable
默认 enable
分别对应
echo dtsdialnorm 1 > /sys/class/audiodsp/dts_dec_control
echo dtsdialnorm 0 > /sys/class/audiodsp/dts_dec_control

## 3.6. Remote Control

### 3.6.1. Remote Controller 简介

Remote Controller 是局域网下客户端控制服务器的一种交互方式。其主要的交互协议如下：



### 3.6.2. Amlogic Remote Controller

Amlogic 从 Android2.2 开始支持 RemoteControl，主要版本有两个。4.2 之前的 Remote Controller Server 端是 j 集成在系统中的，4.2 以及之后的版本都是由单独的 apk(RC_Service.apk)进行控制的。主要客户端/服务器端主要交互的事件有：

- KEY EVENT：单独的 key 事件，包含有 home, search, left, right, center, up , down, volume down, volume up 等按键。
- TOUCH EVENT：客户端发送一系列 touch 事件给服务器端 apk，服务器端响应 touch 事件。
- MOUSE EVENT：客户端可以通过 touch 事件模拟鼠标事件，需要实现的时 touch 与 mouse 模式之间的切换。
- SENSOR EVENT：客户端将自身的 sensor 数据发送给服务器端，服务器端进行响应。注意此部分的响应是需要系统(framework)的支持。

Remote Controller 远程控制，主要是针对 mbox 而实现的。客户端 demo apk 可以安装在 Android1.6 版本以及以上版本上进行控制。Amlogic 支持对 Remote Controller 的客户端的二次开发。

### 3.6.3. Amlogic 服务器相关配置

前面提到对于 Android 4.2 之前的版本，Remote Controller 都是集成在系统 framework 中的。对于 Android 4.2 之后的系统都是单独的 apk，预安装或安装方式都可以使其正常运行。无论是哪个版本的 Remote Controller ，想要系统响应客户端发送过来的 Sensor 数据，需要在 system.prop 中进行如下配置：

hw.has.accelerometer = false

来开启系统对 accelerometer 的响应。

#### 3.6.3.1. key 值的定义

对于客户端/服务器 key 值的交互，主要是：

```
//define protocol keycode
#define RC_KEYCODE_HOME                 0
#define RC_KEYCODE_MENU                 1
#define RC_KEYCODE_BACK                 2
#define RC_KEYCODE_VOLUME_UP            3
#define RC_KEYCODE_VOLUME_DOWN          4
#define RC_KEYCODE_SEARCH               5
#define RC_KEYCODE_MUTE                 6
#define RC_KEYCODE_UP                   7
#define RC_KEYCODE_DOWN                 8
#define RC_KEYCODE_LEFT                 9
#define RC_KEYCODE_RIGHT                10
#define RC_KEYCODE_CENTER               11
#define RC_KEYCODE_POWER                12
#define RC_KEYCODE_CAMERA               13
#define RC_KEYCODE_CLEAR                14
#define RC_KEYCODE_SOFT_LEFT            15
#define RC_KEYCODE_SOFT_RIGHT           16
#define RC_KEYCODE_CALL                 17
#define RC_KEYCODE_ENDCALL              18
#define RC_KEYCODE_0                    19
#define RC_KEYCODE_1                    20
#define RC_KEYCODE_2                    21
#define RC_KEYCODE_3                    22
#define RC_KEYCODE_4                    23
#define RC_KEYCODE_5                    24
#define RC_KEYCODE_6                    25
```

```
#define RC_KEYCODE_7                    26
#define RC_KEYCODE_8                    27
#define RC_KEYCODE_9                    28
#define RC_KEYCODE_STAR                 29
#define RC_KEYCODE_POUND                30
#define RC_KEYCODE_A                    31
#define RC_KEYCODE_B            32
#define RC_KEYCODE_C            33
#define RC_KEYCODE_D            34
#define RC_KEYCODE_E            35
#define RC_KEYCODE_F            36
#define RC_KEYCODE_G            37
#define RC_KEYCODE_H            38
#define RC_KEYCODE_I              39
#define RC_KEYCODE_J              40
#define RC_KEYCODE_K            41
#define RC_KEYCODE_L            42
#define RC_KEYCODE_M            43
#define RC_KEYCODE_N            44
#define RC_KEYCODE_O            45
#define RC_KEYCODE_P            46
#define RC_KEYCODE_Q            47
#define RC_KEYCODE_R            48
#define RC_KEYCODE_S            49
#define RC_KEYCODE_T            50
#define RC_KEYCODE_U            51
#define RC_KEYCODE_V            52
#define RC_KEYCODE_W            53
#define RC_KEYCODE_X            54
#define RC_KEYCODE_Y            55
#define RC_KEYCODE_Z                56
#define RC_KEYCODE_COMMA            57
#define RC_KEYCODE_PERIOD           58
#define RC_KEYCODE_ALT_LEFT         59
#define RC_KEYCODE_ALT_RIGHT        60
#define RC_KEYCODE_SHIFT_LEFT         61
#define RC_KEYCODE_SHIFT_RIGHT        62
#define RC_KEYCODE_TAB              63
#define RC_KEYCODE_SPACE            64
#define RC_KEYCODE_SYM              65
#define RC_KEYCODE_EXPLORER         66
#define RC_KEYCODE_ENVELOPE         67
#define RC_KEYCODE_ENTER            68
#define RC_KEYCODE_DEL              69
#define RC_KEYCODE_GRAVE            70
#define RC_KEYCODE_MINUS            71
#define RC_KEYCODE_EQUALS           72
#define RC_KEYCODE_LEFT_BRACKET         73
#define RC_KEYCODE_RIGHT_BRACKET        74
#define RC_KEYCODE_BACKSLASH        75
#define RC_KEYCODE_SEMICOLON        76
#define RC_KEYCODE_APOSTROPHE       77
#define RC_KEYCODE_SLASH            78
#define RC_KEYCODE_AT           79
```

```
#define RC_KEYCODE_NUM                  80
#define RC_KEYCODE_HEADSETHOOK          81
#define RC_KEYCODE_FOCUS                82// *Camera* focus
#define RC_KEYCODE_PLUS                 83
#define RC_KEYCODE_NOTIFICATION         84
#define RC_KEYCODE_MEDIA_PLAY_PAUSE             85
#define RC_KEYCODE_MEDIA_STOP           86
#define RC_KEYCODE_MEDIA_NEXT           87
#define RC_KEYCODE_MEDIA_PREVIOUS       88
#define RC_KEYCODE_MEDIA_REWIND         89
#define RC_KEYCODE_MEDIA_FAST_FORWARD   90
#define RC_KEYCODE_UNKNOWN                      91
```

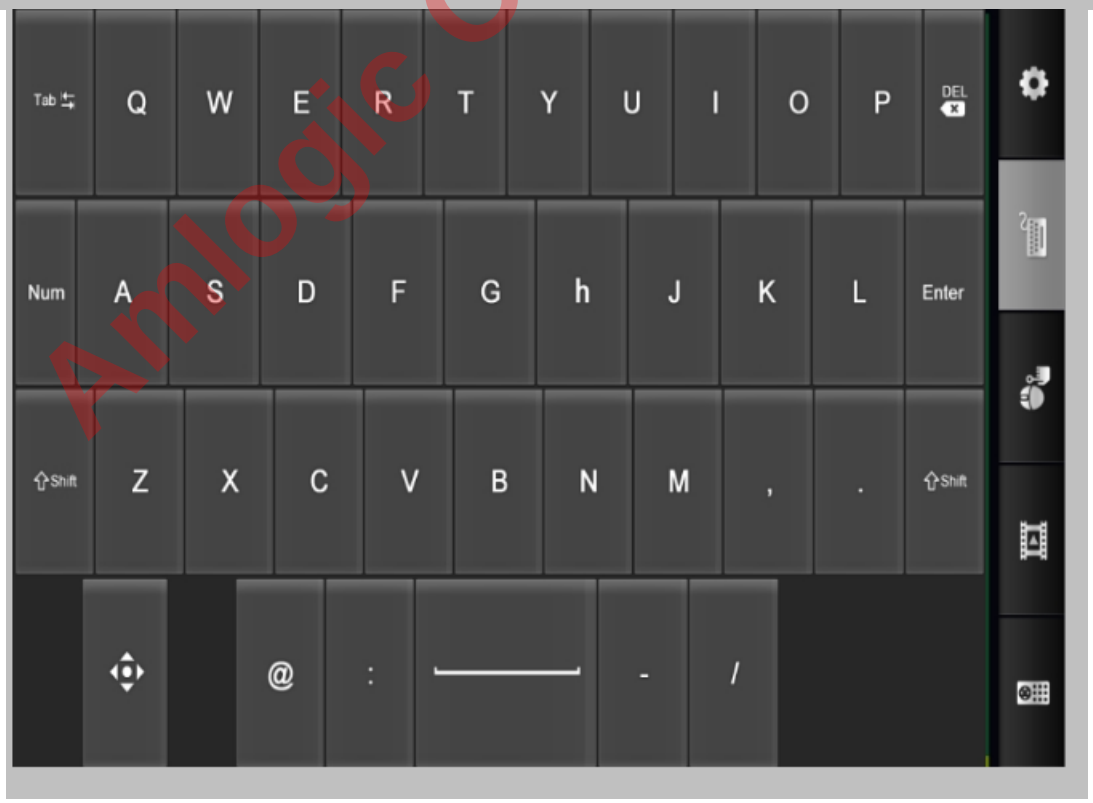
#### 3.6.3.2. Device Filter

Remote Controller 在搜索及连接的时候，会获取 server 的生产厂商(ro.product.manufacturer)，并根据厂商判断是否可以搜索到设备并进行连接。主要的配置是通过客户端 res/values/array.xml 进行配置的。默认情况下 amlogic 发布出来的 apk 只是支持 amlogic 的产品。

```
<string-array translatable="false" name="device_list" >
        <item>"MID"</item>
        <item>"MBX"</item>
 </string-array>
```

### 3.6.4. RC_Client.apk/RC_Server.apk

下面的两张图为客户端/服务器端的 UI。



## 3.7. Miracast

### 3.7.1. Miracast 简介

Wi-Fi Display 经常和 Miracast 联系在一起。实际上，Miracast 是 Wi-Fi 联盟（Wi-Fi Alliance）对支持 Wi-Fi Display 功能的设备的认证名称。通过 Miracast 认证的设备将在最大程度内保持对 Wi-Fi Display 功能的支持和兼容。由此可知，Miracast 考察的就是 Wi-Fi Display（本文后续将不再区分 Miracast 和 Wi-Fi Display）。而 Wi-Fi Display 的核心功能就是让设备之间通过 Wi-Fi 无线网络来分享视音频数据。以一个简单的应用场景为例：有了 Wi-Fi Display 后，手机和电视机之间可以直接借助 Wi-Fi，而无需硬连线（如 HDMI）就可将手机中的视频投递到 TV 上去显示。以目前智能设备的发展趋势来看，Wi-Fi Display 极有可能在较短时间内帮助我们真正实现多屏互动。

### 3.7.2. Amlogic Miracast

Android 4.2 在 Project Butter 基础上再接再厉，新增了对 Wi-Fi Display 功能的支持。主要有如下功能：

- Device Discovery：通过 Wi-Fi P2P 来查找附近的支持 Wi-Fi P2P 的设备。
- Device Selection: 当设备 A 发现设备 B 后，A 设备需要提示用户。用户可根据需要选择是否和设备 B 配对。
- Connection Setup：Source 和 Display 设备之间通过 Wi-Fi P2P 建立连接。根据 Wi-Fi Direct 技术规范，这个步骤包括建立一个 Group Owner 和一个 Client。此后，这两个设备将建立一个 TCP 连接，同时一个用于 RTSP 协议的端口将被创建用于后续的 Session 管理和控制工作。
- Capability Negotiation：在正式传输视音频数据前，Source 和 Display 设备需要交换一些 Miracast 参数信息，例如双方所支持的视音频格式等。二者协商成功后，才能继续后面的流程。
- Session Establishment and streaming：上一步工作完成后，Source 和 Display 设备将建立一个 Miracast Session。而后就可以开始传输视音频数据。Source 端的视音频数据将经由 MPEG2TS 编码后通过 RTP 协议传给 Display 设备。Display 设备将解码收到的数据，并最终显示出来。
- User Input back channel setup：这是一个可选步骤。主要用于在传输过程中处理用户发起的一些控制操作。这些控制数据将通过 TCP 在 Source 和 Display 设备之间传递。
- Payload Control：传输过程中，设备可根据无线信号的强弱，甚至设备的电量状况来动态调整传输数据和格式。可调整的内容包括压缩率，视音频格式，分辨率等内容。
- Session teardown：停止整个 Session。

当前，主要是 MBX 支持 DLNA 功能。

### 3.7.3. Amlogic Miracast apk 的实现部分

Amlogic Miracast apk 主要主要是负责接收 framework 发出来的广播，并启动显示 activity 显现客户端传输过来的数据信息（视频，图片等）。以及发送搜索 p2p 设备的广播，获取 framework 里面搜索到的设备列表等共功能。

#### 3.7.3.1. 相关广播

WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION
表明 Wi-Fi 对等网络（P2P）是否已经启用
WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION
表明可用的对等点的列表发生了改变
WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION
表示 Wi-Fi 对等网络的连接状态发生了改变
WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION
表示该设备的配置信息发生了改变
WifiP2pManager.WIFI_P2P_DISCOVERY_CHANGED_ACTION
表明对等网络设备发现发生了改变
Amlogic 预设广播
WiFiDirectMainActivity.DNSMASQ_IP_ADDR_ACTION = "android.net.dnsmasq.IP_ADDR"

通过此广播获取到连接到本设备的对等设备的 mac 地址，ip 地址以及 port 端口信息
String mac = intent.getStringExtra(WiFiDirectMainActivity.DNSMASQ_MAC_EXTRA);
String ip = intent.getStringExtra(WiFiDirectMainActivity.DNSMASQ_IP_EXTRA);
String port = intent.getStringExtra(WiFiDirectMainActivity.DNSMASQ_PORT_EXTRA);

#### 3.7.3.2. 显示部分

通过接听广播信息获取到连接本设备的对等设备的 ip 以及端口信息之后，启动显示 activity 连接对等设备并显示传过来的信息。

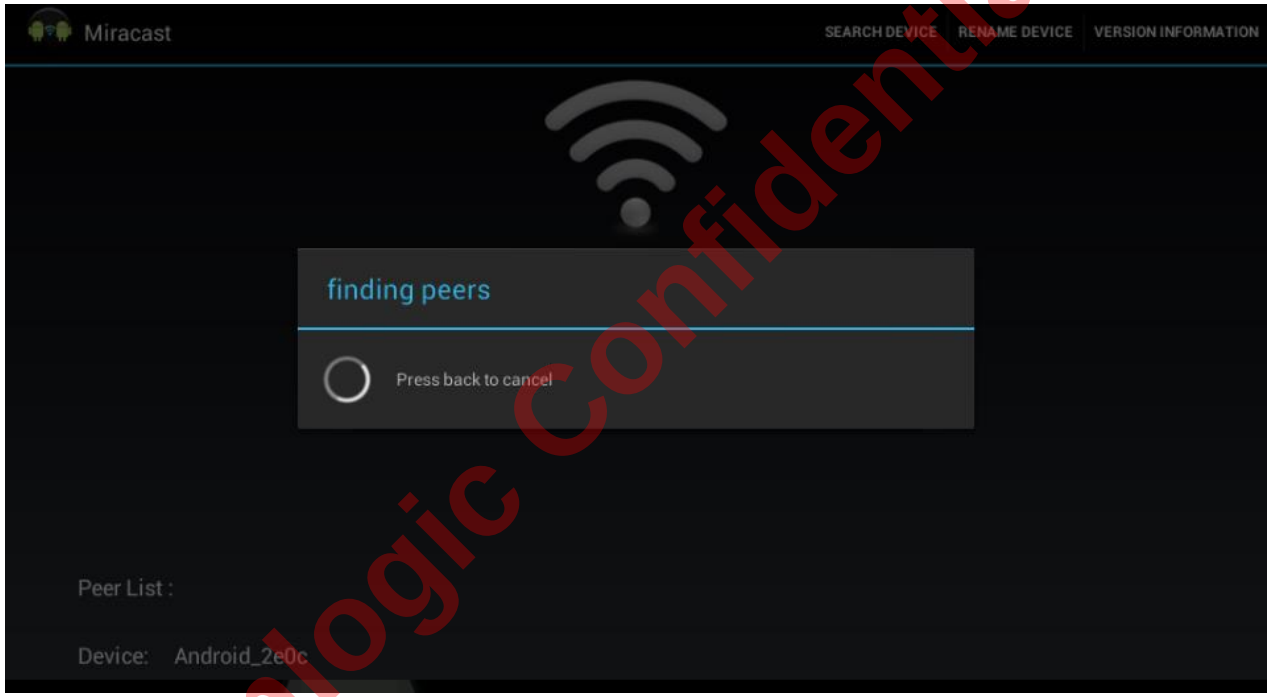
ANetworkSession: 通过此类获得与客户端进行通信的 session 对象。

WifiDisplaySink: 通过此类建立连接。

建立连接之后，通过 SurfaceView 显示传过来的图片，视频等。

### 3.7.4. Miracast APK

下面的两张图为其 UI。




### 3.7.5. Miracast.apk 的二次开发

Amlogic Miracast.apk 的源码是完全公开的，可以根据现有代码进行二次开发。由于 Miracast apk 主要是监听系统中发送出来的广播信息，并进行响应。Amlogic Miracast.apk 的基本上从功能上面不需要再进行二次开发。可以通过修改 UI 来对 Miracast 进行二次开发。

## 3.8. OTA Upgrade 介绍

### 3.8.1. OTA Upgrade 简介

Amlogic 的升级 apk 为 OTA Upgrade，其主要是包含有 OTA 方式的升级。OTA 是 google 提出的 smart app update 升级方式。

**官方说明**

Smart app updates is a new feature of Google Play that introduces a better way of delivering app updates to devices. When developers publish an update, Google Play now delivers only the bits that have changed to devices, rather than the entire APK. This makes the updates much lighter-weight in most cases, so they are faster to download, save the device's battery, and conserve bandwidth usage on users' mobile data plan. On average, a smart app update is about 1/3 the sizeof a full APK update.

### 3.8.2. Amlogic OTA

Amlogic OTA 的服务器与客户端的交互遵守一下协议：

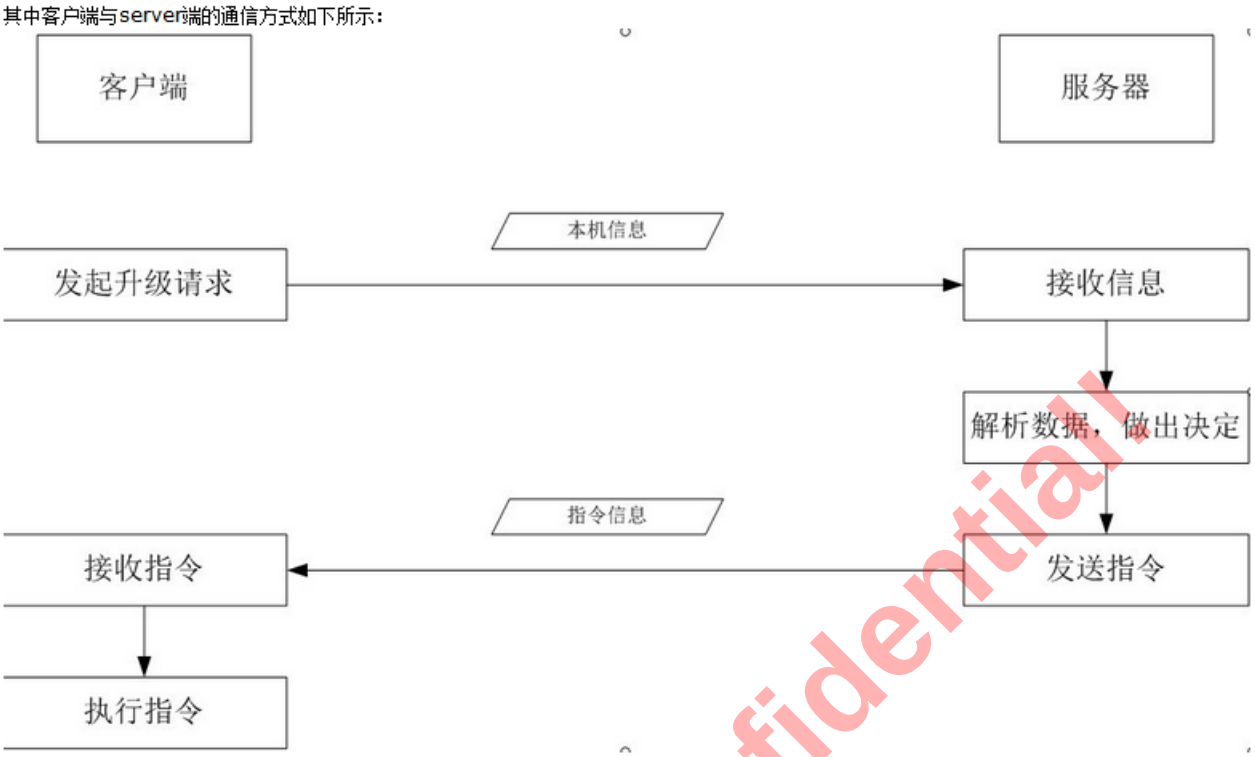



图1

OTA 服务器的搭建可以参考 Amlogic OTA Server 的详细文档。

### 3.8.3. 系统相关配置

当你进行增量包在线升级的时候，服务器需要通过 OTAUpgrade.apk 获取你的系统中的信息。主要是版本信息和服务器地址。OTAUpgrade.apk 在连接时通过系统配置好的服务器的地址连接服务器，并告知服务器自身的系统信息。服务器获取到客户端的信息之后，判定是否有新的版本。若是有新的版本则将新版本的地址发送给客户端，客户端下载新版本的内容并进行升级。

#### 3.8.3.1. 版本信息

firmware 要求数值序列按如下方式编辑 00411003，其中前两位作为预留，3-5 标识系统版本号 Android4.1.1，最后三位标识升级序列。也即是说，对于第一批产品而言，系统的初始版本为**411001，第一次升级包版本为**411002。

debug.conf/update.conf 中对于通配的支持：
整个 android 系统大版本更新 4.1.1-4.2.0，*为通配。
0041104*=http://192.168.0.124:8080/update/xml...ota_test_1.xml
004113**=http://192.168.0.124:8080/update/xml...ota_test_1.xml
.....
00420001=http://192.168.0.124:8080/update/xml...ota_test_3.xml
00420002=http://192.168.0.124:8080/update/xml...ota_test_4.xml
lastest=00411003

#### 3.8.3.2. 配置版本

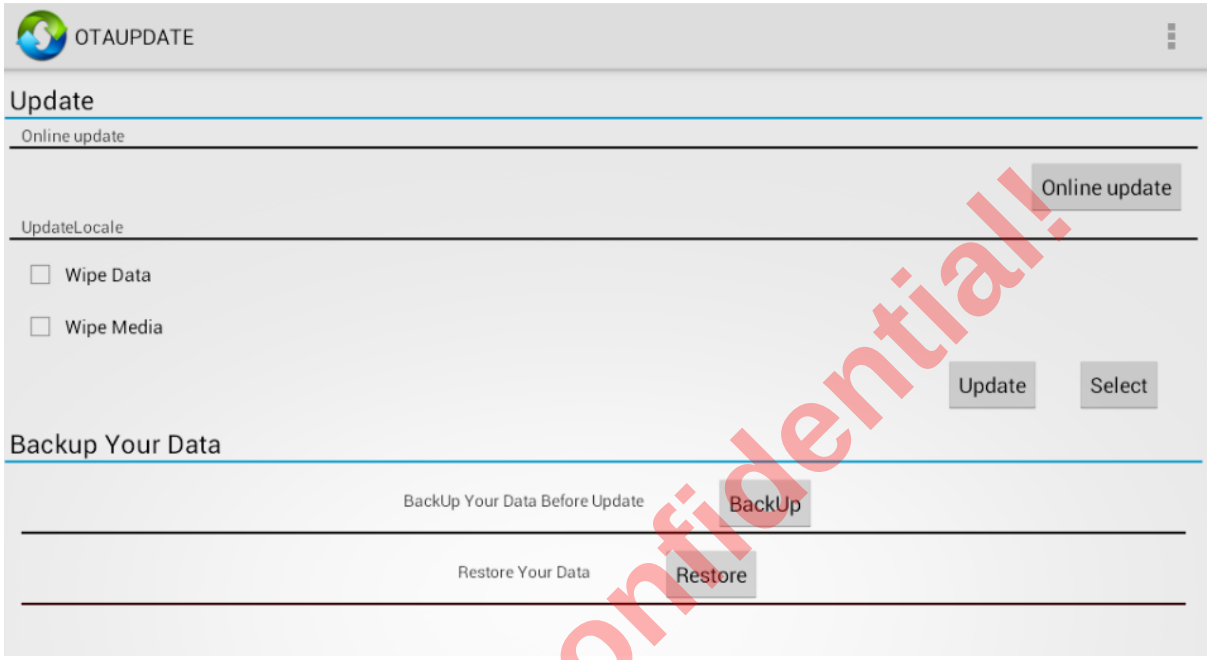
在 system.prop 中 配置升级所需要的固件版本号以及 server 的 url 进行编写。例如：

#upport for OTAUpdate

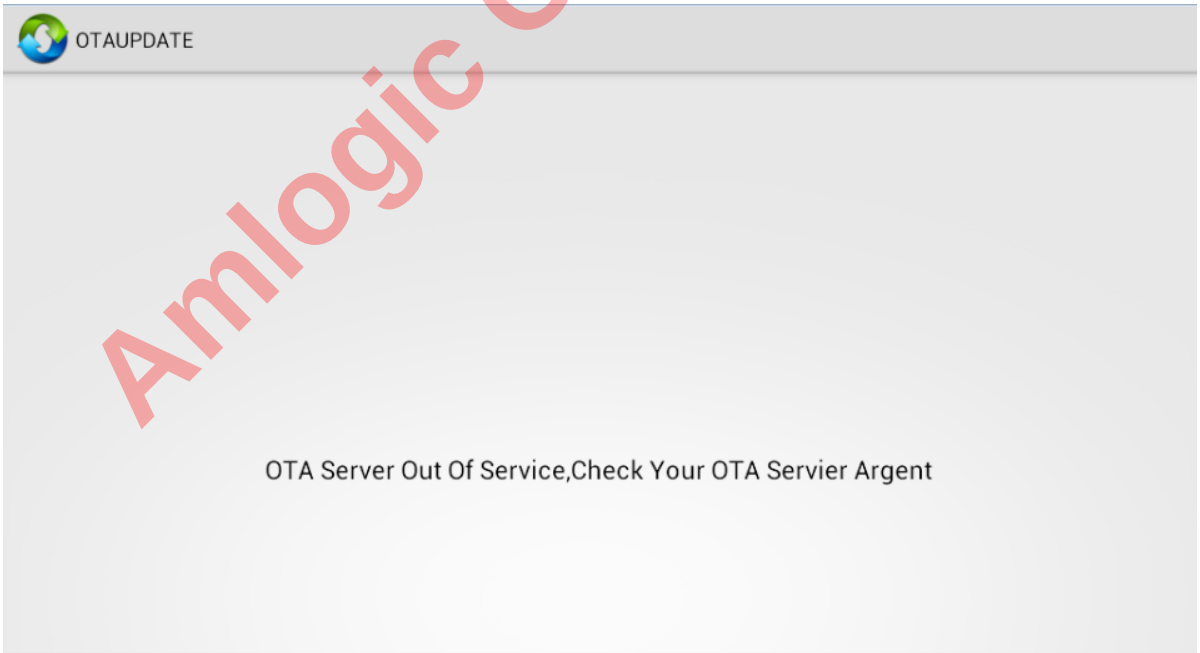
ro.product.firmware=00411001
ro.product.otaupdateurl=http://*.*.*.*:8080/otaupdate/update
上文描述的 url 是根据服务器而变的。Amlogic Demo OTA Server 的地址是上文所描述地址。

#### 3.8.3.3. OTAUpgrade.APK

下面的两张图为其 UI。




Menu 中可以获取到版本信息等辅助操作，点击 Online update 进入在线升级界面。若是服务器不支持此版本的客户端，将会获取到下面的界面。



若是有更新，则会进入更新界面进行下载更新。

# 4. 点对点播放介绍

## 4.1. 简介

在播放高清片源时，特别是在片源分辨率和输出分辨率一致的时候，如果要视频帧无损输出，就需要打开点对点功能，功能打开需要设置: setprop media.p2pplay.enable true；功能关闭需要设置功能打开需要设置: setprop media.p2pplay.enable false

由于需要兼容市面上各种本地播放 APK，点对点功能有一个学习的过程。在安装第三方 APK 后，第一次播视频属于学习过程。然后第二次再用该 APK 播放，已经可以点对点播放了。

如果用户一定要求某些特定 APK，在安装之后，第一次播放就需要点对点，就需要设置黑名单。另外有些客户不要求某些 APK 不支持点对点，或者自己来控制点对点，不由系统来控制，则需要设置白名单。

## 4.2. 黑名单

黑名单就是为了添加默认需要支持点对点的播放器的包名和类名的 xml 文件。

存放的地址是代码中的 frameworks/base/core/res/res/xml/blackpackagefilter.xml.

内容如下：

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<resources>
    <FilterActivity
      packagename="com.funshion.video">
    </FilterActivity>
        <FilterActivity
            packagename="com.qiyi.video">
            <Class name="org.qiyi.android.video.play.impl.mp4.Mp4PlayActivity"/>
            <Class name="com.qiyi.video.ui.VideoPlayActivity"/>
        </FilterActivity>
        <FilterActivity
            packagename="com.sohutv.tv">
            <Class name="com.sohutv.tv.PlayActivity"/>
        </FilterActivity>
</resources>
```

添加黑名单可以分为三种情况：

1) 进入 APK，就关闭 freescale 和 ppscaler。此时在 APK 内的所有 activity 之前切换时，不会开关 free_scale 和 ppscaler。

填写方式如：

```
<FilterActivity
    packagename="com.funshion.video">
</FilterActivity>
```

2) 进入 APK 时，不马上关闭 freescale 和 ppscaler，在切换到需要播放的 activity 时，会关闭 free_scale 和 ppscaler，在退出该 activity 时

又打开 free_scale 和 ppscaler。

```
<FilterActivity
    packagename="com.sohutv.tv">
    <Class name="com.sohutv.tv.PlayActivity"/>
</FilterActivity>
```

3) 在一个 APK 中，有多个 activity 需要关闭 free_scale 和 ppscaler。

```
<FilterActivity
    packagename="com.qiyi.video">
    <Class name="org.qiyi.android.video.play.impl.mp4.Mp4PlayActivity"/>
    <Class name="com.qiyi.video.ui.VideoPlayActivity"/>
```

```
</FilterActivity>
```

## 4.3. 白名单

白名单就是为了添加默认不需要处理点对点流程的包名和类名的 xml 文件，该类 APK 自己处理是否点对点。（如 IPTV APK 和我们的 movieplayer）

存放的地址是代码中的 frameworks/base/core/res/res/xml/whitepackagefilter.xml

内容如下：

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>

<resources>
    <FilterActivity
      packagename="com.farcore.videoplayer">
      <Class name="com.farcore.videoplayer.playermenu"/>
    </FilterActivity>
    <FilterActivity
      packagename="com.amlogic.amvideoplayer">
      <Class name="com.amlogic.amvideoplayer.AmVideoPlayer"/>
      <Class name="com.amlogic.amvideoplayer.playermenu"/>
    </FilterActivity>
    <FilterActivity
      packagename="com.android.browser">
    </FilterActivity>
</resources>
```

添加白名单可以分为三种情况：

1) 进入 APK，就不对 freescale 和 ppscaler 进行操作，保持原来的 freescale 和 ppscaler 状态。

填写方式如：

```
    <FilterActivity
      packagename="com.android.browser">
    </FilterActivity>
```

2) 进入 APK 时，会 enable freescale 和 ppscaler，在切换到指定的 activity 时，会不对 free_scale 和 ppscaler 进行设置，APK 自己设置，等退出该 activity 时会打开 free_scale 和 ppscaler

```
    <FilterActivity
      packagename="com.farcore.videoplayer">
      <Class name="com.farcore.videoplayer.playermenu"/>
    </FilterActivity>
```

3) 在一个 APK 中，有多个 activity 需要进入时不对 free_scale 和 ppscaler 进行设置，退出时打开 free_scale 和 ppscaler。

```
    <FilterActivity
      packagename="com.amlogic.amvideoplayer">
      <Class name="com.amlogic.amvideoplayer.AmVideoPlayer"/>
      <Class name="com.amlogic.amvideoplayer.playermenu"/>
    </FilterActivity>
```

## 4.4. 播放时各 scaler 的使用说明

- M3, M6

1) 点对点打开

```
echo 0 > /sys/class/graphics/fb0/free_scale
echo 0 > /sys/class/ppmgr/ppscaler
```

1080p/i:

echo 7 w/2  h > /sys/class/graphics/fb0/request2XScale
非 1080p/i:
echo 16 w  h > /sys/class/graphics/fb0/request2XScale

2) 点对点关闭
echo 1 > /sys/class/graphics/fb0/free_scale
echo 1 > /sys/class/ppmgr/ppscaler
echo 2 > /sys/class/graphics/fb0/request2XScale
(w 和 h 分别对应设置重现率的宽、高)

- M8:
视频独占 VPP scaler.无需调整各 scaler 状态。