## 系统原Hall逻辑的屏蔽方案

原系统的Hall 功能并非一个独立的应用模块，而是零散的分布在系统的多个应用模块中，我们需要将这些分散的功能模块进行提取、整合，使之完全聚合在一个独立的应用中，这样可以很大程度上的降低系统的耦合度， 提高应用的内聚性。同时这也很好的符合了软件工程中的设计理念，方便了系统的重用，维护，与扩展。

针对我公司项目中的Hall功能模块的整合工作已经完成，该Hall模块目前可以作为一个独立的功能模块嵌入到系统中进行工作。前文我们提到：原Hall功能是分散在多个应用模块中的，因此当我们将整合后的Hall模块嵌入到系统之前，需要先将原系统中分散的多个Hall功能点从系统中移除或屏蔽掉。

原Hall功能主要分布在：SystemUI、Dialer、DeskClock这几个模块中。具体的屏蔽及修改方法可以参看下文的相关步骤。

1. **SystemUI中的相关原文件及修改点**

**1.相关原文件**

----原Hall待机界面布局文件

l9plus\modified\_files\public\frameworks\base\packages\SystemUI\res\layout\keyguard\_hall\_view.xml

----原Hall待机界面控制逻辑在以下两个文件中

l9plus\modified\_files\public\frameworks\base\packages\SystemUI\src\com\android\systemui\statusbar\phone\StatusBarWindowManager.java

l9plus\modified\_files\public\frameworks\base\packages\SystemUI\src\com\android\systemui\statusbar\phone\PhoneStatusBar.java

**2.屏蔽添加hallView的逻辑**

modified\_files\public\frameworks\base\packages\SystemUI\src\com\android\systemui\statusbar\phone\PhoneStatusBar.java 中注释掉以下代码：

// added by Transage shixu on 20161110 begin

//mStatusBarWindowManager.addHallView();

// added by Transage shixu on 20161110 end

**3.屏蔽Hall广播接收器中相关action**

modified\_files\public\frameworks\base\packages\SystemUI\src\com\android\systemui\keyguard\KeyguardViewMediator.java中注释掉下面的广播接收器过滤器中的相关action

//filter.addAction("tecno.keyguard.hall") ;

//filter.addAction(Intent.ACTION\_SCREEN\_ON);

//filter.addAction(Intent.ACTION\_DATE\_CHANGED); //filter.addAction(Intent.ACTION\_UNREAD\_CHANGED); //filter.addAction("com.android.deskclock.ALARM\_ALERT"); //filter.addAction(Intent.ACTION\_TIME\_TICK); //filter.addAction(Intent.ACTION\_TIME\_CHANGED); //filter.addAction("android.telecom.action.SHOW\_MISSED\_CALLS\_NOTIFICATION\_ TO\_SYSTEMUI"); //filter.addAction(TelephonyManager.ACTION\_PHONE\_STATE\_CHANGED); //filter.addAction("com.android.deskclock.ALARM\_SNOOZE"); //filter.addAction("com.android.deskclock.ALARM\_DISMISS"); //filter.addAction("com.android.deskclock.ALARM\_DONE"); mContext.registerReceiver(mBroadcastReceiver, filter);

屏蔽原hall待机界面时，不处理isHallCovered()方法，因为会影响防误触，导致盒盖时候防误触必现。所以我们从上述的广播接收器入手进行屏蔽。

1. **Dialer中的相关原文件及修改点**

**1.相关原文件**

----原Hall电话界面布局文件

l9plus\packages\apps\Dialer\InCallUI\res\layout\hall\_incall.xml

----原Hall电话界面控制逻辑在以下文件中

l9plus\packages\apps\Dialer\InCallUI\src\com\android\incallui\HallAnswerFragment.java

**2.处理isHallCovered（）方法**

packages\apps\Dialer\InCallUI\src\com\android\incallui\util\HallStateUtils.java 中

将以下该方法中的原内容全部注释掉，直接返回false

public static boolean isHallCovered() {

//该方法中的原内容全部注释掉，直接返回false

return false;

}

或者可以直接在 /InCallUI/src/com/android/incallui/AnswerPresenter.java 中

作如下处理：

boolean isHallCovered=HallStateUtils.getInstance().isHallCovered();

Log.d(this, " isHallCovered==" + isHallCovered);

if (activity != null) {

//activity.showAnswerFragment(show,isHallCovered); 1.直接注释掉

activity.showAnswerFragment(show,false); 2.或二参赋false值

//Transage <shixuemei> <2016-11-26> Modify <hall\_call> end

**3.注释掉Hall广播的注册反注册逻辑**

packages\apps\Dialer/InCallUI/src/com/android/incallui/InCallActivity.java中，

注释掉原Hall广播注册逻辑

//Transage <shixuemei> <2016-11-26> add <hall\_call> begin

if (!mReceiverRegistered) {

Log.i("shixuemei", "registerReceiver------");

//final IntentFilter filter = new IntentFilter();

//filter.addAction(TECNO\_KEYGUARD\_HALL\_ACTION);

//registerReceiver(mAnswerReceiver, filter);

// mReceiverRegistered = true;

}

//Transage <shixuemei> <2016-11-26> add <hall\_call> end

广播的反注册逻辑也注释掉

private void unregisterAnswerReceiver(){

//if (mReceiverRegistered) {

// unregisterReceiver(mAnswerReceiver);

// mReceiverRegistered = false;

//}

}

1. **DeskClock中的相关原文件及修改点**

**1.相关原文件**

----原Hall闹钟界面布局文件

l9plus\packages\apps\DeskClock\res\layout\alarm\_activity.xml

----原Hall闹钟界面控制逻辑在以下文件中

l9plus\packages\apps\DeskClock\src\com\android\deskclock\alarms\AlarmActivity.java

**2.注释掉霍尔广播接收action**

modified\_files/public/packages/apps/DeskClock/src/com/android/deskclock/alarms/AlarmActivity.java 中

if (!mReceiverRegistered) {

// Register to get the alarm done/snooze/dismiss intent.

final IntentFilter filter = new IntentFilter(AlarmService.ALARM\_DONE\_ACTION);

filter.addAction(AlarmService.ALARM\_SNOOZE\_ACTION); filter.addAction(AlarmService.ALARM\_DISMISS\_ACTION);

// filter.addAction("tecno.keyguard.hall"); **//注释掉霍尔广播接收action**

registerReceiver(mReceiver, filter);

mReceiverRegistered = true;

}

bindAlarmService();

resetAnimations();

}

// added by Transage shixu on 20161124 begin

private void registerAlarmReceiver(){

if (!mReceiverRegistered) {

// Register to get the alarm done/snooze/dismiss intent.

final IntentFilter filter = new IntentFilter();

filter.addAction(AlarmService.ALARM\_DONE\_ACTION);

filter.addAction(AlarmService.ALARM\_SNOOZE\_ACTION);

filter.addAction(AlarmService.ALARM\_DISMISS\_ACTION);

// filter.addAction("tecno.keyguard.hall"); // **注释掉霍尔广播接收action**

registerReceiver(mReceiver, filter);

mReceiverRegistered = true;

}

}

// added by Transage shixu on 20161124 end

注释isHallCovered()方法内容，直接返回false

public static boolean isHallCovered() {

//注释掉内容，直接返回false

return false;

}

**四、修改Hall广播**

modified\_files\public\frameworks\base\services\core\java\com\android\server\policy\PhoneWindowManager.java

文件中有两个地方的 intent.setAction("tecno.keyguard.hall");

都修改为 intent.setAction("tecno.keyguard.hall\_1");

**五、闹钟标签的 获取方法**

modified\_files\public\packages\apps\DeskClock\src\com\android\deskclock\alarms\AlarmService.java

文件中将private void startAlarm(AlarmInstance instance) 方法中的发送广播 sendBroadcast(new Intent(ALARM\_ALERT\_ACTION));

修改为：sendBroadcast(new Intent(ALARM\_ALERT\_ACTION)

.putExtra("lable",instance.getLabelOrDefault(mContext)));

这样在新的 Hall 的广播接收器中就可接收到 这个闹钟lable了

以上步骤就是系统原Hall逻辑的屏蔽方案了。参看上述步骤可以将原系统中的Hall功能全部移除，然后就可以将整合后的Hall模块嵌入系统中工作了。