获取系统通知内容的解决方案

1、 技术预研目标：

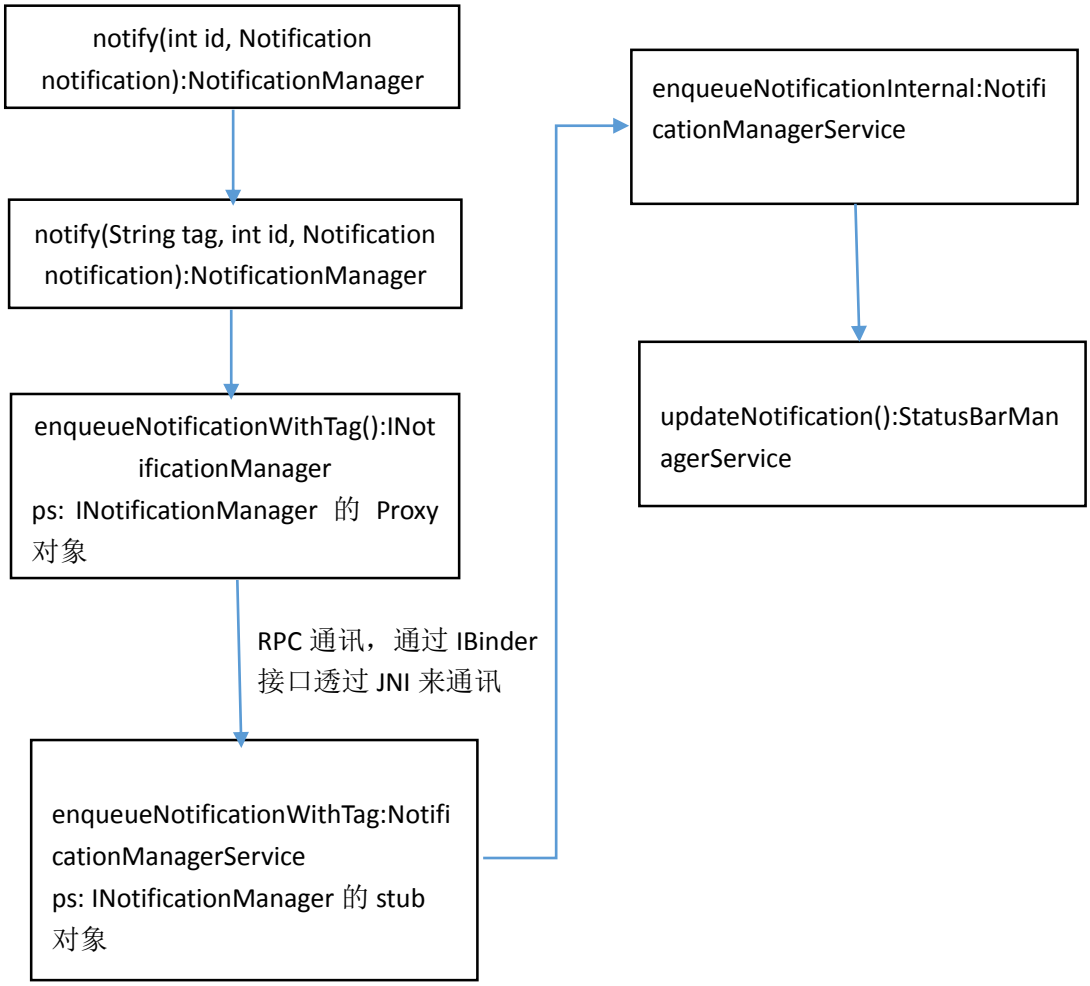
研究如何获取系统通知的内容，包括所有应用和系统 App 发出来的通知，要求兼容至 Android 2.3（API 10）,使用这些解决方案有没有什么限制，如需要向系统申请什么权限。

2、 预研结论：

4.3 版本以上(包括 4.3 版本)可以通过系统提供的接口 NotificationListenerService 来完成，但需要用户手动在设置里开启 App 的 Notification Access 权限。4.0 版本以上(包括 4.0 版本)也可以通过 Android 系统的一个钩子 AccessibilityService 来完成,但需要用户手动在设置里开启 App Accessbility 权限。4.0 版本以下没有 NotificationListenerService，尝试使用 AccessibilityService，但是在 2.3 版本运行时有安全异常无法解决。

3、 分析过程：

通知的显示过程：(冒号表示方法所在的类)



4、 方案

方案一:

在 enqueueNotificationInternal 方法有这样一段代码:

```
if (notification.icon != 0) {
    StatusBarNotification n = new StatusBarNotification(pkg, id, tag,
            r.uid, r.initialPid, notification);
    if (old != null && old.statusBarKey != null) {
        r.statusBarKey = old.statusBarKey;
        long identity = Binder.clearCallingIdentity();
        try {
            mStatusBar.updateNotification(r.statusBarKey, n);
        }
        finally {
            Binder.restoreCallingIdentity(identity);
        }
    } else {
        long identity = Binder.clearCallingIdentity();
        try {
            r.statusBarKey = mStatusBar.addNotification(n);
            mAttentionLight.pulse();
        }
        finally {
            Binder.restoreCallingIdentity(identity);
        }
    }
    sendAccessibilityEvent(notification, pkg);
```

最后一行代码 sendAccessibilityEvent()代表可以使用 Android 的钩子来或取系统通知,该方法来自 AccessibilityManager,其回调为 AccessbilityService 中的 abstract void onAccessibilityEvent(AccessibilityEvent event)。

AccessibilityService 简介

AccessibilityService 是一个辅助类，可以监听我们手机的焦点，窗口变化，按钮点击等等。实现它的服务需要在手机设置里面->辅助功能在这里面找到你自己实现的辅助类，然后打开它就可以进行我们一系列的监听了。**通过该类可以取到 notification 的 ticker 信息，触发时间和触发源的 app 的包名，但是通知的内容、标题，图标声音等配置无法取到。通过该方法，需要用户手动在 settings>accessibillity 里开启 App 的 Accessibillity 权限，**

公共方法

abstract void onAccessibilityEvent(AccessibilityEvent event)

参数：event 一个事件,里面可以拿到 notification 的 ticker 信息，触发时间和触发源的 app 的包名.

public final IBinder onBind (Intent intent)实现返回一个内部的辅助接口的实现，子类不能被重写。

参数：intent 与服务相绑定的意图，注意其他任何包含在 Intent 的外部意图将不能在此使用。

返回值：返回一个客户端可以在服务上访问的 IBinder。

public abstract void onInterrupt () 打断辅助回馈内容时呼叫。

受保护方法

protected void onServiceConnected ()这个方法是 AccessibilityService 声明周期的一部分，在系统成功与服务绑定后才被呼叫，如果用来设定 AccessibilityServiceInfo.这个方法更为方便。

详细代码:

onAccessibilityEvent 的实现：

```
public void onAccessibilityEvent(AccessibilityEvent event) {
        List<CharSequence> list=event.getText();
        StringBuilder sb=new StringBuilder();
        for(int i=0,len=list.size();i<len;i++){
            sb.append(list.get(i)).append(",");
        }
        sb.deleteCharAt(sb.length()-1);
        Log.i(tag,"text:"+sb.toString());
    }
```

Activity 中启动该 Service 的方法:

```
  bindService(new Intent(this,NoficationService.class), new ServiceConnection() {

            public void onServiceDisconnected(ComponentName name) {


            }

            public void onServiceConnected(ComponentName name, IBinder service) {
                // TODO Auto-generated method stub
                tv_msg.setText("onServiceConnected success!");
            }
        },Service.BIND_AUTO_CREATE);
```

配置：

```
<service
 android:name="com.jason.Service.NoficationService"
 android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE" >
  <intent-filter>
        <action android:name="android.accessibilityservice.AccessibilityService" />
    </intent-filter>
 </service>
```

存在问题:

在 2.3 版本运行会出现该错误: java.lang.SecurityException: Not allowed to bind to service Intent

方案二:

Android在4.3的版本中(即API 18)加入了NotificationListenerService，根据SDK的描述(AndroidDeveloper)可以知道，当系统收到新的通知或者通知被删除时，会触发

NotificationListenerService的回调方法。同时在Android 4.4 中新增了Notification.extras字段，**也就是说可以使用NotificationListenerService获取系统通知具体信息,触发时间，触发源包名等等信息**，这在以前是需要用反射来实现的。

NotificationListenerService 主要方法(成员变量)：

cancelAllNotifications() ：删除系统中所有可被清除的通知；

cancelNotification(String pkg, String tag, int id) ：删除具体某一个通知；

getActiveNotifications() ：返回当前系统所有通知到 StatusBarNotification[]；

onNotificationPosted(StatusBarNotification sbn) ：当系统收到新的通知后出发回调；

onNotificationRemoved(StatusBarNotification sbn) ：当系统通知被删掉后出发回调；

以上是 NotificationListenerService 的主要方法，通过这些方法就可以在应用中操作系统通知，在 NotificationListenerService 中除了对通知的操作之外，还可以获取到通知的 StatusBarNotification 对象，通过该对象可以获取通知更详细的数据。

StatusBarNotification 主要方法(成员变量)：

getId()：返回通知对应的 id；

getNotification()：返回通知对象；

getPackageName()：返回通知对应的包名；

getPostTime()：返回通知发起的时间；

getTag()：返回通知的 Tag，如果没有设置返回 null；

getUserId()：返回 UserId，用于多用户场景；

isClearable()：返回该通知是否可被清楚，FLAG_ONGOING_EVENT、FLAG_NO_CLEAR；

isOngoing()：检查该通知的 flag 是否为 FLAG_ONGOING_EVENT；

详细代码:

```
public class NoficationService extends NotificationListenerService {
    private static final String TAG = "NoficationService";

    @Override
    public void onNotificationPosted(StatusBarNotification sbn) {
        // TODO Auto-generated method stub
        Log.i(TAG, "Posted:" + sbn.getNotification().tickerText.toString());
    }
```

```
    @Override
    public void onNotificationRemoved(StatusBarNotification sbn) {
        // TODO Auto-generated method stub
        Log.i(TAG, "Remove:" + sbn.getNotification().tickerText.toString());
    }
}
```

配置:

```
<service
        android:name="com.example.notificationmonitor.NoficationService"

        android:permission="android.permission.BIND_NOTIFICATION_LISTENER_SER
        VICE" >
            <intent-filter>
        <action android:name="android.service.notification.NotificationListenerService" />
             </intent-filter>
 </service>
```

Activity 里启动该 Service 的代码:

```
        this.bindService(new Intent(this, NoficationService.class),
                        new ServiceConnection() {

                            @Override
                            public void onServiceDisconnected(ComponentName name) {

                            }

                            @Override
                            public void onServiceConnected(ComponentName name,
                                    IBinder service) {
                                // TODO Auto-generated method stub
                                Log.i(TAG, "onServiceConnected");
                            }
                        }, Service.BIND_AUTO_CREATE);
```

最后，需要打开该应用的 Notification Access。当手机上没有安装任何使用 NotificationListenerService 的应用时，系统默认不会显示"Notification access"选项。只有手机中安装了使用 NotificationListenerService 的应用，才**可以在"Settings > Security > Notification access" 找到对应的设置页面**。也可以手动引导用户点击打开，其 activity action 为 android.settings.ACTION_NOTIFICATION_LISTENER_SETTINGS。

后续扩展:

为了提高该方案的可行性，继续研究如何用代码打开应用的 Notification Access,在系统设置里源码程序找到开启 Notification Access 的代码，提取为以下方法:

```
    private void switchOnPermission() {
        // TODO Auto-generated method stub
        ComponentName cn = new ComponentName(this, NoficationService.class);
```

```
            StringBuilder sb = new StringBuilder();
            sb.append(':').append(cn.flattenToString());
            Settings.Secure.putString(mCR, ENABLED_NOTIFICATION_LISTENERS,
                    sb != null ? sb.toString() : "");
    }
```

需要的权限:

```
<uses-permission android:name="android.permission.WRITE_SETTINGS"/>
<uses-permission android:name="android.permission.WRITE_SECURE_SETTINGS"/>
```

存在问题:

<uses-permission android:name="android.permission.WRITE_SECURE_SETTINGS"/>只能是 system app 才能使用。需想办法把 app 变为 system app.

5、 两种方案对比:

方案二可以取到通知的全部信息，但仅限于 4.3 版本以上(包括 4.3);方案一可以兼容低版本，但 2.3 版本无法兼容，能获取到的信息也有限，只有通知的 ticker 内容、触发时间和触发源的包名。两种方案都需要用户手动去开启对应所需的权限，除非是 System App,但把 App 变为 System App，偏离了 uc 的开发方向。