SystemUI 模块分析

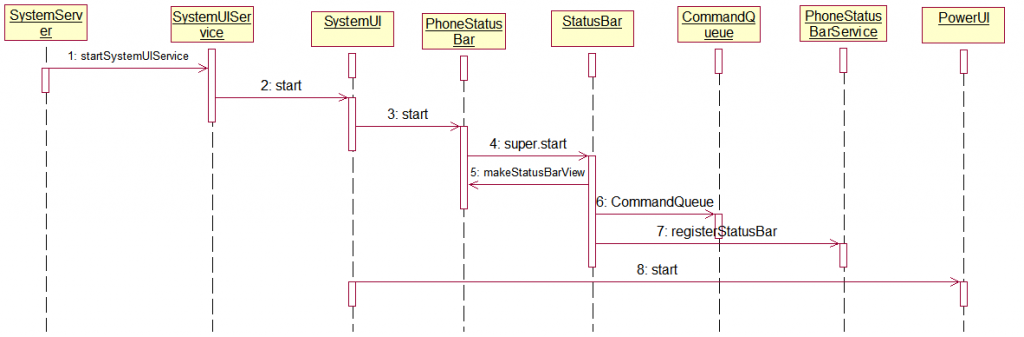
1. SystemUI 定义

对于Phone来说SystemUI指的是：StatusBar(状态栏)、NavigationBar(导航栏)。而对于Tablet或者是TV来说SystemUI指的是：CombinedBar(包括了StatusBar和NavigationBar)。

SystemUI作为一个service长存系统中，反馈系统状态。

1. SystemUI的启动

systemserver.java启动了systemservice，接着就注册了状态管理服务和通知管理服务。

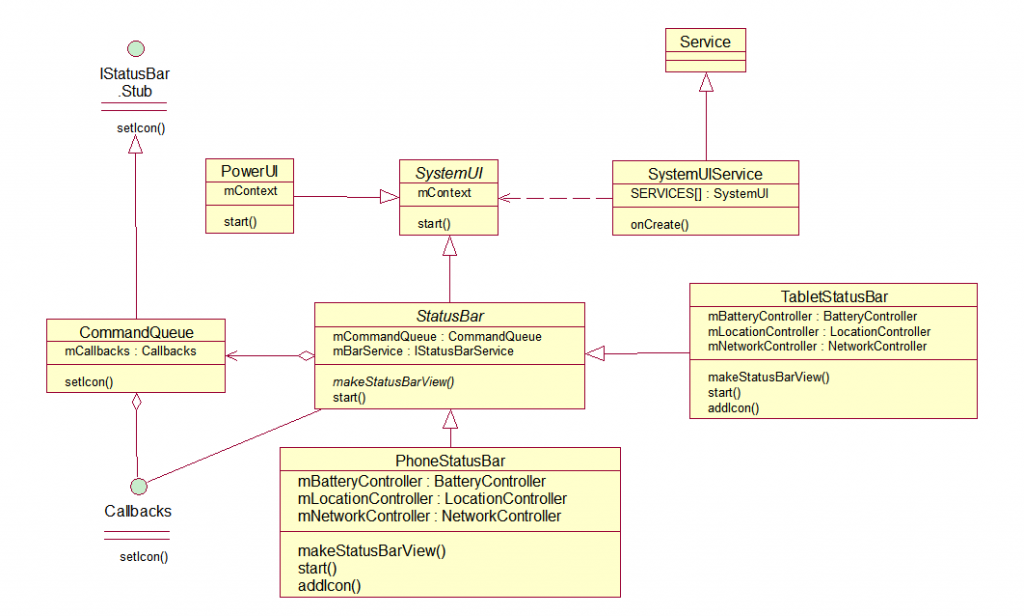


在init启动系统的框架服务（ActivityManagerService，PackageManagerService等）后，SystemServer启动SystemUI服务（注：SystemServer中调用了startSystemUI()方法,此方法中通过context.start方法开始启动SystemUIService）。

在start启动service时调用SystemUIService（\frameworks\base\packages\SystemUI\src\com\android\systemui）的onCreate（）方法创建service，在创建service时做了两件事，一是判断设备是手机还是平板，确定具体的状态栏操控类。二是调用已确定SystemUI的start方法启动状态栏。

疑问就来了，那是怎么判断设备类型的呢？跟踪代码，判断依据是根据显示分辨率来判断的，具体判断流程在phoneWindowManager.Java中（android4.3 是在phoneWindowManager.Java的setInitialDisplaySize（）方法）。

1. SystemUI的类结构



Statusbar模块分析

1. 启动流程
   1. StatusbarService的启动流程

系统进程system\_process启动之后，调用系统SystemServer.java，在SystemServer中运行ServerThread.run()方法时会注册StatusBarManagerService，接着在StatusBarManagerService中会启动StatusbarService

* 1. 系统图标初始化流程

在启动StatusbarService后，StatusbarService会调用makeStatusBarView的方法，在里面将创建StatusBarView，在创建StatusbarVIew的过程中会加载系统图标。在启动 StatusbarService 的过程中会创建 StatusBarPolicy 的对象，StatusBarPolicy.java 主要负责状态栏显示策略的管理（如状态栏的图标什么时候显示，在什么位置显示等）。StatusBarPolicy 的构造函数中初始化了很多系统图标（如电池信息图标，闹钟图标，声音图标，信号栏图标等）。默认时有很多图标是不显示的，需要显示的时候再进行更新。

StatusBarManager是客户端使用的状态栏管理类，通过IBinder机制跨进程调用StatusBarManagerService，StatusBarManagerService派生于IStatusBarService.Stub，是状态栏管理的服务端，是具体实现。

1. 通知处理

在应用activity中实现通知栏图标变化的程序中，是用NotificationManager对象mNotificationManager 来发送通知。

调用StatusBarManagerService.java中的addNotification方法->（mBar不为空的话）执行mBar.addNotification(key, notification);->对应的是CommandQueue中的addNotification(IBinder key, StatusBarNotification notification)->CommandQueue中的mCallbacks.addNotification(ne.key, ne.notification);->StatusBarService中的addNotification

1. 图标更新

StatusBarPolicy 调用 registerReceiver 注册了感兴趣的intent, 当感兴趣的intent 发生时，对图标进行更新。例如，设置一个闹钟后，闹钟模块会发出一个叫做Intent.ACTION\_ALARM\_CHANGED 的广播，然后StatusBarPolicy 接收到此广播，继而更新状态栏上的闹钟图标。StatusBarPolicy 只是一个策略管理，实际的功能是StatusBarService 来实现的。

StatusBarService 初始化时初始化了一个用于显示statusbar 的StatusBarView 。StatusBarView 里面定义了icon 名字，的显示顺序，对应的png 图等，在StatusBarService 调用makeStatusBarView 方法时实现statusbar 的初始化。

**总结**：

1. **StatusBarManagerService是服务端StatusBarService的管理者**

StatusBarManagerService是StatusBarService的管理者，是StatusBarService与外界通信的桥梁，在StatusBarManagerService.java中，有addNotification ， removeNotification,updateNotification 等方法用于管理传递给他的通知对象。这个类是一些管理方法，实际执行相关动作的是在 IStatusBar.java 里面，这个是 framework/base/core/java/com /android/internal/statusbar/IStatusBar.aidl 自动生成的用于 IPC 的类。

**注**：IPC（Inter-Process Conmmunication）首先想到的就是Handler和Looper，Handler用于多进程之间的通信和数据交换，将各进程之间通信的数据 Message放置到Message Queue里，而Looper用于创建各进程自身的message queue，然后在适当的时候分发给相应的进程。

1. **StatusBarService是Statusbar的核心**

StatusBarService 这个服务是Statusbar 模块的中心点，所有关于图标的加载、更新、删除等处理，与应用的交互，对通知信息的处理，动画的完成等都是建立在StatusBarService 这个基础之上的。

1. **StatusBarPolicy** **负责状态栏显示的策略管理**

Android 中状态栏上有很多图标，这些图标什么时候显示什么时候不显示 ，这些都是 StatusBarPolicy 来管理的。  
StatusBarPolicy 的构造函数里初始化了好几个图标，如闹钟icon ，信号栏icon 等。默认时有很多图标是不显示的，需要显示时再进行更新。StatusBarPolicy 调用 registerReceiver 注册了感兴趣的intent, 当感兴趣的intent 发生时，对图标进行更新。

StatusBarPolicy 只是一个策略管理，实际的功能是StatusBarService 来实现的。StatusBarService 初始化时初始化了一个用于显示statusbar 的StatusBarView 。StatusBarView 里面定义了icon 名字，的显示顺序，对应的png 图等，在StatusBarService 调用makeStatusBarView 方法时实现statusbar 的初始化。

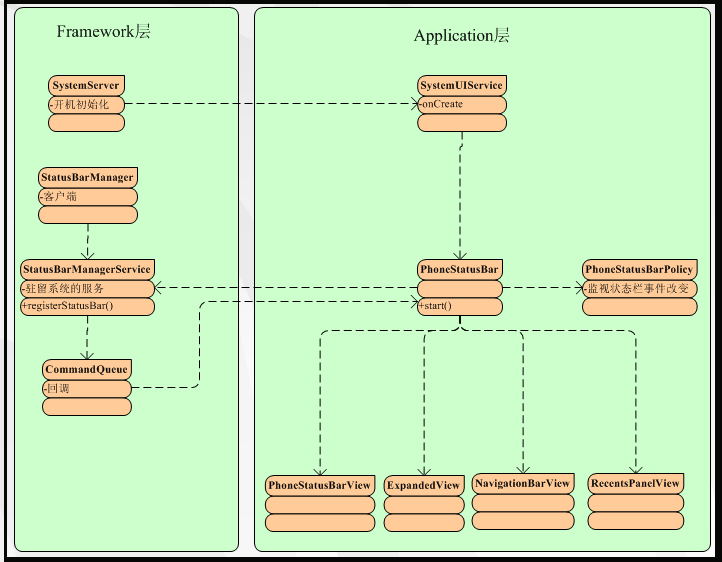
1. **CommandQueue是StatusBarService和StatusBarManagerService交互的枢纽**

IStatusBar.java 里面对应的方法是用 CommandQueue 的接口 callback 的形式调用的， callback 的实现在对应的服务提供类也就是 StatusBarService.java 中提供的。

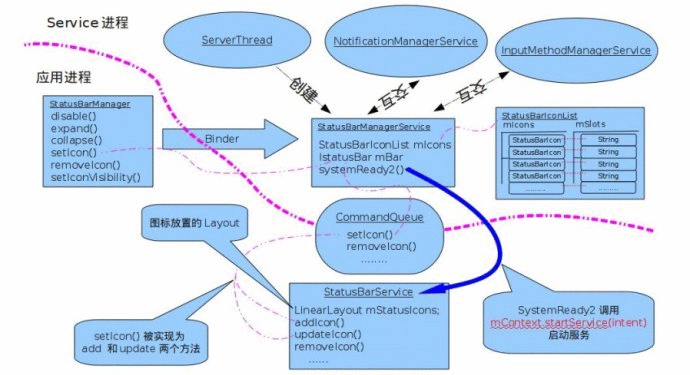
最终执行状态栏更新通知等事件都是在实现的 CommandQueue.Callbacks 里面执行。

statusBarManagerService是一个系统服务，这个服务能通过回调来操作状态栏，使用时一般调用它的客户端——StatusBarManager方法

五、**结构流程图**



六、**交互关系图**



**分析Android 4.3 的状态栏的启动流程**：

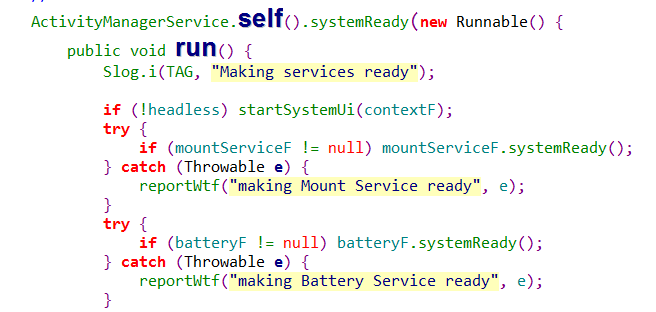
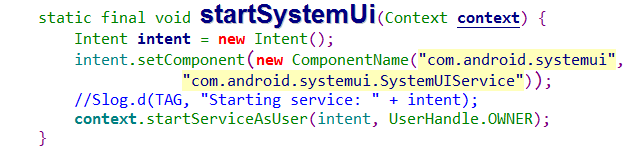
不同版本的android，有不同风格的，从SystemUi的代码分类我就可以看出来，google考虑了不同情况下状态栏的显示等，在源代码里面有这么两个文件夹需要注意:

com.android.systemui.statusbar.phone这个是针对手机而需要的status bar

com.android.systemui.statusbar.tablet这个是针对平板电脑而需要的system bar。

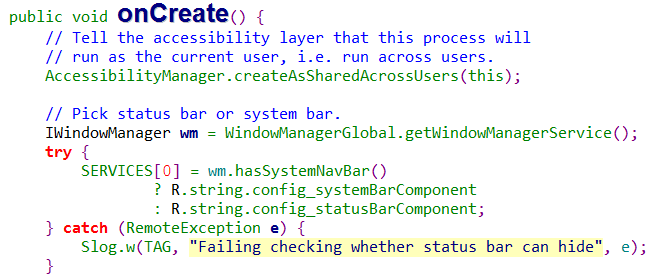
我们都知道系统刚启动，在SystemServer会加载系统的各种服务，状态栏也不例外，就是在这个时候创建的：

SystemServer.java

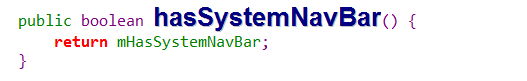
 

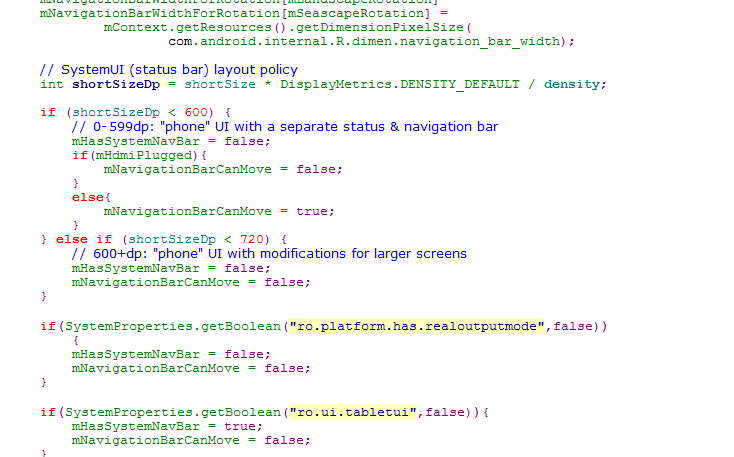
通过上面的代码我就看到系统启动了com.android.systemui.SystemUIService 这个服务，在这个服务里面就会启动状态栏，

在SystemUIService.java文件中会判断选择什么类型的状态栏。



上面的wm.hasSystemNavBar()会调用到phoneWindowManager.java里面对应的那个函数，在那里面会根据配置文件，显示设备宽高等信息来判断使用哪一种bar，代码如下:





这里因为显示设备的宽（高）大于720dp，就认为是平板了，这样mHasSystemNavBar = true;

**注：但是在此处amlogic不知道为什么又加了一些属性字段来对**mHasSystemNavBar来进行限制。（此处很让人费解，导致通知栏无法显示出来）

我们回到这里



这样SERVICES[0]  = R.string.config\_systemBarComponent



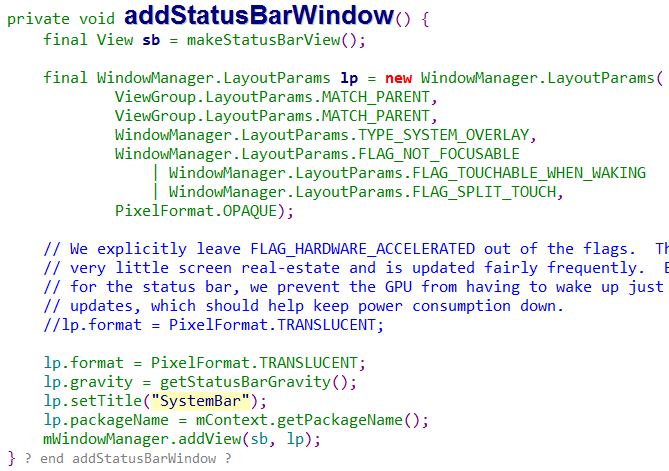
字符串就是：com.android.systemui.statusbar.tablet.TabletStatusBar

实际上SERVERCES这个数组里面就两个元素，一个是我们上面的那个com.android.systemui.statusbar.tablet.TabletStatusBar，另外一个是PowerUI，这个暂且不管，这个主要是根据电量等信息弹出一些提示框，比如电量低，或者充电器有问题等框。

接下来会调用com.android.systemui.statusbar.tablet.TabletStatusBar的start方法，注意com.android.systemui.statusbar.tablet.TabletStatusBar

这个类不是一个服务，就是一个普通的类，start方法也是它的一个普通的方法而已。

在start方法里面就会创建状态栏那些界面对应的各种view，包括下拉状态栏后的view等，把这些view都创建好以后，就把这个view加载windowmanager里面就可以了，这样状态栏就可以显示出来了，代码如下：



这样，状态栏的整个启动过程就完成了。