1、项目背景

外部第三方（例如运营商 inside等等）需要可配置

iot设备上其他的自研模块（例如：通话模块、儿童模式、开机启动等等）也需要自定义禁用和启用的功能

但是如果涉及到这些需求的更改，由于技术方案实现的原因，都需要systemui这边评估，然后重新拉分支，更新版本，才可以满足这些业务场景的需求。而且一旦出现了状态栏或者左滑条无法显示的问题，测也拉动模块来一起看问题，一般还需要第三方业务转交bug到systemui 造成人员参与的一个复杂度。

2、systemui之前的技术方案

实现时序图如下

使用辅助功能（AccessibilityService）监控当前 Active Window，然后利用回调函数 onAccessibilityEvent 过滤事件类型 TYPE_WINDOW_STATE_CHANGED 获知当前 window 改变，通过调用ActivityManager.getRunningTasks()获取当前最前台应用的Activity。

代码块

```
1
```

获取到当前最前台应用的Activity后，判断当前界面是否需要禁用快捷控制页&左滑返回|回桌面等快捷控制功能。

代码块

```
1  public static ComponentName getTopActivity(Context c) {
2          ActivityManager am = (ActivityManager) c
3                  .getSystemService(Context.ACTIVITY_SERVICE);
4          if (am != null) {
5              List<ActivityManager.RunningTaskInfo> infos
6                      = am.getRunningTasks(1);
7              if (infos != null && infos.size() != 0) {
8                  return infos.get(0).topActivity;
9              }
10         }
11         return null;
12     }
```

3、在开发以及维护过程中，发现存在的问题有以下几点

1、代码实现逻辑全放在SystemUI模块，造成了systemUI和其他模块的耦合程度过高；

2、老的方案是通过监听界面的变化，但是在用户进行操作的时候，界面的变化恰恰是超级频繁的，而且这些界面的变化很多都是不需要systemUI进行处理的，造成了需要过滤掉很多的噪声信息。

3、systemUI每次在监听到上层界面的变化时，都要通过ActivityManager.getRunningTasks()来获取当前最顶层的Activity，造成CPU的使用会过高。

4、针对第三方模块，如果是Activity里内嵌Fragment的界面实现。用户进入这个三方模块的时候，SystemUI能监听到的是当前处于该Activity，但是并不知道用户界面已经切换到哪一个Fragment里。而有些业务场景需求不是在该Acitivity的所有界面都禁用下拉和左滑操作，而是在该Acitivity的某一个Fragment界面禁用，所以在开发过程中，也遇到测试反馈很多类似的这些Bug，这也是该技术方案的一个很大的缺陷。

4、面临以下几个问题?

新方案选型。应该采用何种新方案？既保证工作量改动较小，同时又不会引起新的问题？
如何保证该方案对于其他三方的可实施性？简单可快速集成？
改进后的实现逻辑如何和团队阐述清楚？
在阐述清楚的基础上，如何推动以及说服其他业务方一起跟随修改方案？

5、如何去解决当前遇到的问题?

1、新方案选型：前期调研

网上搜索了目前是否有其他行业也有这种业务场景需求，以及他们是如何在实现的？

搜索Android原生SystemUI是否提供类似的能力，已达到一个复用的目的？

2、先更新systemUI本模块内的代码，将可配置的能力提供出去之后。尝试写了一个简单的三方模块Demo，**来集成这个可配置的能力。通过亲自的Demo实践来做问题的验证：是否满足当前的业务需求，是否会带来新的问题，技术变更的工作量&开发时间**。摸底其他模块更新新方案的开发代价，做到心理有数，也可以在此基础上更好的说服别人进行方案的更新。

3、为了让团队内其他模块的开发更容易了解以及接受当前的技术方案，通过语雀文档（**阿里内部的知识库共享平台**）撰写的方式，将上述的项目背景、遇到的问题、以及方案更新的原因，第三方如何使用这个提供的能力等等。为了使得该方案可以快速得到其他小组成员的认可，先是在本小组内部进行了一轮分享，收集组内其他成员的意见，针对不好的地方以及需要补充的知识点，进行了一轮完善后。利用每周的分享会，主动和团队的所有成员进行了一轮技术的分享，针对大家有问题的点进行一次讲解。

4、新的方案需要其他三方一起修改，虽然代码的修改工作量并不大，但是只要是涉及到开发工作，势必就要有合理的理由以及数据为基础来达到说服别人的目的。为了达到推动大家一起配合的目的，**前期也做了很多技术的调研、文档的撰写、数据的整理、第三方需求的整理**

6、最终新的技术方案

业务时序图如下：

代码逻辑：

解决的问题：

最后，我想说，这个技术方案的更新其实并不是项目PD的需求，或者说是领导要求我做的。而是我在模块代码需求开发以及维护过程中发现的问题，基于自己的开发经验，以及参考其他行业的实现方案，做出思考&权衡后觉得有必要花精力去做的一件事情，所以整一个事情需要我这边主动来牵头，主动咨询周围有经验的同事，吸收和采纳他们的意见。也通过向上反馈，向周围同事分享&沟通，用实际的业务场景需求，以及更新后可以带来的好处，来说服他们配合我一起逐步进行集成。在这件事情中，通过自己的努力，看到别人对我的认可，满足自己工作成就感的同时，也确实帮助业务后续的一个快速迭代和三方扩展，觉得是一件很有意义的事情。