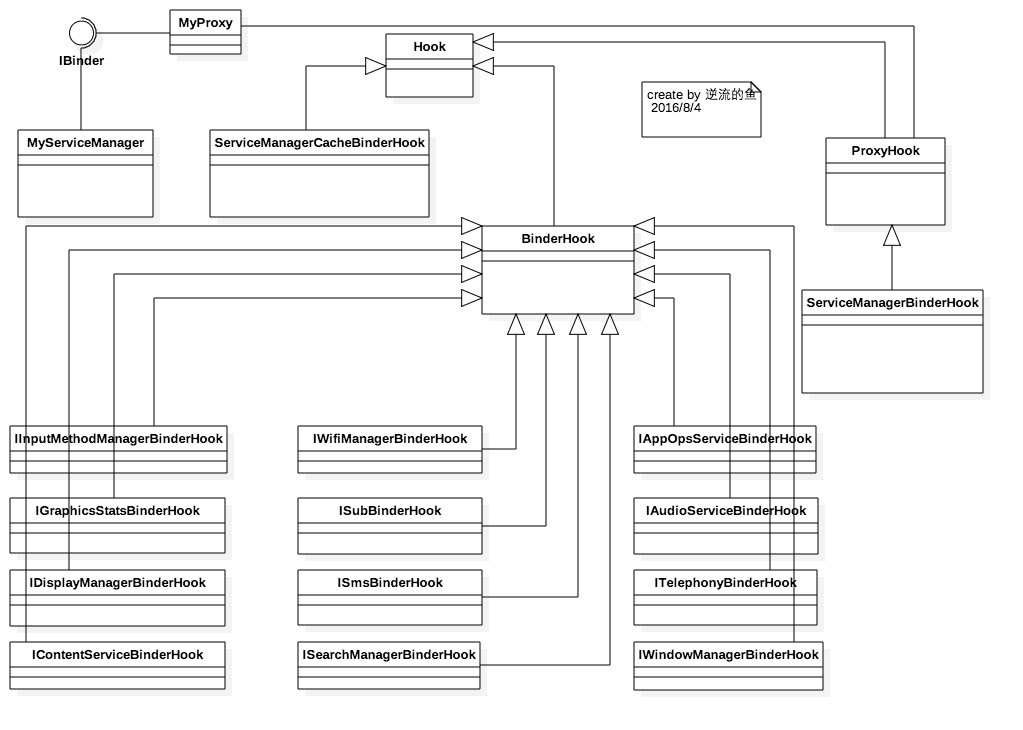
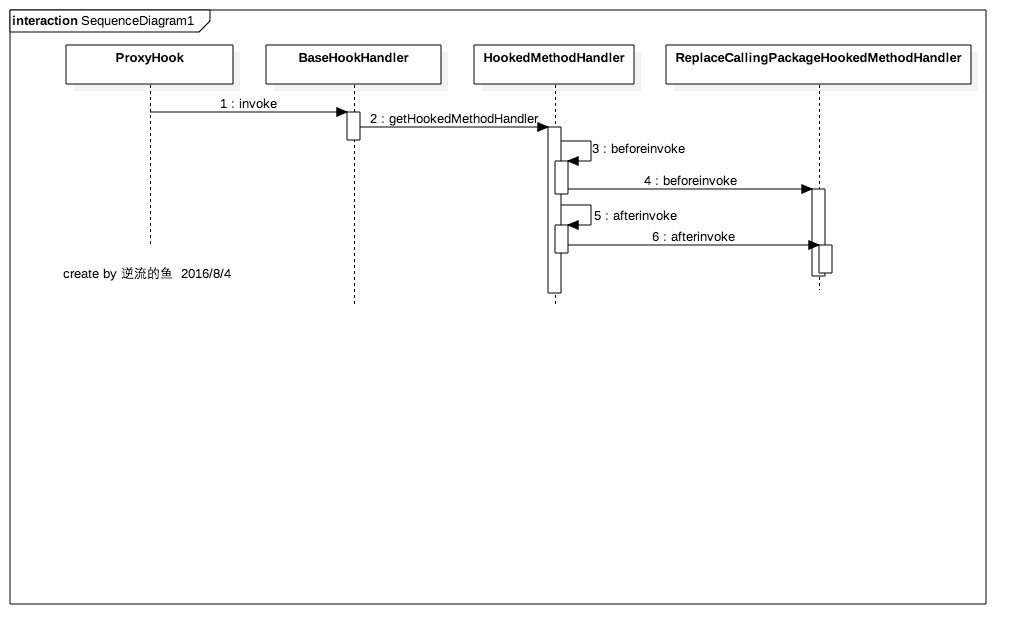
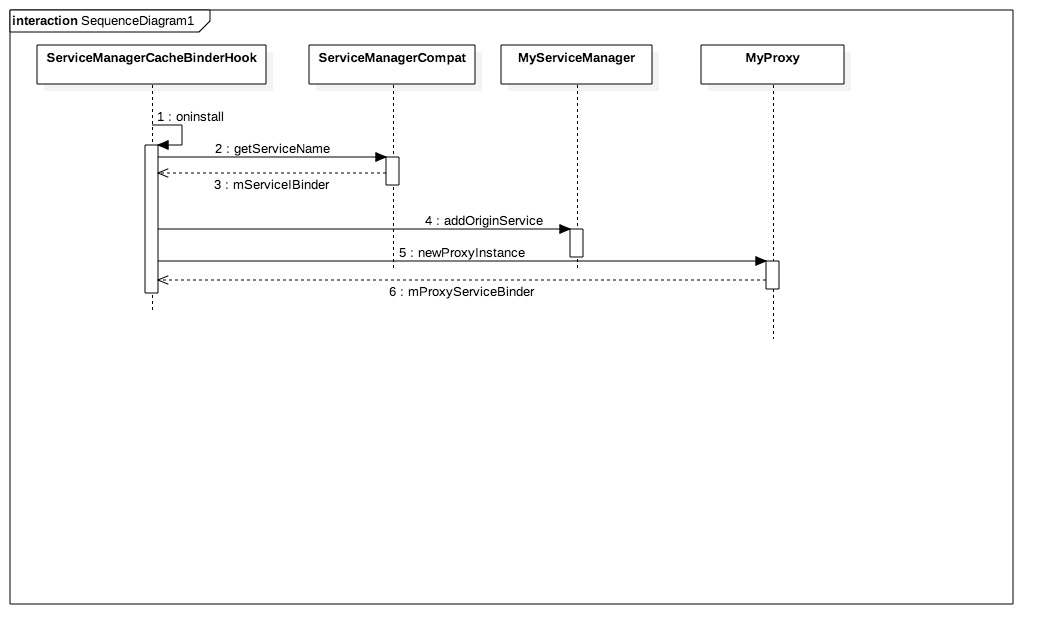
[插件开发之360 DroidPlugin源码分析（三）Binder代理](http://blog.csdn.net/hejjunlin/article/details/52138483)

* **Hook机制中Binder代理类关系图**
* **Hook机制中Binder代理时序图**
* **MyServiceManager**
* **ServiceManagerCacheBinderHook**
* **ServiceManagerBinderHook**
* **BinderHook**

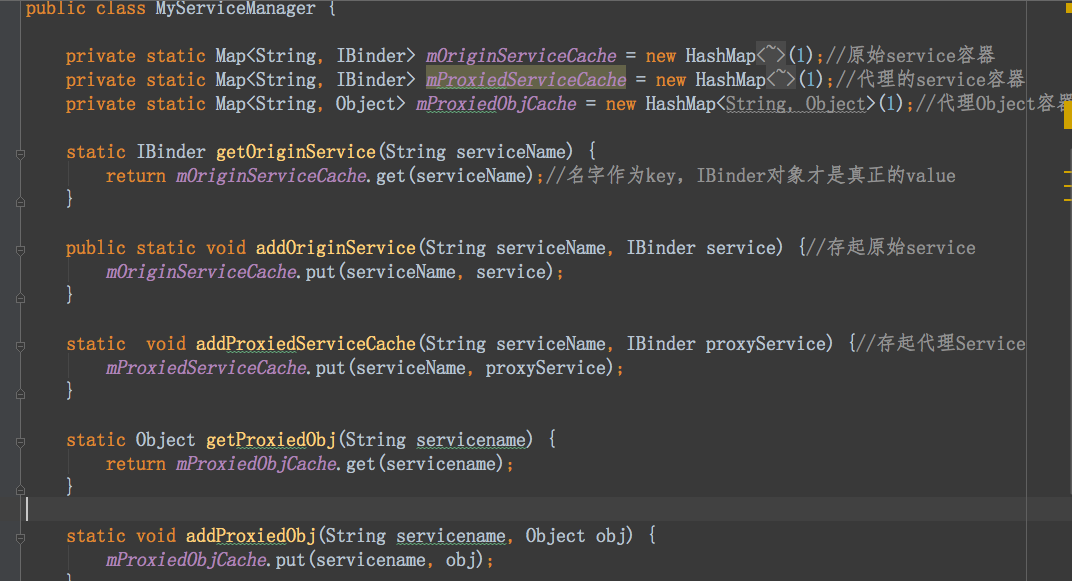
## Hook机制中Binder代理类关系图



## Hook机制中Binder代理时序图

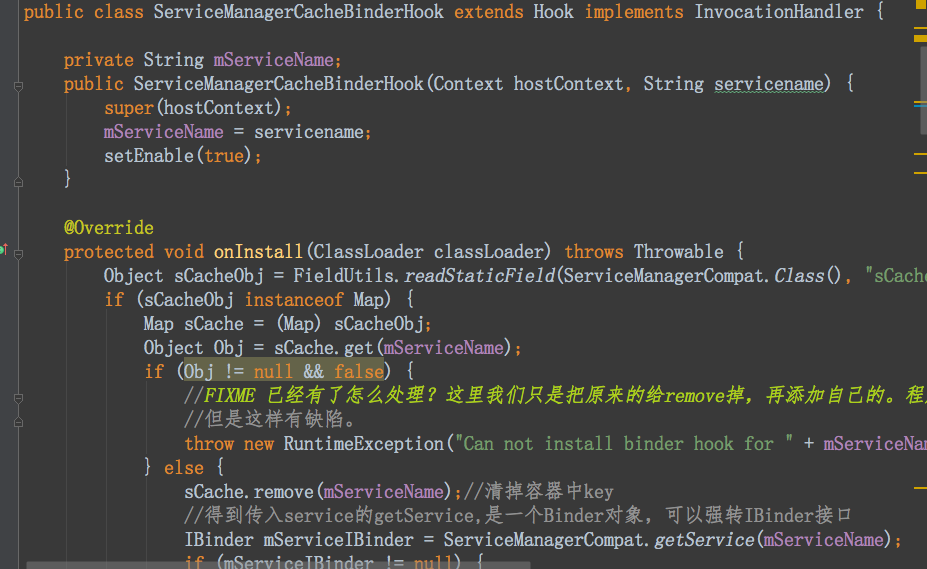
  


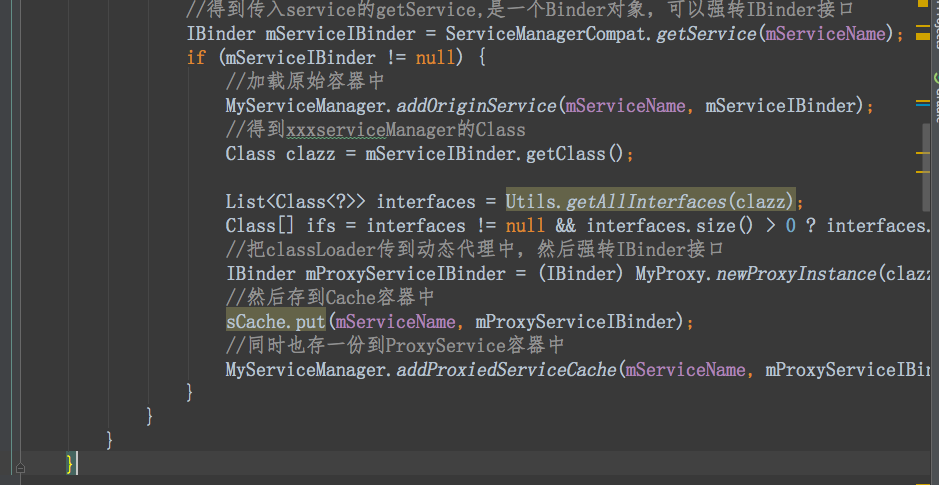
## MyServiceManager

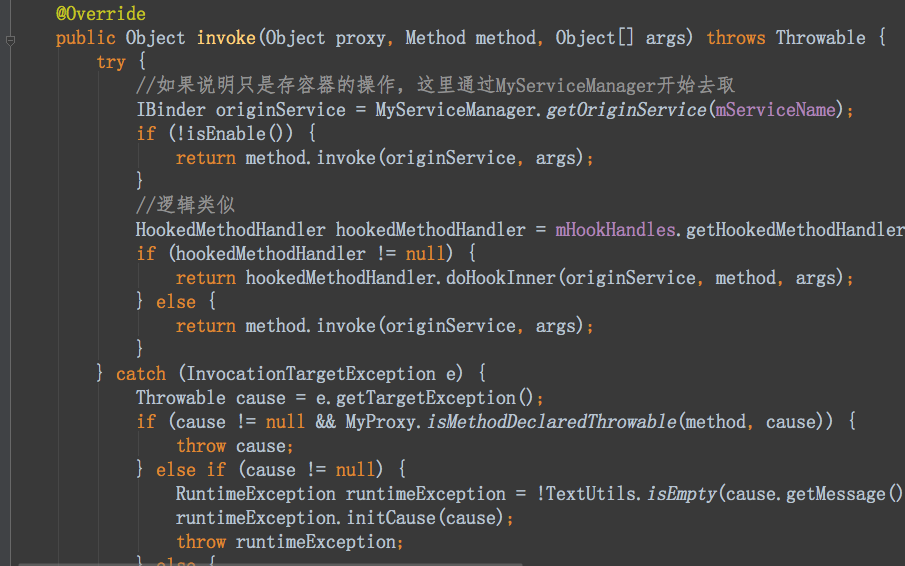


* **mOriginServiceCache：这里存储的是原始的service cache。每个ActivityThread在bindApplication()的时候，会从ServiceManager那边获得一个service cache（可以减少和Binder代理之间通信，系统Binder是一个专门的BinderProxy和把上层的service和Binder driver进行IPC），每次要和某个service通信时，会先检查这个cache里有没有代理对象，如果有的话就直接用，不需要再和ServiceManager进行一次binder交互了。**
* **mProxiedServiceCache：这里存储的就是service cache的代理对象了，因为我们要hook这些binder和上层（serviceConnection时会转成IBinder接口）调用，所以必须把service cache也替换成我们的代理对象，每次调用都会走进ServiceManagerCacheBinderHook对象的invoke()方法。**
* **mProxiedObjCache：这里存储的是所有的proxyservice Object，那原始的service对象放在哪里呢？其实是在BinderHook的mOldObj里。**

## ServiceManagerCacheBinderHook



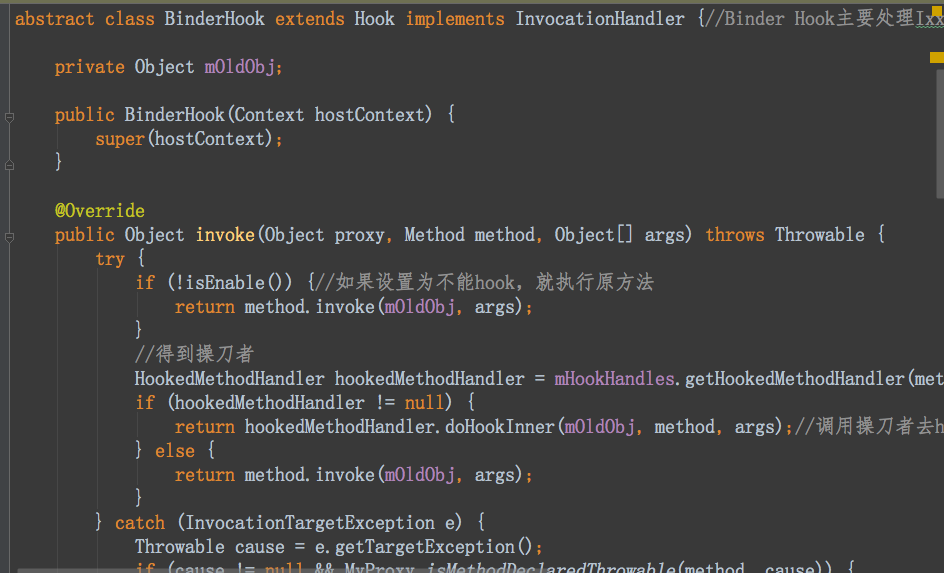


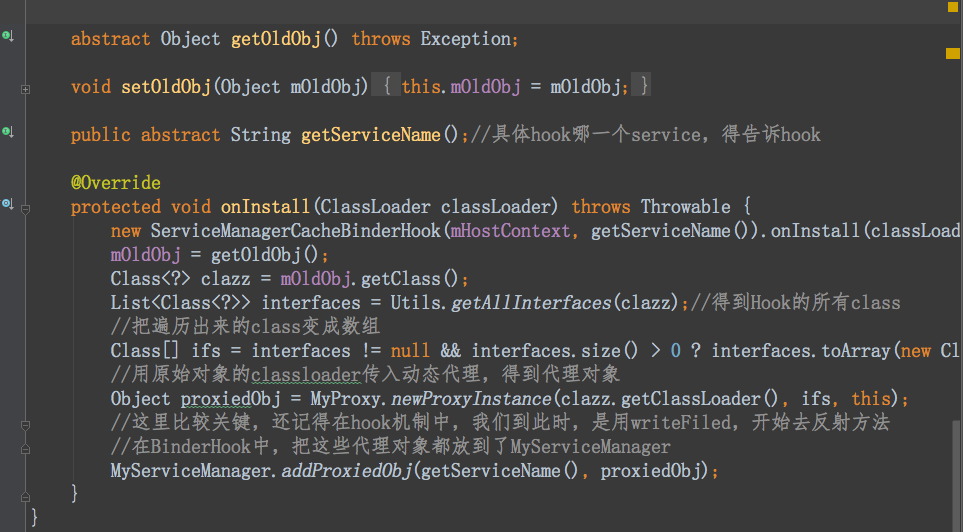
   
前面把service cache存起来，下次如果要真正和service进行通信，通过getOriginService()把原始的service cache拿出来用就行了。

## ServiceManagerBinderHook

   
这个类继承自ProxyHook，主要是用来hook住getService()和checkService()这两个API。如果这两个API被调用，并且在mProxiedObjCache发现有对应的代理对象，则直接返回这个代理对象。

## BinderHook



  
先调用ServiceManagerCacheBinderHook的onInstall()方法更新一下service cache，然后生成一个新的代理对象放到mProxiedObjCache里。这样下次不管是从cache里取，还是直接通过binder调用，就都会返回我们的代理对象。

Binder代理其实在android 系统中也是一个十分重要的角色。

这部分可以从网上搜下相关资料。360 plugin 的Binder代理就是借鉴了系统的Binder相关