**NLP开发维护**

1. **目录**
   1. NLP的用途以及手机厂商如何将NLP预置到手机中
   2. NLP工程目录简介
   3. NLP工程源代码情景分析（含NLP与SDK以及与百度开放平台的关系）
   4. 本地如何调试NLP
   5. Framework层 NLP相关Android 源代码情景分析
      1. Android平台上进程间通信（IPC）机制binder浅析
      2. LocationManagerService如何初始化 NLP
      3. 定位请求如何经过Framework层传递给NLP（留做作业）
      4. NLP内部获取定位结果后如何抛给Framework层（留做作业）

1.6 NLP经典问题总结

**2. NLP的用途以及手机厂商如何将NLP预置到手机中**

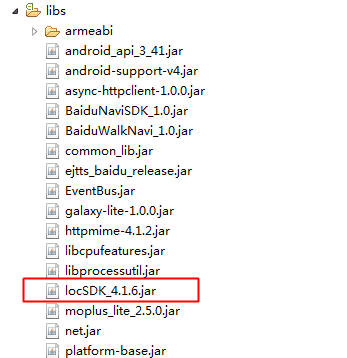
NLP是Network Location Provider的简称，顾名思义它可以提供网络定位服务，请注意GPS定位的实现是涉及到GPS硬件模块儿，如果使用GPS定位的话只能通过系统api调用，因为你别无选择，这里我们只讨论网络定位功能，说道这里那NLP到底有什么特别之处呢？

**看下面的一个问题：如果你开发的项目中需要用到网络定位功能，你会怎么做？？？**

方案一：使用Android 原生api提供的位置服务

|  |
| --- |
| **LocationManager** mLocationManager = (**LocationManager**) getSystemService(Context.LOCATION\_SERVICE);  **LocationListener** locationListener = new LocationListener() {  public void onLocationChanged(Location location) {  if (location != null) {  double longitude = location.getLongitude();//纬度  double latitude = location.getLatitude();//经度  double accuracy = location.getAccuracy();//精度  }  } **使用原生api获取位置**  }  locationManager.requestLocationUpdates(**LocationManager.NETWORK\_PROVIDER**, 45000, 0,locationListener); |

方案二：集成第三方提供的SDK，如从百度LBS开发平台获取定位SDK

**Baidu地图中集成定位定位SDK**

**使用方案二基本上是开发者必选的途径，因为定位功能的实现更可靠，可控性相对较强，开发者之所以不会选择方案一是因为Android系统中原生的网络定位功能使用的是Google的服务，但是无奈它受限于天朝伟大的”墙”，既然系统的网络定位api一般开发者是不会用的，为什么厂商还要预装预装一个NLP在系统中呢，答案很简单，厂商自己要用，比如天气，短信，相机，都可能会用到定位功能，每个应用中都集成一个定位SDK显然不如使用同一个服务。**

**既然Google的服务不可用，那么手机厂商就会用一个可用的服务来替换它，so 我们的NLP就应运而生了，NLP是一个apk，并且作为系统级应用预装到手机中，不同的是它是一个服务提供者，它没有用户操作界面，从始至终它都默默无闻的工作在用户无法感知的后台。**

**NLP作为一个系统级的服务，它的生命周期完全由系统控制，设备启动时Android os会初始化many many服务，其中就包括LocationManagerService 而LocationManagerService会又会初始化更为具体的位置服务提供者 GPSLocation、 NetworkLocation、FusedLocation 我们的NLP就是作为NetworkLocation的具体提供者开始初始化，然后等待着通过Android 位置相关原生api传递过来的定位请求。**

**LocationManagerService 如何能够控制我们的NLP进行初始化呢，其实很简单就是通过bindService（）**

**方法来发出一个绑定action，看下面的两个截图**

**LocationManagerService源码截图：**

|  |
| --- |
|  |

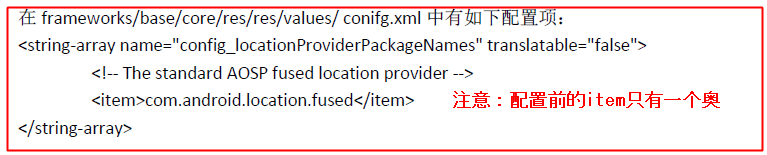
**NLP入口Service的截图：**

|  |
| --- |
|  |

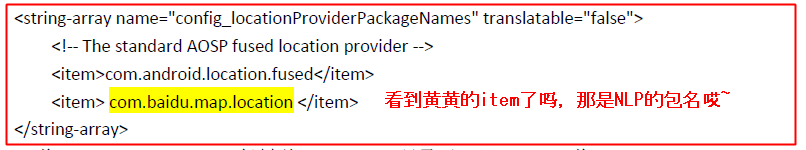
**所以我们只要响应这个绑定action然后进行初始化就ok了,这样NLP就能跟随Android os的启动了，初始化过程中包括定位SDK的初始化，以及Geocoder的初始化。一切搞定之后NLP就能等待定位请求并作出响应了**

**我们的NLP可以接受系统的绑定，是需要厂商在编译自己的系统时进行必要的配置的，我们以Android4.4系上的配置将讲解，当然AndroidL上跟4.4是一样的，看下面的两个截图**

**原生配置文件截图：**



**厂商修改过后的配置文件截图：**



**基本的配置弄好了，NLP的apk文件要放到哪里呢？？？**

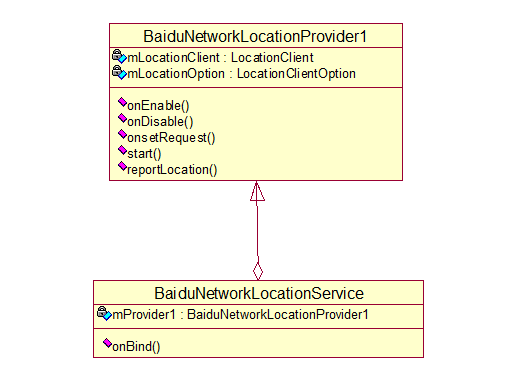
**答案是：system/priv-app 目录下**

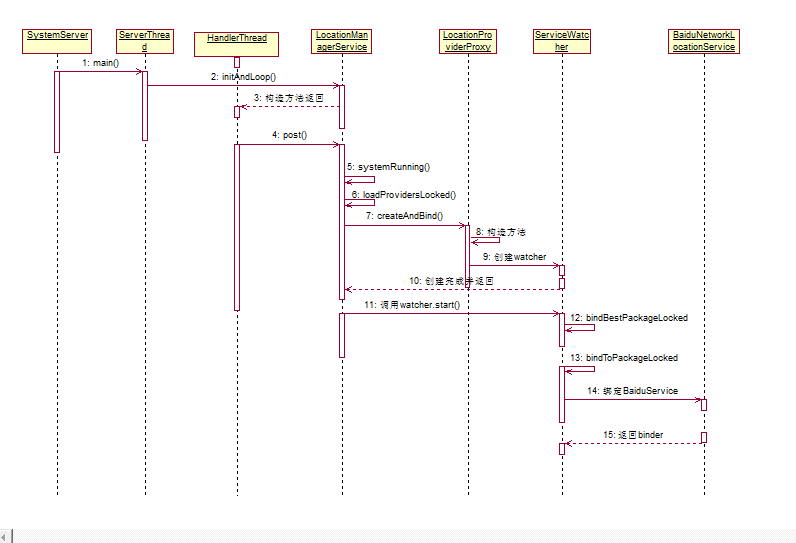
**为什么要放到这里呢？**

**因为在这里我们可以作为系统级应用获取一些普通app无法获取的权限**

1. **NLP工程目录简介**

这里我们直接看eclipse中的工程源码,以定位为例





**NLP的本地调试**

**调试设备要求：**

1进行NLP本地调试的设备必须是已经配置了NLP的包名的设备，也就是上面提到的手机厂商必须在编译系统之前进行的配置

2设备必须有root权限，可以push或者删除系统文件目录下的文件，基本上所有的三星样机都可以用来调试NLP

不同厂商的设备可能有些调试指令不同，这个可直接向厂商接口RD询问

**调试步骤：**

1 打包NLP（使用ant进行编译）生成networklocation.apk

2 将设备连接pc输入如下指令（使用三星设备）

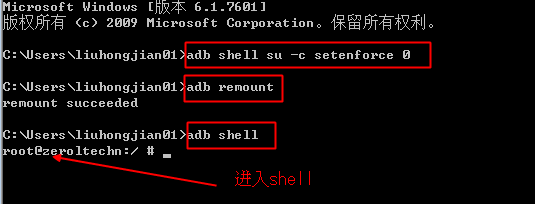
**adb shell su –c setenforce 0 (回车)**

**adb remount (回车)**

**adb shell (回车后就可以操作设备文件目录中的文件了)**

**cd system/priv-app (进入到priv-app目录)**

**rm networklocation-release.apk （删除已有的apk）**



另外NLP中用到了两个so文件，这两个so是放到**system/lib** 目录下的，在Android L 64bit os上可能要放到**system/lib64**下，所以如果so有改动的话需要进入相应的目录删除已有so，具体删除方式跟上面删除apk是相同的，只不过目录不同罢了，如果没有改动没必要执行此操作

**exit （退出shell）**

**adb push 新打包的apk路径 system/priv-app**

**如果so有更新执行下面操作**

**adb push 新编译的so路径 system/lib (or system/lib64)**

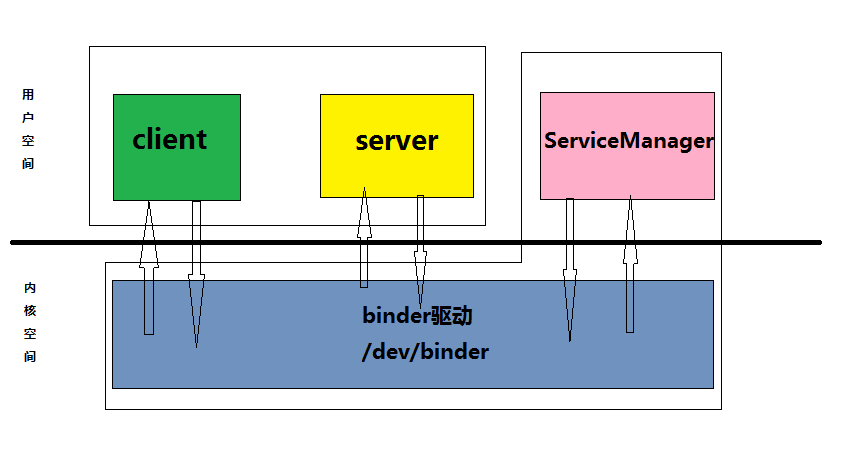
**adb reboot （重启设备）**

**为了测试NLP的基本功能我们自己写了一个简单的DemoNLPTest.apk 运行主界面如下：**

 **在设备重启后就可以测试基本功能了**

1. **Framework层源代码分析**
2. **Binder机制浅析**

**Binder进程间通讯机制实现结构图：**



**1. Client、Server和Service Manager实现在用户空间中，Binder驱动程序实现在内核空间中**

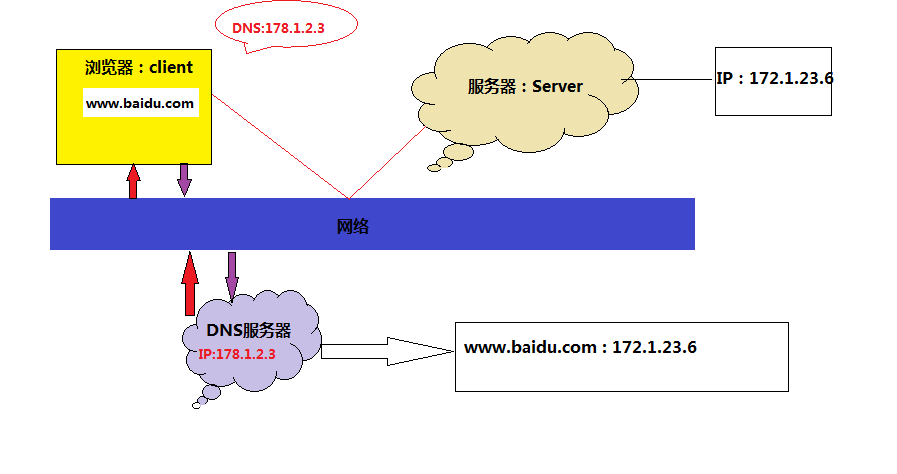
**2. Binder驱动程序和Service Manager在Android平台中已经实现，开发者只需要在用户空间实现自己的Client和Server**

**3. Binder驱动程序提供设备文件/dev/binder与用户空间交互，Client、Server和Service Manager通过open和ioctl文件操作函数与Binder驱动程序进行通信**

**4. Client和Server之间的进程间通信通过Binder驱动程序间接实现**

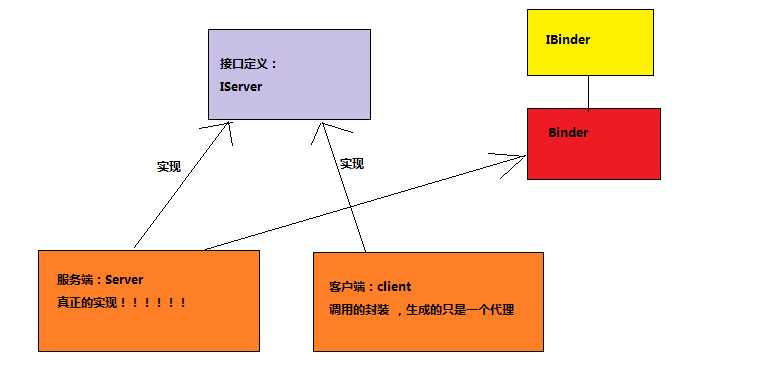
**5. Service Manager是一个守护进程，用来管理Server，并向Client提供查询Server接口的能力**

**通过一个通俗的例子来理解binder IPC 机制中的各个组件之间的关系，看下图：**

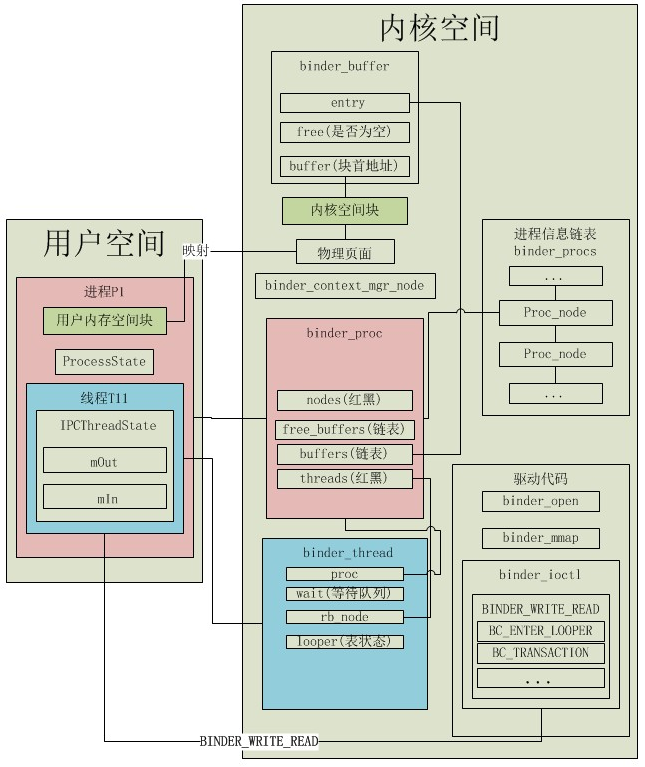


**什么是binder实体：通过binder机制进行进程间通信的一个进程**

**通过面向对象的语言表述：**

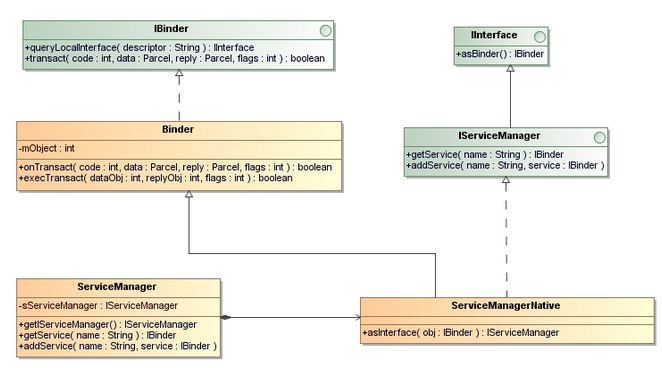


**Binder机制中内存中的相关模型：**



1. **aidl是如何通过binder机制完成进程间通讯的**

**下面是ServiceManager在framework层的java实现**



frameworks/base/core/java/android/os/ServiceManager.java

frameworks\base\core\java\android\os\ServiceManagerNative

frameworks/base/core/java/com/android/internal/os/BinderInternal.java

**AIDL举例：通过aidl实现远程绑定服务**

**6 经典问题梳理**

**1.高通平台上因修改了Android源码导致的NLP无法成功初始化问题处理：**

|  |
| --- |
| The first method involves Nlp Proxy (Combo Provider), which is the middle man between Location Manger Service and Baidu NLP.   Location Manager Service will choose NlpProxy (com.qualcomm.location) as the NLP (it is highest version) and NlpProxy will call directly into Baidu NLP.   Location Manager Service -> NlP Proxy -> Baidu NLP   The only change that should be needed for this option is in vendor\qcom\proprietary\gps\framework\java\com\qualcomm\location\nlp\NlpProxyProvider.java:  private static final String[] GNP\_PACKAGE\_NAMES =  {“com.baidu.map.location”  "com.google.android.gms",  "com.google.android.location"};  private static final String[] GNP\_ACTION\_NAMES =  {"com.android.location.service.v3.NetworkLocationProvider",  "com.android.location.service.v3.NetworkLocationProvider",  "com.android.location.service.v2.NetworkLocationProvider"};    NlpProxy will try to bind to com.baidu.map.location first, since it is first in the GNP\_PACKAGE\_NAMES list.   You can also remove google’s packages in the list if you don't want them as backup in case Baidu isn’t there and google packages are there.  (GNP can be used if there is no Baidu package so it can be used device which is targeting other carrier and using GNP - control through include |

**2.NLP定位结果无法返回处理**

|  |
| --- |
| 1、保证加载的provider的包名中有百度的包com.baidu.map.location。   2、在获取comboNlpLocationProvier的包名，前需要判断一下之前加载的provider的包名中是否有百度的包名com.baidu.map.location，如果有就不要再加载comboNlpLocationProvier的包名了，把保  存comboNlpLocationProvier包名的变量置成""，后面再上报位置时就不会因为passiveProvider中有保存高通comboNlpLocationProvier的包名而禁止上报位置了。 |