获取第三方应用调用系统具体的服务和方法

第三方应用调用系统服务的方法是通过ipc进行的

ipc机制通过Binder来实现跨进程通信

通过AIDL文件生成的接口：以ITelephony为例

```
public interface ITelephony extends android.os.IInterface

{
/** Local-side IPC implementation stub class. */
public static abstract class Stub extends android.os.Binder implements
com.android.internal.telephony.ITelephony
{
private static final java.lang.String DESCRIPTOR =
"com.android.internal.telephony.ITelephony";
/** Construct the stub at attach it to the interface. */
public Stub()
{
this.attachInterface(this, DESCRIPTOR);
}
```

上图的stub抽象类就是某个服务具体实现的父类，而stub 是Binder的子类

第三方应用调用系统服务的方法都会先将要调用的系统服务方法必要的参数传递给Binder,最终都会执行到Binder类的这个方法

```
private boolean execTransact(int code, long dataObj, long replyObj,
        int flags) {
```

这个方法最终又会去调用stub重写Binder类的一个方法onTransaction（xxxx）

```
@Override public boolean onTransact(int code, android.os.Parcel data, android.os.Parcel reply, int flags) throws android.os
```

这个方法中传过来的参数int code,根据不同的code,决定调用跨进程类中不同的方法。

基于这个通过在Binder类中的execTransact（）中的根据这个code值，通过反射拿到stub类中定义的所有的属性值，然后比较，获取属性值对于的属性名

属性值如下：属性的名中就包含了调用的具体方法名

```
static final int TRANSACTION_call = (android.os.IBinder.FIRST_CALL_TRANSACTION + 1);
static final int TRANSACTION_endCall = (android.os.IBinder.FIRST_CALL_TRANSACTION + 2);
static final int TRANSACTION_endCallForSubscriber = (android.os.IBinder.FIRST_CALL_TRANSACTION + 3);
```