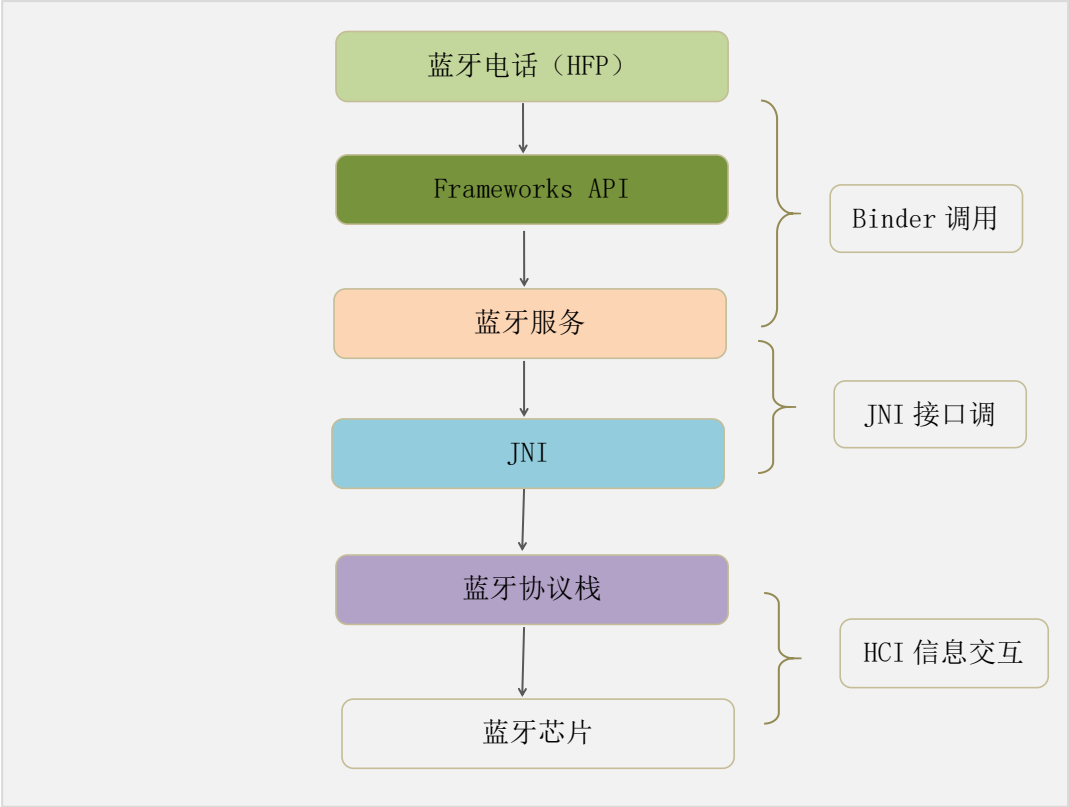
# 蓝牙电话交互流程解析

## 一、前言

蓝牙电话是 Android 系统里非常重要的一部分，也是车机系统里的重要组成之一，所以熟悉蓝牙电话原理以及相关 API 有助于初学者快速的进入开发状态，本文就谈谈蓝牙的电话的原理与相关 API 代码。

## 二、Android 蓝牙电话框架

先看电话框架流程图，如下图 1.




基于目前我们现在车机的情况，蓝牙协议栈与蓝牙芯片是采取第三方顾凯的芯片，我们集成的是其提供的 SDK，所以这里两部分不需要我们修改，只需要跟顾凯方联调，我们只需要搞明白上面四部分蓝牙电话 APP、Frameworks API、蓝牙服务、JNI 模块即可。这些模块之间是如何交互的，在另一个文档中《蓝牙框架原理以及数据通讯交互流程》已做详细说明，此处，不在做详细解析，此文将着重介绍 HFP 电话 API 以及拨打流程。

## 三、蓝牙电话状态

在蓝牙电话中，一共有以下 7 种状态，需要了解这 7 种状态，才能针对这七种状态做相应的处理：

1、CALL_STATE_ACTIVE : 0 通话中状态

2、CALL_STATE_HELD : 1 电话保持中

3、CALL_STATE_DIALING : 2 正在拨号中

4、CALL_STATE_ALERTING : 3 正在响铃中

5、CALL_STATE_INCOMING : 4 正在来电中

6、CALL_STATE_WAITING : 5 正在等待中

7、CALL_STATE_HELD_BY_RESPONSE_AND_HOLD : 6 已响应并保持中

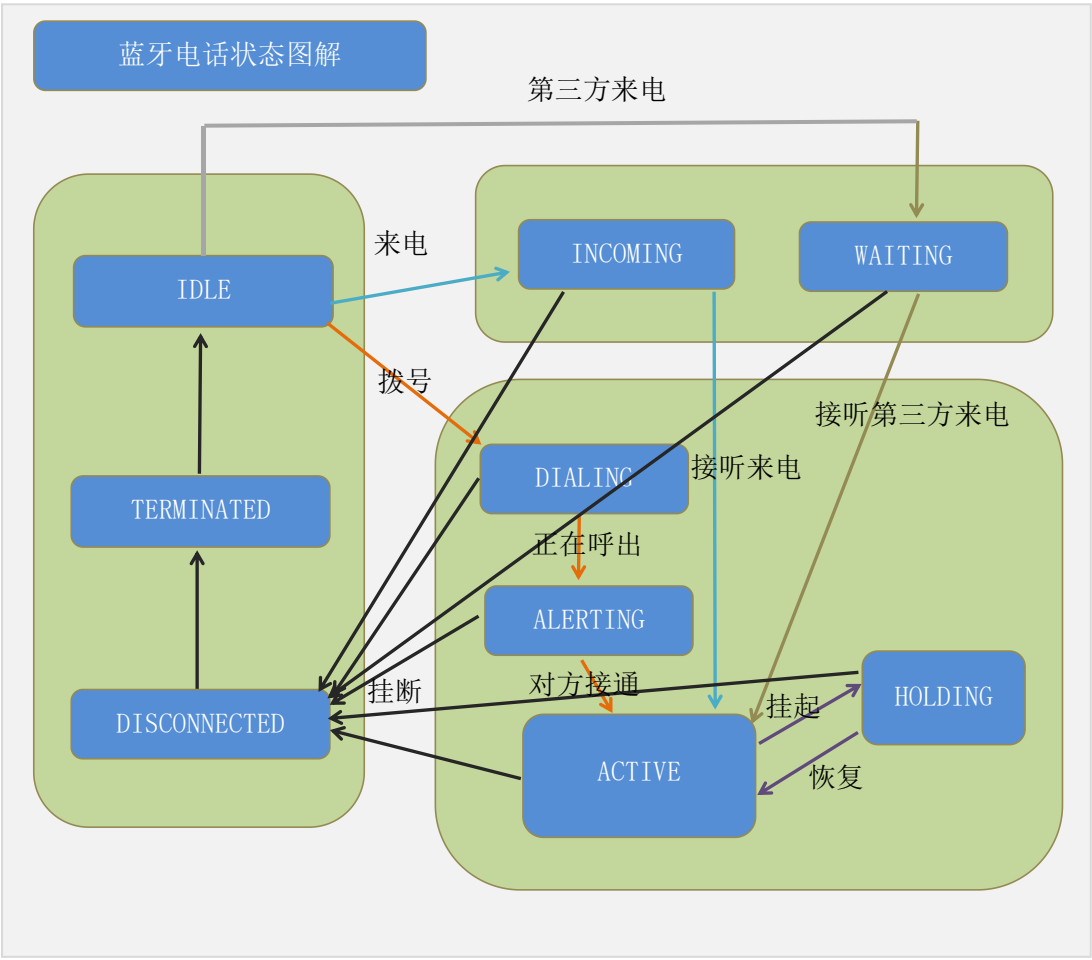
8、CALL_STATE_TERMINATED : 7 挂断状态

结合我们现有项目以及源码分析，我们可以简化以上状态处理：

1、“响铃”状态其实是在拨号过程中，对方已收到并处于响铃返回的状态，但目前是没有的，只要是对方没有接听，则一直处于“拨号中”，是没有收到对方的响铃状态回复的，所以可以不处理此状态，但为了预防万一，此状态可以合并拨号中状态做一样的处理；

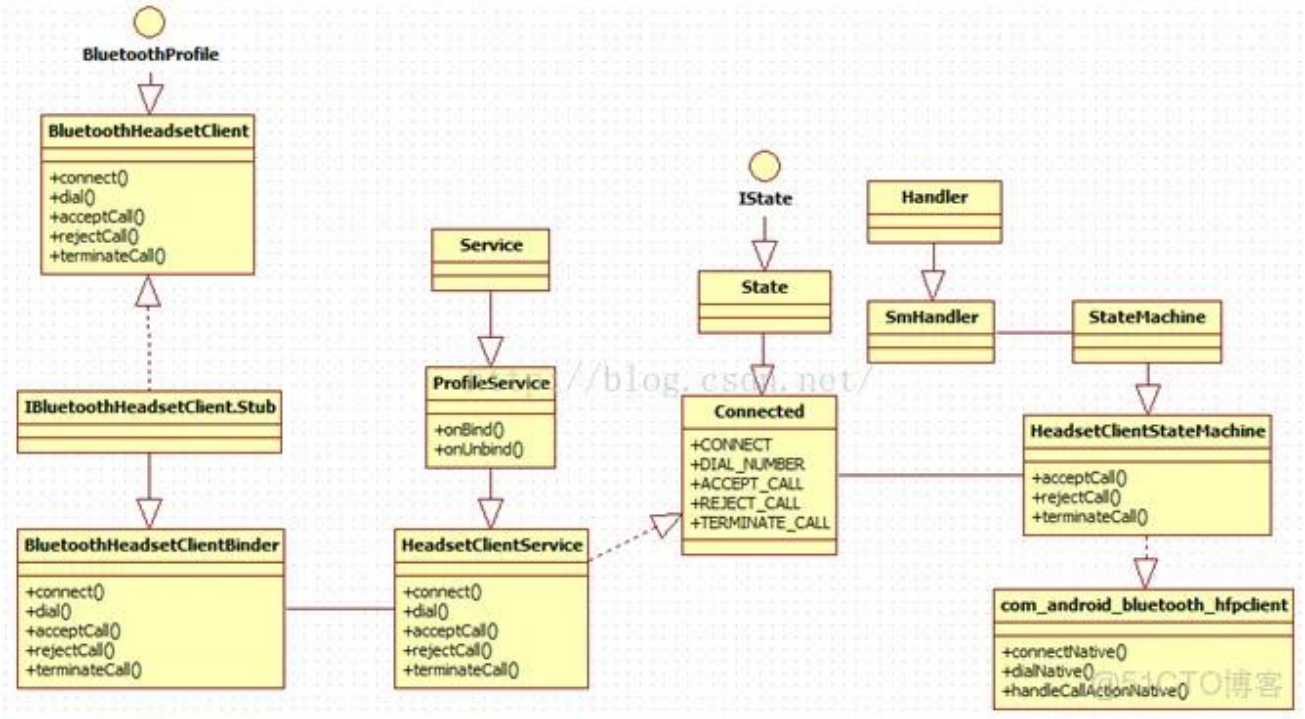
2、同样“响应并保持中”状态，也是一样，目前也是没有此状态的，目前只有保持中状态，所以，此状态也是合并“保持中”一样处理；

3、如下为蓝牙电话状态变化图解：



## 四、蓝牙电话核心类图

了解电话的状态类型以及其状态变化顺序，下来我们了解下开发过程中核心类图，如下图：




1、BluetoothHeadsetClient 是一个 api,由第三方 apk 直接调用,可以进行拨号/接听/拒接/挂断操作,对应的方法依次为 dial()/acceptCall()/rejectCall()/terminateCall()

2、HeadsetClientService 是一个服务，BluetoothHeadsetClientBinder 是它的内部类，BluetoothHeadsetClientBinder 是 BluetoothHeadsetClient 在 HeadsetClientService 中的代理,通过 aidl 进行方法的调用。

3、Connected(已连接状态)是 HeadsetClientStateMachine 的其中一种状态,通话的相关操作都建立在已连接状态之上,其它三种状态为 Disconnected, Connecting,AudioOn 状态。HeadsetClientService 根据不同的方法给状态机发送不同的消息，最后通过 HeadsetClientStateMachine 根据 JNI 机制调用 com_android_bluetooth_hfpclient.cpp 对应的方法,最后调用底层 C/C++ 来真正的实现拨号/接听/拒接/挂断操作。

4、拨号/接听/拒接/挂断方法调用的流程完全是一模一样的，下面结合我们项目中代码已拨号为例详细说明调用流程；

## 五、蓝牙电话代码调用流程(拨号)

1、上层应用 YX_Bluetooth 通过中间件 android.bluetooth.BluetoothHeadsetClient api 进行调用，如下图：

```java
/**
 * 拨号
 */
public void dial(String phoneNumber) {
    YXLog.i(TAG, msg: "dial phoneNumber = " + phoneNumber);

    if (phoneNumber == null) {...}

    Pattern mPattern = Pattern.compile("^[+-]?[0-9]*");
    Matcher mMatcher = mPattern.matcher(phoneNumber);

    if (TextUtils.isEmpty(phoneNumber) || phoneNumber.equals(Constant.PHONE_NUMBER_ERROR)) {...}

    if (!(mMatcher.matches())) {...}

    BluetoothDevice connectBluetoothDevice = mBluetoothConnectListenerClient.getConnectedPhoneDevice();
    if (mBluetoothHeadsetClient == null || connectBluetoothDevice == null) {...}
    boolean isHfpConnect = connectHfp(connectBluetoothDevice);
    if (!isHfpConnect) {...}

    List<BluetoothHeadsetClientCall> calls = getCurrentCalls();
    if (calls != null && calls.size() >= 1) {...}

    if (ActivityManager.isUserAMonkey()) {...}

    YXLog.i(TAG, msg: "start dial phoneNumber = " + phoneNumber);
    mBluetoothHeadsetClient.dial(connectBluetoothDevice, phoneNumber);
}
```

Frameworks api 掉用dial()

2、中间件 Frameworks 再以 Binder 调用对接 Bluetooth 服务接口：

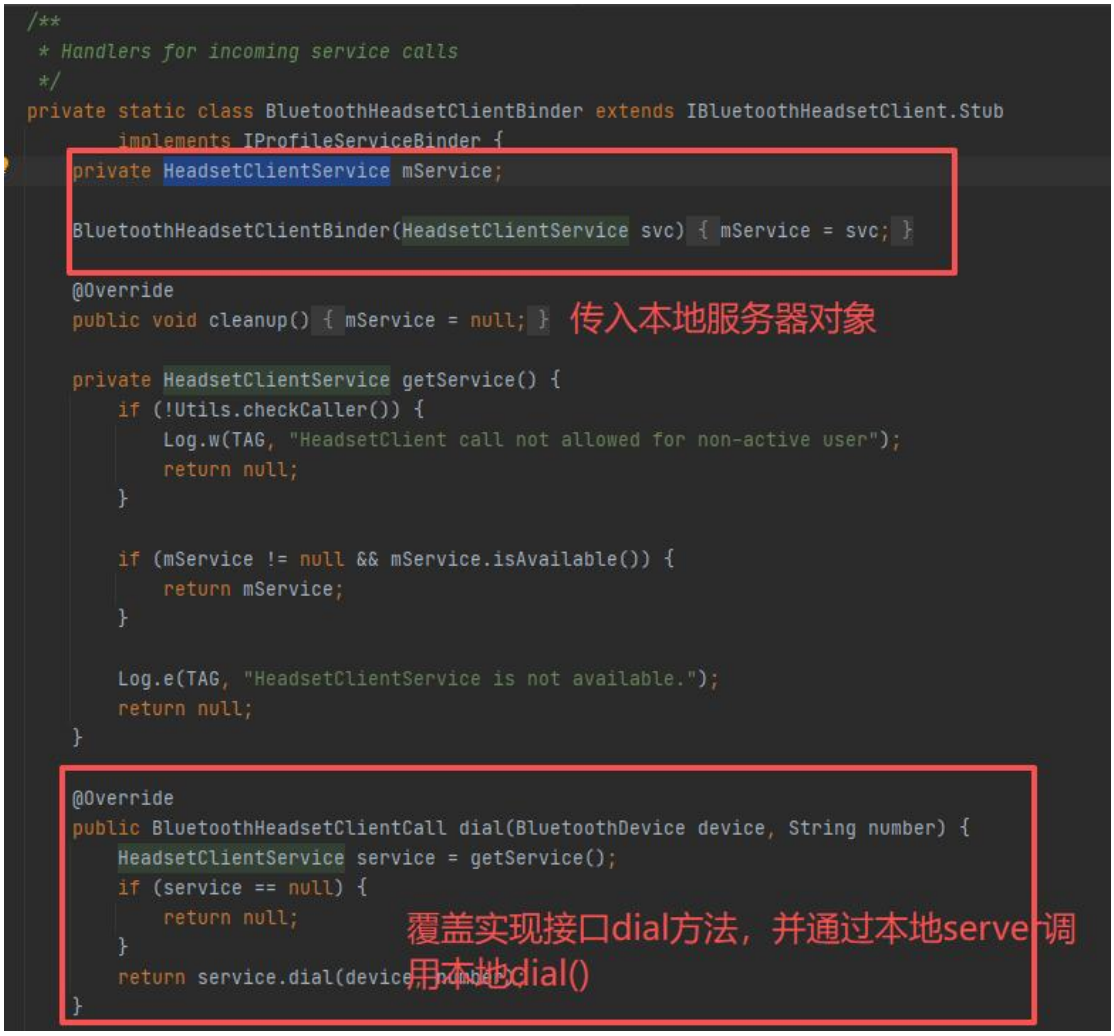
BluetoothHeadsetClient.class

Decompiled .class file, bytecode version: 53.0 (Java 9)

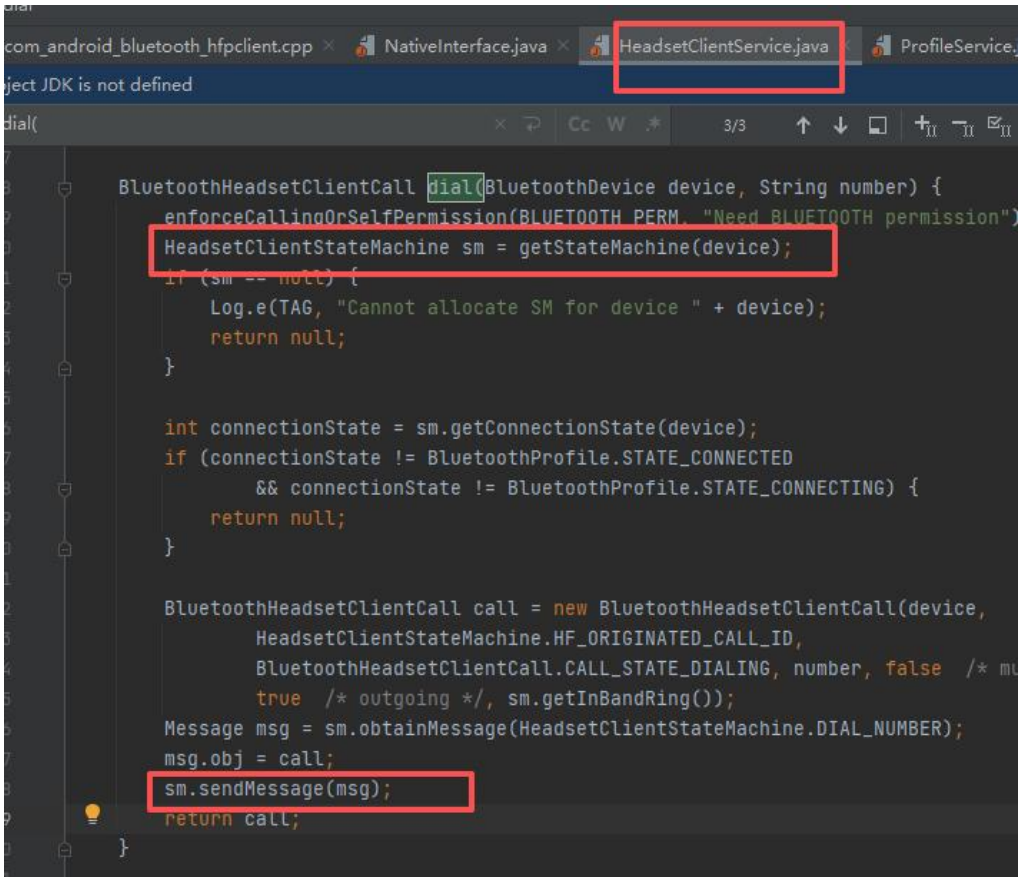
```java
479     public BluetoothHeadsetClientCall dial(BluetoothDevice device, String number) {
480         log( msg: "dial()");
481         IBluetoothHeadsetClient service = this.getService();
482         if (service != null && this.isEnabled() && isValidDevice(device)) {
483             try {
484                 return service.dial(device, number);
485             } catch (RemoteException var5) {
486                 Log.e( tag: "BluetoothHeadsetClient", Log.getStackTraceString(new Throwable()));
487             }
488         }
489
490         if (service == null) {
491             Log.w( tag: "BluetoothHeadsetClient", msg: "Proxy not attached to service");
492         }
493
494         return null;
495     }
```

Binder 调用dial()

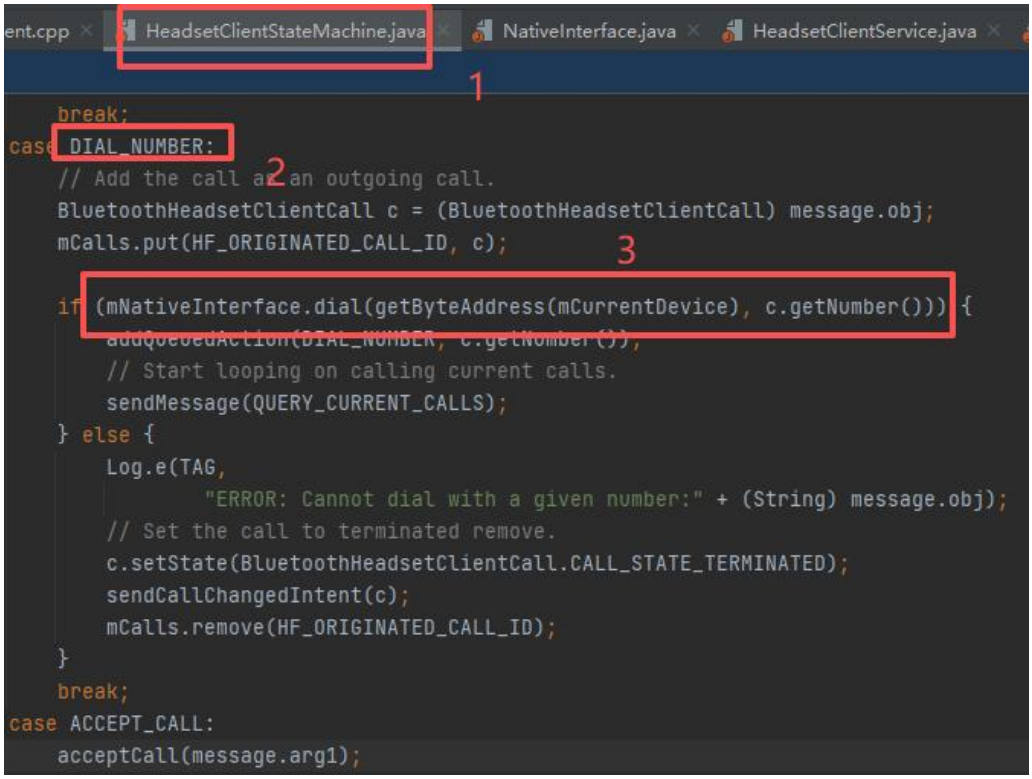
3、蓝牙服务中 BluetoothHeadsetClientBinder 继承 IBluetoothHeadsetClient.Stub，覆盖对应的方法 dial()，再本地服务 HeadsetClientService 绑定，通过 HeadsetClientService 调用本地 dial 方法，如下图：




4、通过本地服务 HeadsetClientService 生成一个 BluetoothHeadsetClientCall 对象，并发送到业务类 HeadsetClientStateMachine 进行处理，此服务如一个中转器，只执行对事件的接受与分发，如下图：

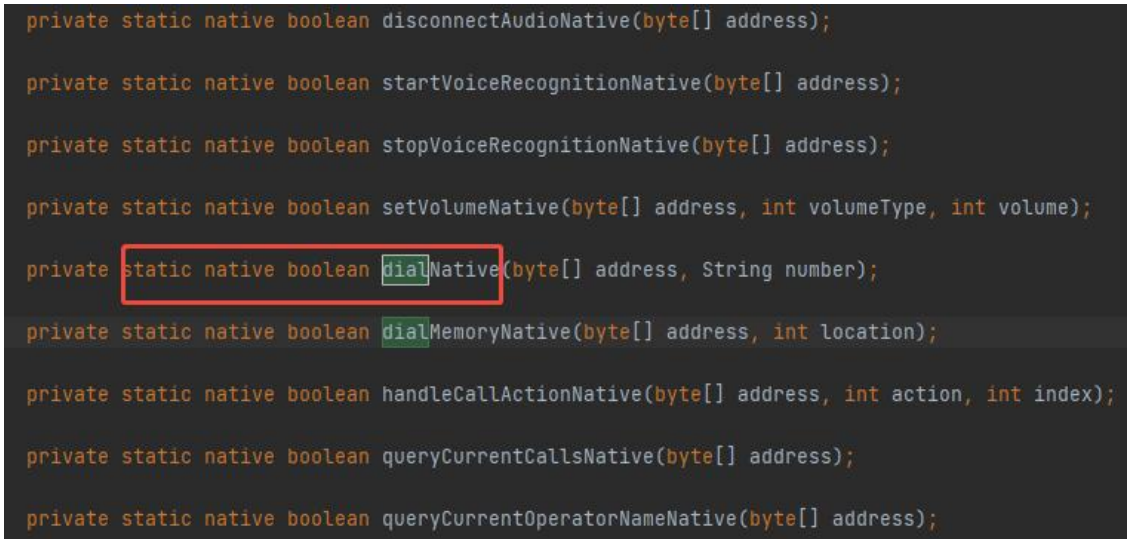



5、前面已经说过，蓝牙电话相关事件都在 Connected State 对象中处理，当接受到 Server 发送过来的拨打事件后，会在 Connect 对像中处理，通过 Event.type 类型判断，调用与底层交互的本地接口类 NativeInterface，如下图：

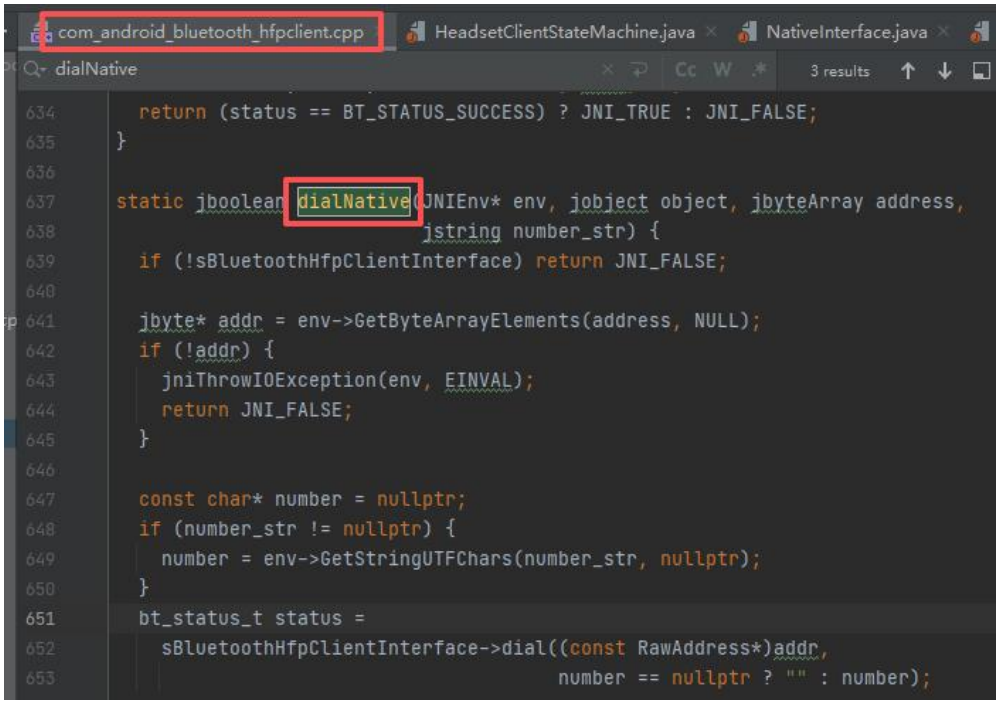


6、NativeInterface 类是与蓝牙协议栈对接的 Java 本地接口类，通过 native 方法与协议栈方法进行对接，如下图：




7、最后调到蓝牙协议库的接口,如下图：



8、再往下，就调到第三方芯片商（顾凯）的 SDK 里面了，这里不做详述。

9、最后附上代码调用流程图，如下。

## 六、蓝牙电话代码调用流程图(拨号)

