# TODO

# Wificond

具体分析：https://blog.csdn.net/sinat\_20059415/article/details/80945447

system/connectivity/wificond/

前言：之前在[（五十） Android O WiFi的扫描流程梳理](https://blog.csdn.net/sinat_20059415/article/details/80784129#t12) 中梳理到wificond，就是找不到wificond对应的具体实现类

mWificond = mWifiInjector.makeWificond();找不到对应的java实现

IBinder binder = ServiceManager.getService(WIFICOND\_SERVICE\_NAME)

bullhead:/ $ service list | grep wificond

128 wificond: []

服务端是由cpp实现的，这种aidl实现方式被Google成为aidl-cpp

clientInterface = mWificond.createClientInterface()

IWifiScannerImpl也是类似实现

mWificondScanner = mClientInterface.getWifiScannerImpl();

**aidl存放路径**

./system/connectivity/wificond/aidl/android/net/wifi/IWificond.aidl

编译之后

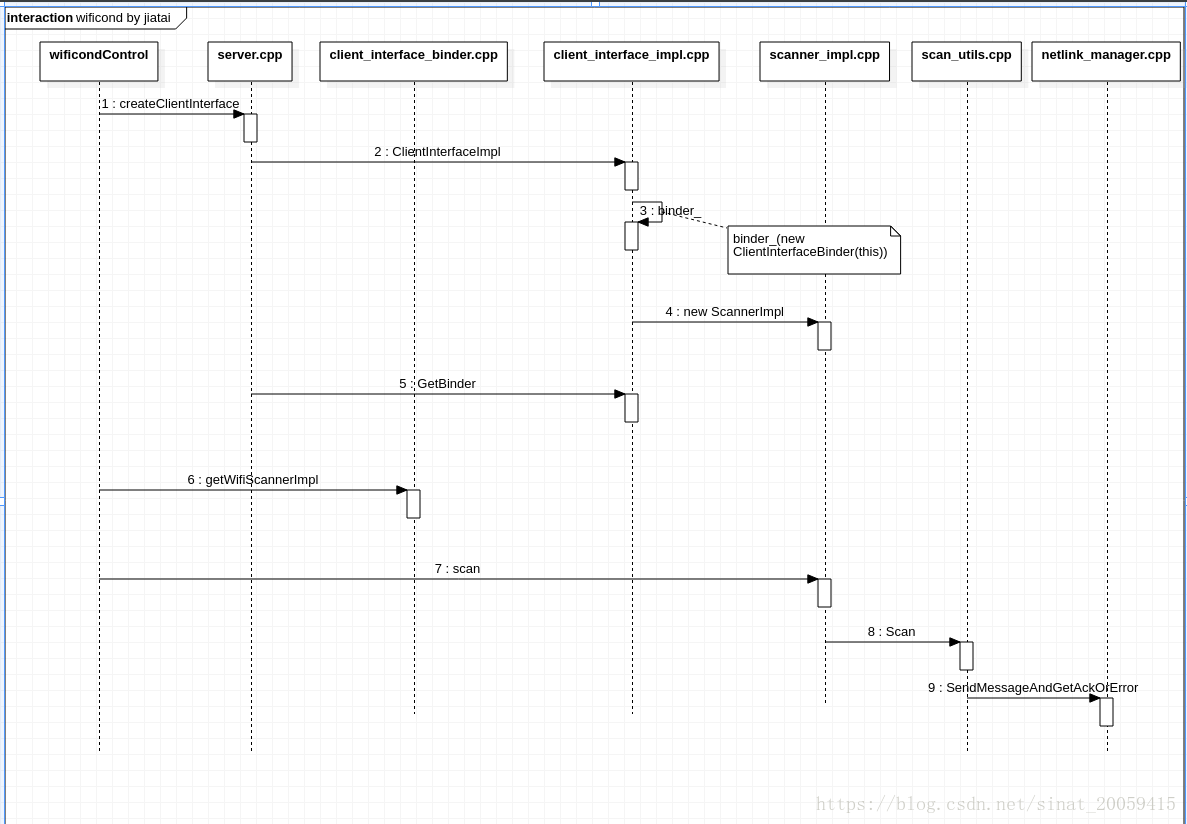
aidl对应的.h文件存放路径：

aidl对应的.cpp文件存放路径：

/out/target/product/xxxxxx/obj/STATIC\_LIBRARIES/libwificond\_ipc\_intermediates/aidl-generated/src/aidl/android/net/wifi/IWificond.cpp

可以通过“make wificond”编译出如下的执行文件，在system/bin下

## 框架



## main

system/connectivity/wificond/main.cpp

**int** main(**int** argc, **char**\*\* argv) {  
 android::base::InitLogging(argv, android::base::LogdLogger(android::base::SYSTEM));  
 LOG(INFO) << **"wificond is starting up..."**;  
  
 unique\_ptr<android::wificond::LooperBackedEventLoop> event\_dispatcher(  
 **new** android::wificond::LooperBackedEventLoop());  
 ScopedSignalHandler scoped\_signal\_handler(event\_dispatcher.get());  
  
 **int** binder\_fd = SetupBinderOrCrash();  
 CHECK(event\_dispatcher->WatchFileDescriptor(  
 binder\_fd,  
 android::wificond::EventLoop::kModeInput,  
 &OnBinderReadReady)) << **"Failed to watch binder FD"**;  
  
 android::wificond::NetlinkManager netlink\_manager(event\_dispatcher.get());  
 CHECK(netlink\_manager.Start()) << **"Failed to start netlink manager"**;  
 android::wificond::NetlinkUtils netlink\_utils(&netlink\_manager);  
 android::wificond::ScanUtils scan\_utils(&netlink\_manager);  
  
 unique\_ptr<android::wificond::Server> server(**new** android::wificond::Server(  
 unique\_ptr<InterfaceTool>(**new** InterfaceTool),  
 unique\_ptr<SupplicantManager>(**new** SupplicantManager()),  
 unique\_ptr<HostapdManager>(**new** HostapdManager()),  
 &netlink\_utils,  
 &scan\_utils));  
 server->CleanUpSystemState();  
 RegisterServiceOrCrash(server.get());  
  
 event\_dispatcher->Poll();  
 LOG(INFO) << **"wificond is about to exit"**;  
 **return** 0;  
}

### RegisterServiceOrCrash

**void** RegisterServiceOrCrash(**const** android::sp<android::IBinder>& service) {  
 android::sp<android::IServiceManager> sm = android::defaultServiceManager();  
 CHECK\_EQ(sm != NULL, **true**) << **"Could not obtain IServiceManager"**;  
  
 CHECK\_EQ(sm->addService(android::String16(kServiceName), service),  
 android::NO\_ERROR);  
}

system/connectivity/wificond/ipc\_constants.cpp

**namespace** android {  
**namespace** wificond {  
**namespace** ipc\_constants {  
  
**const char** kServiceName[] = **"wificond"**;  
  
} *// namespace ipc\_constants*} *// namespace wificond*} *// namespace android*

**using** android::wificond::ipc\_constants::kServiceName;

## Server

### Server（）

Server::Server(unique\_ptr<InterfaceTool> if\_tool,  
 unique\_ptr<SupplicantManager> supplicant\_manager,  
 unique\_ptr<HostapdManager> hostapd\_manager,  
 NetlinkUtils\* netlink\_utils,  
 ScanUtils\* scan\_utils)  
 : if\_tool\_(std::move(if\_tool)),  
 supplicant\_manager\_(std::move(supplicant\_manager)),  
 hostapd\_manager\_(std::move(hostapd\_manager)),  
 netlink\_utils\_(netlink\_utils),  
 scan\_utils\_(scan\_utils) {  
}

## StartHostapd时序

**bool** ApInterfaceImpl::StartHostapd() {  
 **return** hostapd\_manager\_->StartHostapd();  
}

## ApInterfaceImpl

ApInterfaceImpl::ApInterfaceImpl(**const** string& interface\_name,  
 uint32\_t interface\_index,  
 NetlinkUtils\* netlink\_utils,  
 InterfaceTool\* if\_tool,  
 HostapdManager\* hostapd\_manager)

# REF

Android O WiFi的扫描流程梳理续——梳理java与c++之间的aidl-cpp通信

https://blog.csdn.net/sinat\_20059415/article/details/80945447