Android 9.0 BT 打开流程分析

Date:2020-07-21

Version:A

Auth:陶 冬

Revision History	Date	Description
A	2020-07-21	Initial Draft

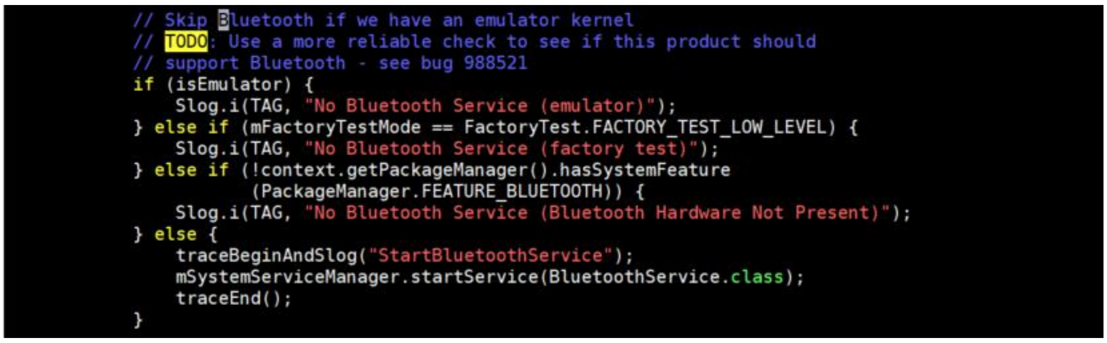
Note:

本文档对 Android 9.0 BT 开启流程进行分析，供参考。

目录

1. 系统启动

1.1 代码实现



```
// Skip Bluetooth if we have an emulator kernel
// TODO: Use a more reliable check to see if this product should
// support Bluetooth - see bug 988521
if (isEmulator) {
    Slog.i(TAG, "No Bluetooth Service (emulator)");
} else if (mFactoryTestMode == FactoryTest.FACTORY_TEST_LOW_LEVEL) {
    Slog.i(TAG, "No Bluetooth Service (factory test)");
} else if (!context.getPackageManager().hasSystemFeature
           (PackageManager.FEATURE_BLUETOOTH)) {
    Slog.i(TAG, "No Bluetooth Service (Bluetooth Hardware Not Present)");
} else {
    traceBeginAndSlog("StartBluetoothService");
    mSystemServiceManager.startService(BluetoothService.class);
    traceEnd();
}
```

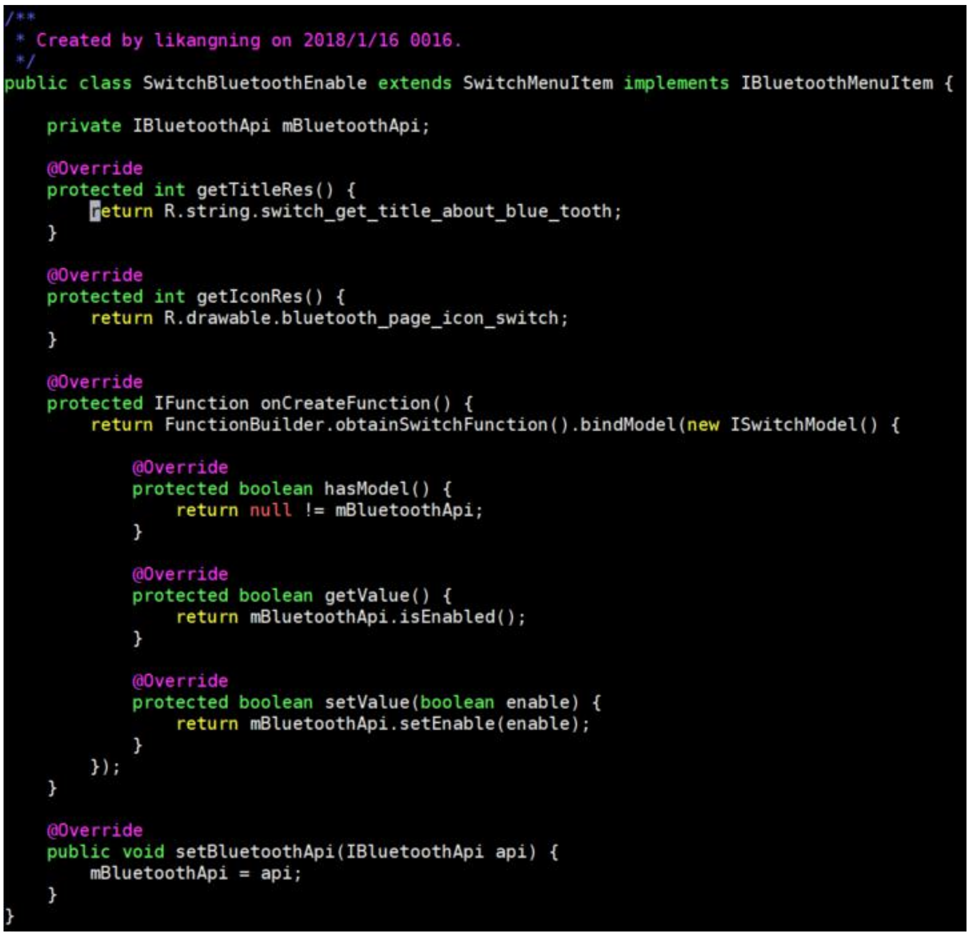
1.2 Log 打印

```
07-15 14:07:45.536   3368   3368 I SystemServer: StartBluetoothService

07-15 14:07:45.536   3368   3368 I SystemServiceManager: Starting com.android.server.BluetoothService

07-15 14:07:45.554   3368   3368 E BluetoothManagerService: td:BluetoothManagerService.java

BluetoothManagerService enter.

07-15 14:07:45.555   3368   3368 D BluetoothManagerService: Loading stored name and address

07-15 14:07:45.556   3368   3368 D BluetoothManagerService: Stored bluetooth Name=Bluedroid TV

1.0,Address=C0:8A:CD:50:D9:44

07-15 14:07:45.556   3368   3368 D BluetoothManagerService: Bluetooth persisted state: 0

07-15 14:07:45.556   3368   3368 D BluetoothManagerService: bluetoothDongleExist = 1

07-15 14:07:45.556   3368   3368 D BluetoothManagerService: isBluetoothPersistedStateOn = 0

07-15 14:07:45.557   3368   3368 D BluetoothManagerService: Detected SystemUiUid: 10013
```

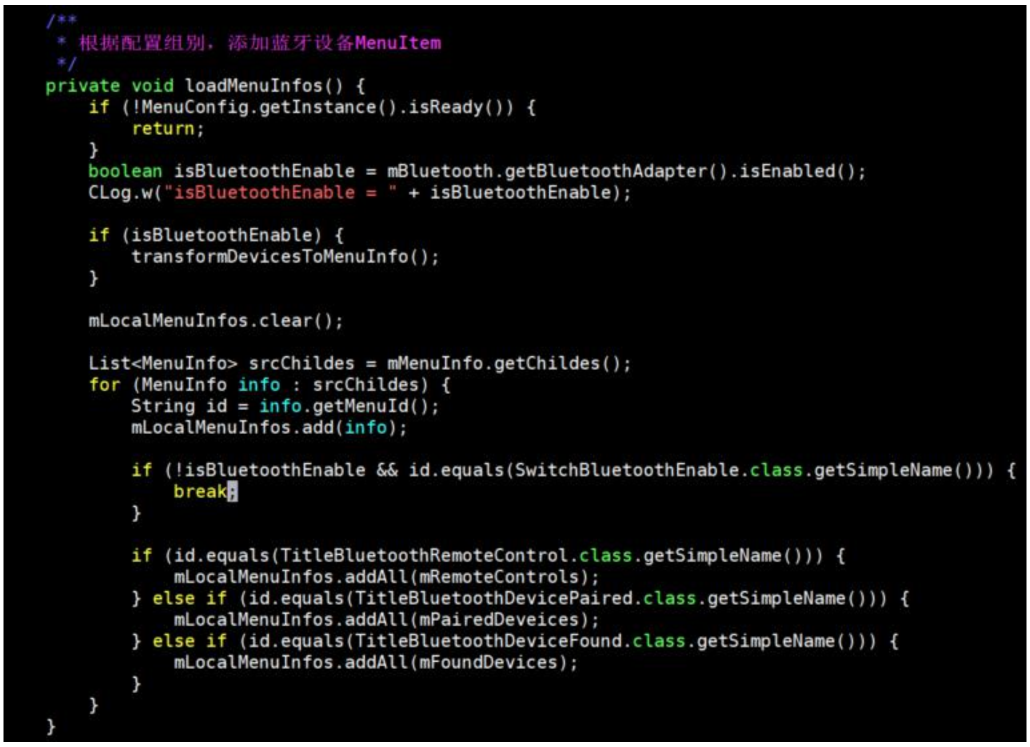
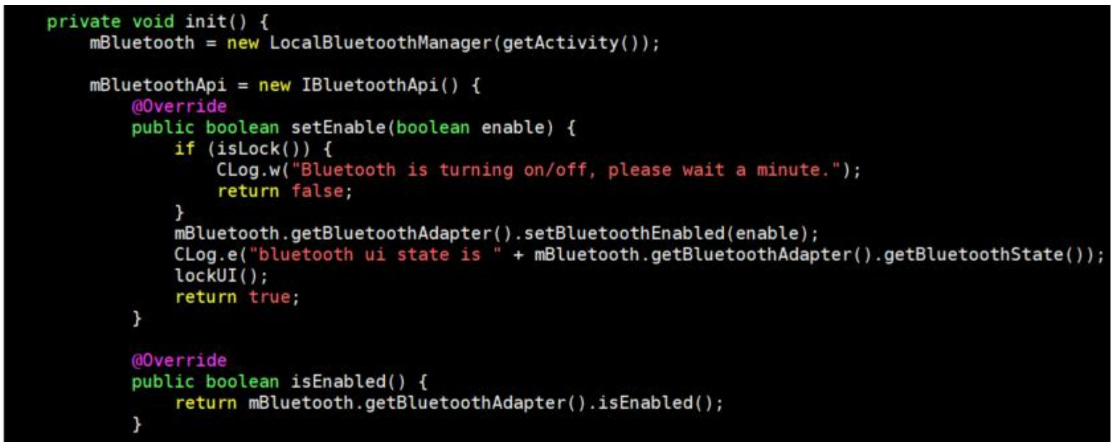
2. Setting 层

2.1 代码实现

app/src/main/java/com/cvte/settings/fragment/BluetoothDevicesFragment.java

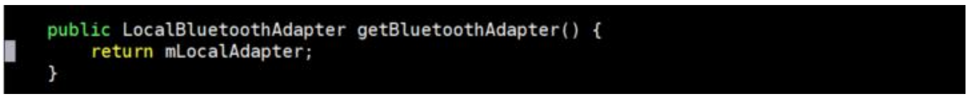
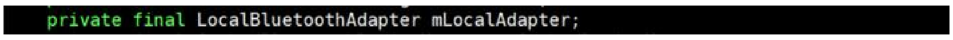


```
/**
 * Created by likangning on 2018/1/16 0016.
 */
public class SwitchBluetoothEnable extends SwitchMenuItem implements IBluetoothMenuItem {

    private IBluetoothApi mBluetoothApi;

    @Override
    protected int getTitleRes() {
        return R.string.switch_get_title_about_blue_tooth;
    }

    @Override
    protected int getIconRes() {
        return R.drawable.bluetooth_page_icon_switch;
    }

    @Override
    protected IFunction onCreateFunction() {
        return FunctionBuilder.obtainSwitchFunction().bindModel(new ISwitchModel() {

            @Override
            protected boolean hasModel() {
                return null != mBluetoothApi;
            }

            @Override
            protected boolean getValue() {
                return mBluetoothApi.isEnabled();
            }

            @Override
            protected boolean setValue(boolean enable) {
                return mBluetoothApi.setEnable(enable);
            }
        });
    }

    @Override
    public void setBluetoothApi(IBluetoothApi api) {
        mBluetoothApi = api;
    }
}
```

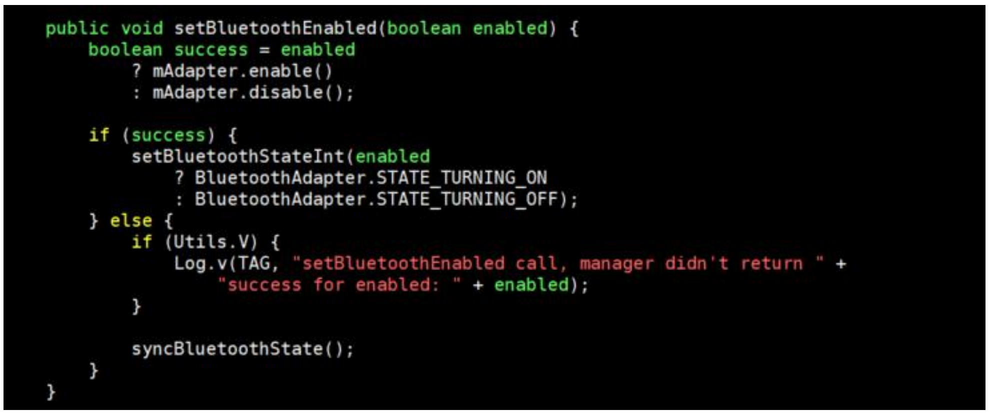

```
/**
 * 根据配置组别，添加蓝牙设备MenuItem
 */
private void loadMenuInfos() {
    if (!MenuConfig.getInstance().isReady()) {
        return;
    }
    boolean isBluetoothEnable = mBluetooth.getBluetoothAdapter().isEnabled();
    CLog.w("isBluetoothEnable = " + isBluetoothEnable);

    if (isBluetoothEnable) {
        transformDevicesToMenuInfo();
    }

    mLocalMenuInfos.clear();

    List<MenuInfo> srcChildes = mMenuInfo.getChildes();
    for (MenuInfo info : srcChildes) {
        String id = info.getMenuId();
        mLocalMenuInfos.add(info);

        if (!isBluetoothEnable && id.equals(SwitchBluetoothEnable.class.getSimpleName())) {
            break;
        }

        if (id.equals(TitleBluetoothRemoteControl.class.getSimpleName())) {
            mLocalMenuInfos.addAll(mRemoteControls);
        } else if (id.equals(TitleBluetoothDevicePaired.class.getSimpleName())) {
            mLocalMenuInfos.addAll(mPairedDeveices);
        } else if (id.equals(TitleBluetoothDeviceFound.class.getSimpleName())) {
            mLocalMenuInfos.addAll(mFoundDevices);
        }
    }
}
```

```
private void init() {
    mBluetooth = new LocalBluetoothManager(getActivity());

    mBluetoothApi = new IBluetoothApi() {
        @Override
        public boolean setEnable(boolean enable) {
            if (isLock()) {
                CLog.w("Bluetooth is turning on/off, please wait a minute.");
                return false;
            }
            mBluetooth.getBluetoothAdapter().setBluetoothEnabled(enable);
            CLog.e("bluetooth ui state is " + mBluetooth.getBluetoothAdapter().getBluetoothState());
            lockUI();
            return true;
        }

        @Override
        public boolean isEnabled() {
            return mBluetooth.getBluetoothAdapter().isEnabled();
        }
```
SettingsLib/src/main/java/com/android/settingslib/bluetooth/LocalBluetoothManager.

java

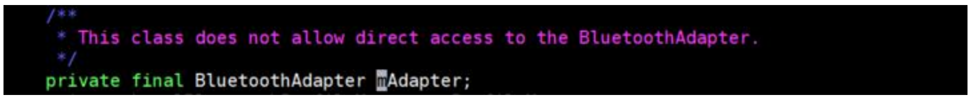
private LocalBluetoothManager mBluetooth;


```
public LocalBluetoothAdapter getBluetoothAdapter() {
    return mLocalAdapter;
}
```

```
private final LocalBluetoothAdapter mLocalAdapter;
```

SettingsLib/src/main/java/com/android/settingslib/bluetooth/LocalBluetoothAdapter.java



```java
public void setBluetoothEnabled(boolean enabled) {
    boolean success = enabled
        ? mAdapter.enable()
        : mAdapter.disable();

    if (success) {
        setBluetoothStateInt(enabled
            ? BluetoothAdapter.STATE_TURNING_ON
            : BluetoothAdapter.STATE_TURNING_OFF);
    } else {
        if (Utils.V) {
            Log.v(TAG, "setBluetoothEnabled call, manager didn't return " +
                "success for enabled: " + enabled);
        }

        syncBluetoothState();
    }
}
```



```java
/**
 * This class does not allow direct access to the BluetoothAdapter.
 */
private final BluetoothAdapter mAdapter;
```

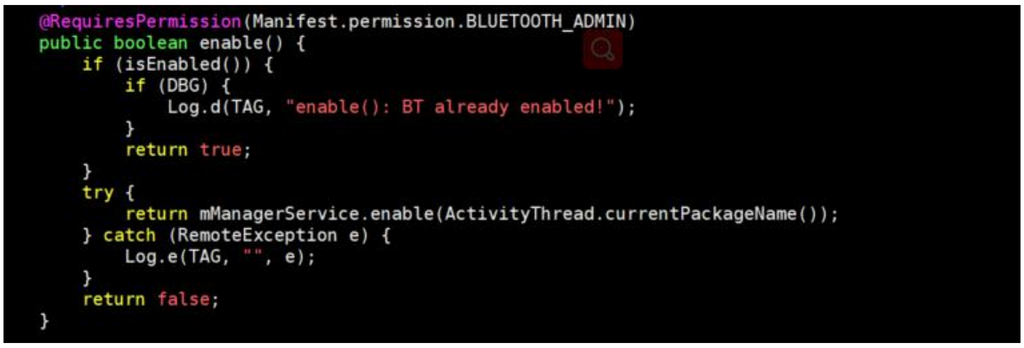
2.2 Log 打印

```
07-15 14:08:50.727  4561  4584 D Evan      : handleKeyEvent:23
07-15 14:08:50.778  4783  4783 D SettingsBaseFragment: onViewCreated:
class=BluetoothDevicesFragment{c5d6245 #1 id=0x7f0f0080
com.cvte.settings.fragment.BluetoothDevicesFragment}
07-15 14:08:50.804  4783  4783 D SettingsBaseFragment: onStart: class=BluetoothDevicesFragment{c5d6245
#1 id=0x7f0f0080 com.cvte.settings.fragment.BluetoothDevicesFragment}
07-15 14:08:50.784  4783  4783 D CLog      : [BluetoothDevicesFragment.java:317::createMenuItem]:
createMenuItem SwitchBluetoothEnable
07-15 14:08:50.806  4783  4783 D CLog      :   class=SwitchBluetoothEnable
07-15 14:08:50.806  4783  4783 D CLog      :   status=ENABLE
07-15 14:08:50.806  4783  4783 D CLog      :   menuId=SwitchBluetoothEnable
```

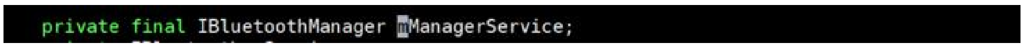
3. Framework 层

3.1 代码实现

Framework/base/core/java/android/bluetooth/BluetoothAdapter.java

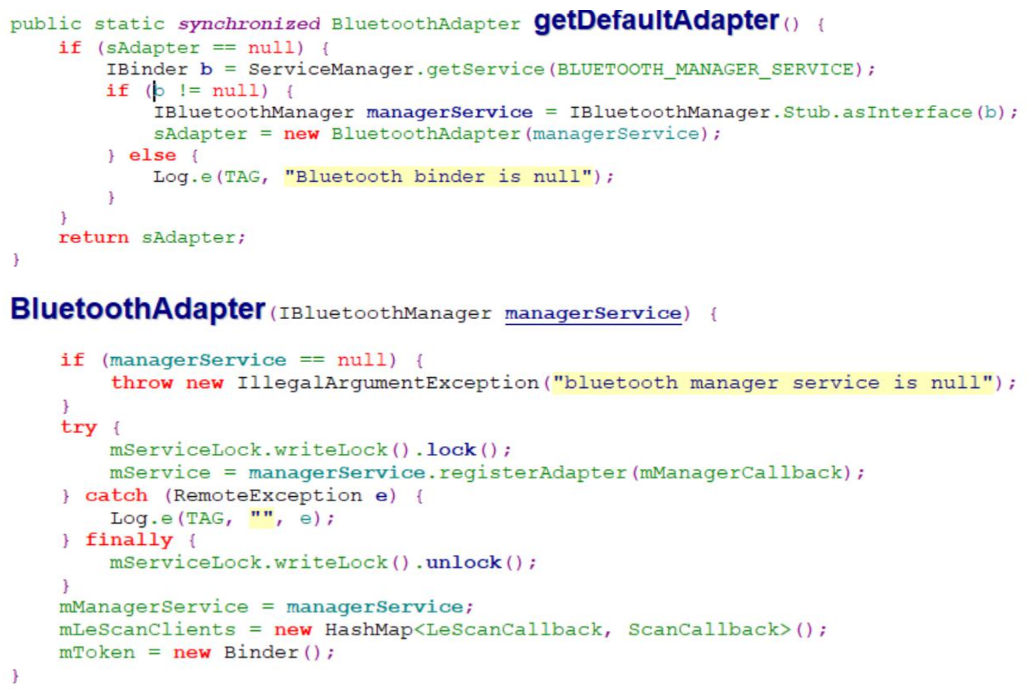


```java
@RequiresPermission(Manifest.permission.BLUETOOTH_ADMIN)
public boolean enable() {
    if (isEnabled()) {
        if (DBG) {
            Log.d(TAG, "enable(): BT already enabled!");
        }
        return true;
    }
    try {
        return mManagerService.enable(ActivityThread.currentPackageName());
    } catch (RemoteException e) {
        Log.e(TAG, "", e);
    }
    return false;
}
```



```java
private final IBluetoothManager mManagerService;
```

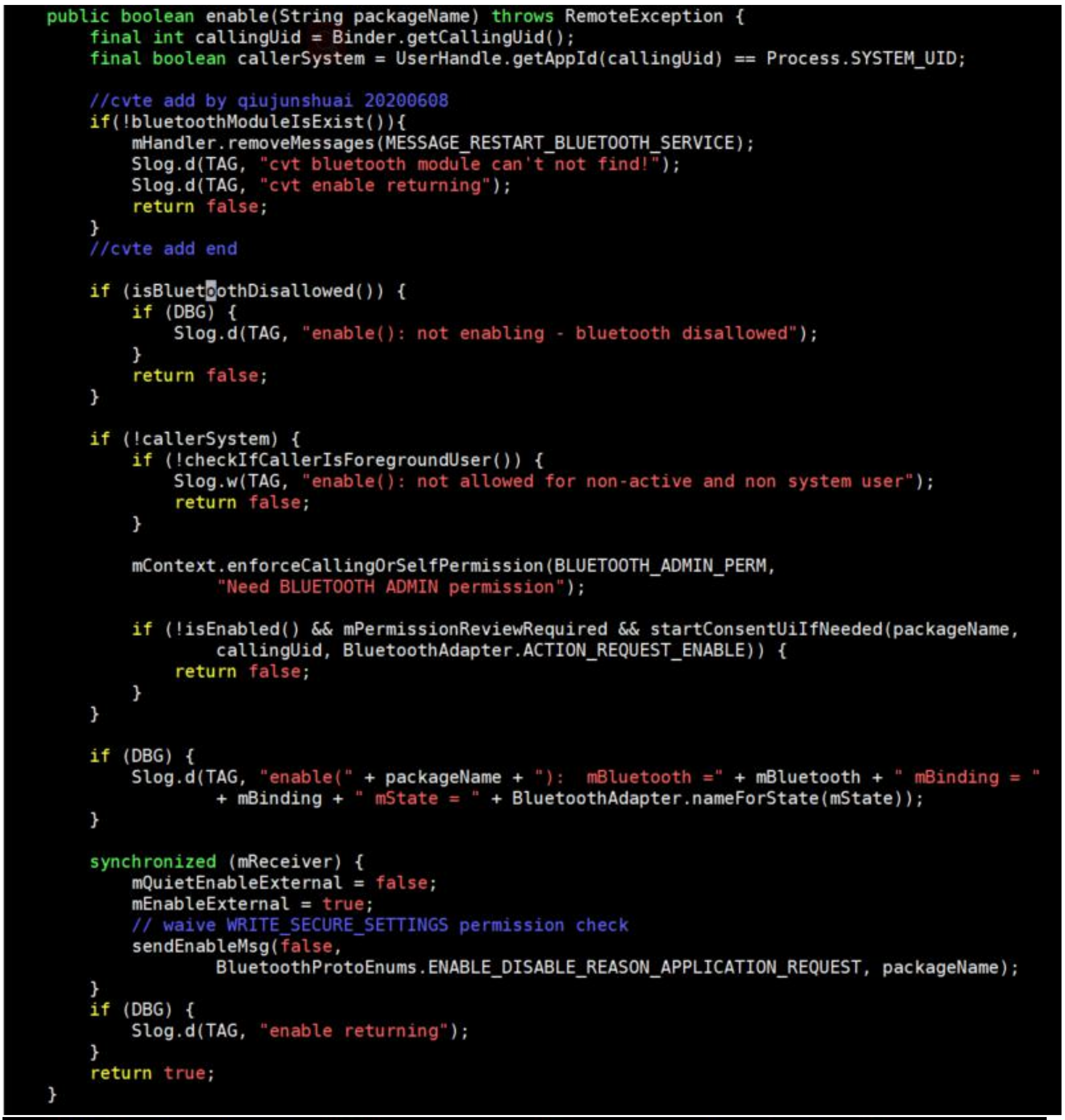
mManagerService 是什么呢？其实就是 BluetoothManagerService 的一个 proxy（代理）。在 getDefaultAdapter()可看到：



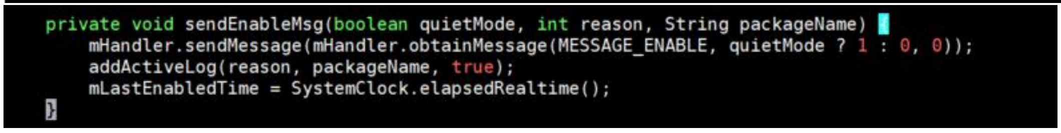
```java
public static synchronized BluetoothAdapter getDefaultAdapter() {
    if (sAdapter == null) {
        IBinder b = ServiceManager.getService(BLUETOOTH_MANAGER_SERVICE);
        if (b != null) {
            IBluetoothManager managerService = IBluetoothManager.Stub.asInterface(b);
            sAdapter = new BluetoothAdapter(managerService);
        } else {
            Log.e(TAG, "Bluetooth binder is null");
        }
    }
    return sAdapter;
}

BluetoothAdapter(IBluetoothManager managerService) {

    if (managerService == null) {
        throw new IllegalArgumentException("bluetooth manager service is null");
    }
    try {
        mServiceLock.writeLock().lock();
        mService = managerService.registerAdapter(mManagerCallback);
    } catch (RemoteException e) {
        Log.e(TAG, "", e);
    } finally {
        mServiceLock.writeLock().unlock();
    }
    mManagerService = managerService;
    mLeScanClients = new HashMap<LeScanCallback, ScanCallback>();
    mToken = new Binder();
}
```

它是通过 ServiceManager 获取了一个系统服务，然后转换为了 IBluetoothManager 接口，让 mManagerService 作为了 bluetoothmanagerservice 服务的代理。

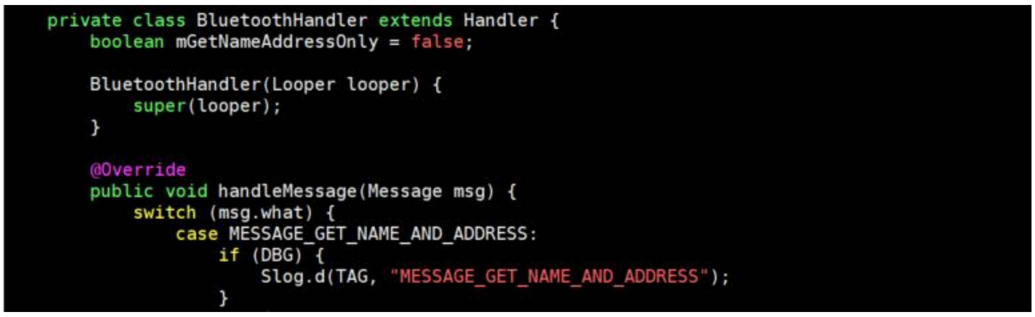
Framework/base/services/core/java/com/android/server/BluetoothManagerService.java



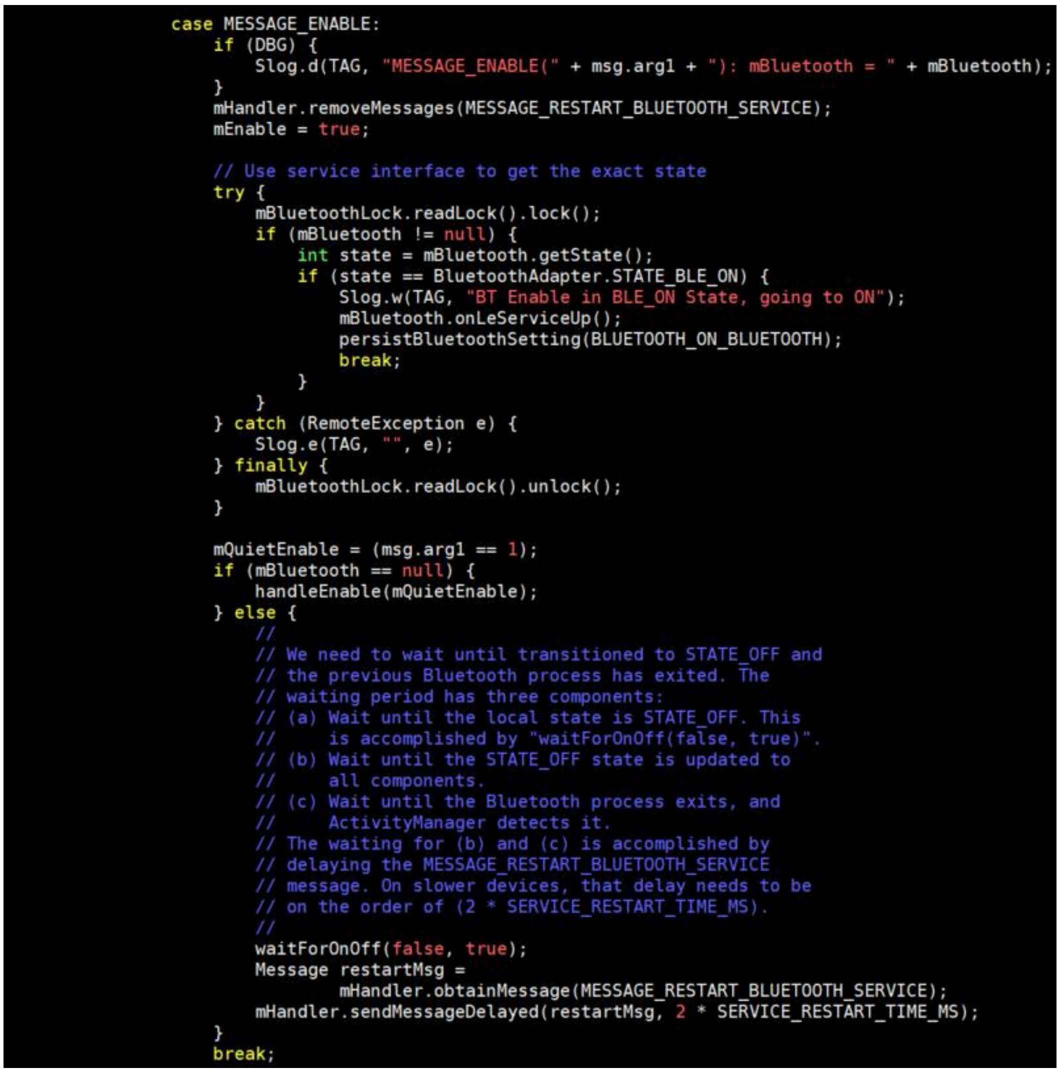
```
    public boolean enable(String packageName) throws RemoteException {
        final int callingUid = Binder.getCallingUid();
        final boolean callerSystem = UserHandle.getAppId(callingUid) == Process.SYSTEM_UID;

        //cvte add by qiujunshuai 20200608
        if(!bluetoothModuleIsExist()){
            mHandler.removeMessages(MESSAGE_RESTART_BLUETOOTH_SERVICE);
            Slog.d(TAG, "cvt bluetooth module can't not find!");
            Slog.d(TAG, "cvt enable returning");
            return false;
        }
        //cvte add end

        if (isBluetoothDisallowed()) {
            if (DBG) {
                Slog.d(TAG, "enable(): not enabling - bluetooth disallowed");
            }
            return false;
        }

        if (!callerSystem) {
            if (!checkIfCallerIsForegroundUser()) {
                Slog.w(TAG, "enable(): not allowed for non-active and non system user");
                return false;
            }

            mContext.enforceCallingOrSelfPermission(BLUETOOTH_ADMIN_PERM,
                    "Need BLUETOOTH ADMIN permission");

            if (!isEnabled() && mPermissionReviewRequired && startConsentUiIfNeeded(packageName,
                    callingUid, BluetoothAdapter.ACTION_REQUEST_ENABLE)) {
                return false;
            }
        }

        if (DBG) {
            Slog.d(TAG, "enable(" + packageName + "):  mBluetooth =" + mBluetooth + " mBinding = "
                    + mBinding + " mState = " + BluetoothAdapter.nameForState(mState));
        }

        synchronized (mReceiver) {
            mQuietEnableExternal = false;
            mEnableExternal = true;
            // waive WRITE_SECURE_SETTINGS permission check
            sendEnableMsg(false,
                    BluetoothProtoEnums.ENABLE_DISABLE_REASON_APPLICATION_REQUEST, packageName);
        }
        if (DBG) {
            Slog.d(TAG, "enable returning");
        }
        return true;
    }
```



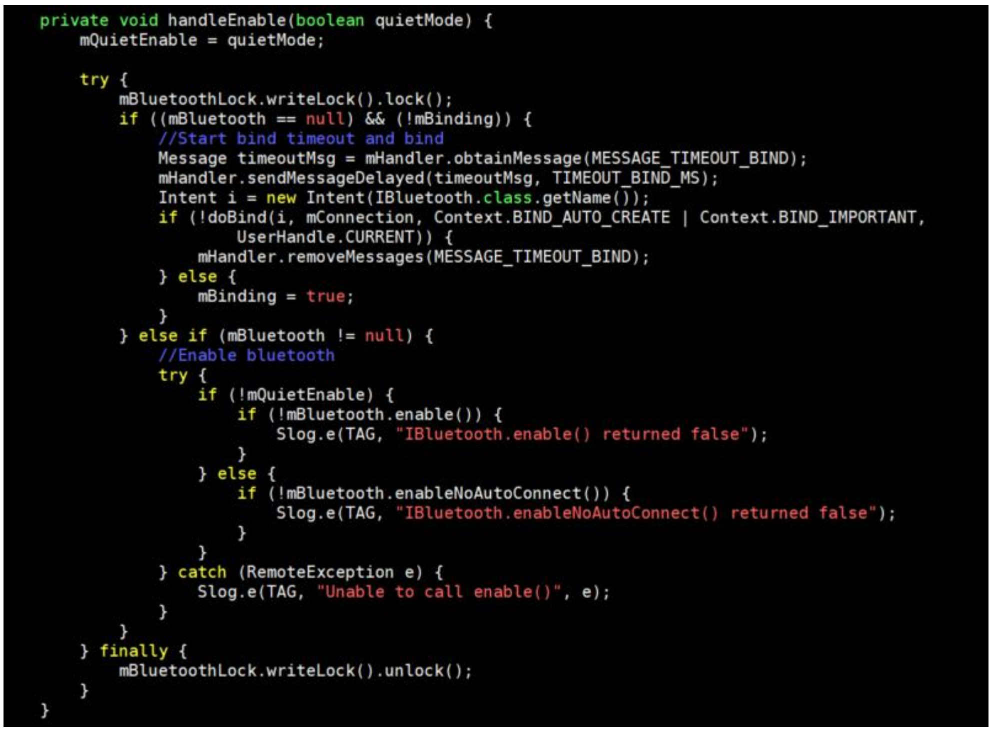
```
    private void sendEnableMsg(boolean quietMode, int reason, String packageName) {
        mHandler.sendMessage(mHandler.obtainMessage(MESSAGE_ENABLE, quietMode ? 1 : 0, 0));
        addActiveLog(reason, packageName, true);
        mLastEnabledTime = SystemClock.elapsedRealtime();
    }
```



```
    private class BluetoothHandler extends Handler {
        boolean mGetNameAddressOnly = false;

        BluetoothHandler(Looper looper) {
            super(looper);
        }

        @Override
        public void handleMessage(Message msg) {
            switch (msg.what) {
                case MESSAGE_GET_NAME_AND_ADDRESS:
                    if (DBG) {
                        Slog.d(TAG, "MESSAGE_GET_NAME_AND_ADDRESS");
                    }
```



```
                case MESSAGE_ENABLE:
                    if (DBG) {
                        Slog.d(TAG, "MESSAGE_ENABLE(" + msg.arg1 + "): mBluetooth = " + mBluetooth);
                    }
                    mHandler.removeMessages(MESSAGE_RESTART_BLUETOOTH_SERVICE);
                    mEnable = true;

                    // Use service interface to get the exact state
                    try {
                        mBluetoothLock.readLock().lock();
                        if (mBluetooth != null) {
                            int state = mBluetooth.getState();
                            if (state == BluetoothAdapter.STATE_BLE_ON) {
                                Slog.w(TAG, "BT Enable in BLE_ON State, going to ON");
                                mBluetooth.onLeServiceUp();
                                persistBluetoothSetting(BLUETOOTH_ON_BLUETOOTH);
                                break;
                            }
                        }
                    } catch (RemoteException e) {
                        Slog.e(TAG, "", e);
                    } finally {
                        mBluetoothLock.readLock().unlock();
                    }

                    mQuietEnable = (msg.arg1 == 1);
                    if (mBluetooth == null) {
                        handleEnable(mQuietEnable);
                    } else {
                        //
                        // We need to wait until transitioned to STATE_OFF and
                        // the previous Bluetooth process has exited. The
                        // waiting period has three components:
                        // (a) Wait until the local state is STATE_OFF. This
                        //     is accomplished by "waitForOnOff(false, true)".
                        // (b) Wait until the STATE_OFF state is updated to
                        //     all components.
                        // (c) Wait until the Bluetooth process exits, and
                        //     ActivityManager detects it.
                        // The waiting for (b) and (c) is accomplished by
                        // delaying the MESSAGE_RESTART_BLUETOOTH_SERVICE
                        // message. On slower devices, that delay needs to be
                        // on the order of (2 * SERVICE_RESTART_TIME_MS).
                        //
                        waitForOnOff(false, true);
                        Message restartMsg =
                                mHandler.obtainMessage(MESSAGE_RESTART_BLUETOOTH_SERVICE);
                        mHandler.sendMessageDelayed(restartMsg, 2 * SERVICE_RESTART_TIME_MS);
                    }
                    break;
```



```
    private void handleEnable(boolean quietMode) {
        mQuietEnable = quietMode;

        try {
            mBluetoothLock.writeLock().lock();
            if ((mBluetooth == null) && (!mBinding)) {
                //Start bind timeout and bind
                Message timeoutMsg = mHandler.obtainMessage(MESSAGE_TIMEOUT_BIND);
                mHandler.sendMessageDelayed(timeoutMsg, TIMEOUT_BIND_MS);
                Intent i = new Intent(IBluetooth.class.getName());
                if (!doBind(i, mConnection, Context.BIND_AUTO_CREATE | Context.BIND_IMPORTANT,
                        UserHandle.CURRENT)) {
                    mHandler.removeMessages(MESSAGE_TIMEOUT_BIND);
                } else {
                    mBinding = true;
                }
            } else if (mBluetooth != null) {
                //Enable bluetooth
                try {
                    if (!mQuietEnable) {
                        if (!mBluetooth.enable()) {
                            Slog.e(TAG, "IBluetooth.enable() returned false");
                        }
                    } else {
                        if (!mBluetooth.enableNoAutoConnect()) {
                            Slog.e(TAG, "IBluetooth.enableNoAutoConnect() returned false");
                        }
                    }
                } catch (RemoteException e) {
                    Slog.e(TAG, "Unable to call enable()", e);
                }
            }
        } finally {
            mBluetoothLock.writeLock().unlock();
        }
    }
```

这里通过 AIDL 的方式，调用 Bluetooth App 中的 AdapterService 。先绑定服务，然后注册 Ibluetooth 回调函数，之后调用 enable 方法方法开启蓝牙。所以之后就从 Framworks 跳到 Bluetooth APP 中继续分析。

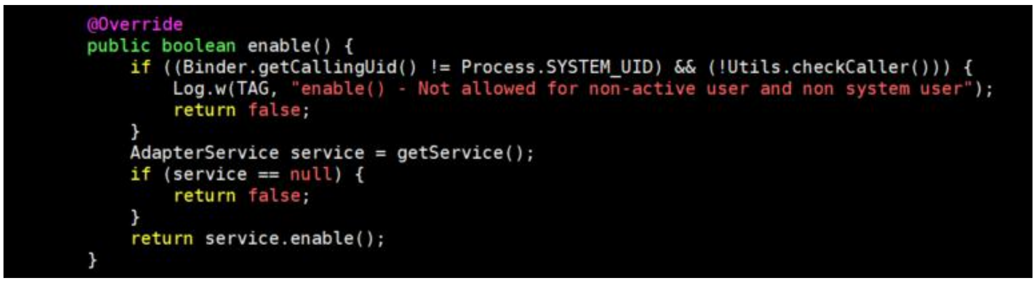
3.2 Log 打印

```
07-15 14:08:51.724  4783  4783 E BluetoothAdapter: td: BluetoothAdapter.java enable.
07-15 14:08:51.724  3368  3448 E BluetoothManagerService: td: BluetoothManagerService.java enable().
07-15 14:08:51.724  3368  3448 D BluetoothManagerService: cvt bluetooth sys.usb.bluetooth = 1
07-15 14:08:51.724  3368  3448 E BluetoothManagerService: td: BluetoothManagerService.java
sendEnableMsg().
07-15 14:08:51.725  3368  3417 E BluetoothManagerService: td: BluetoothManagerService.java invoke
handleEnable mQuietEnable=false
07-15 14:08:51.725  3368  3417 E BluetoothManagerService: td: BluetoothManagerService.java handleEnable()
enter.
07-15 14:08:51.725  3368  3417 E BluetoothManagerService: td: BluetoothManagerService.java doBind() enter.
07-15 14:08:51.743  3368  3412 I ActivityManager: Start proc 4841:com.android.bluetooth/1002 for service
com.android.bluetooth/.btservice.AdapterService
```

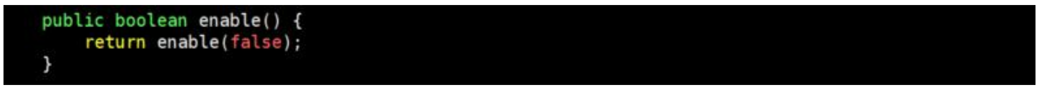
4. APP 层

4.1 代码实现

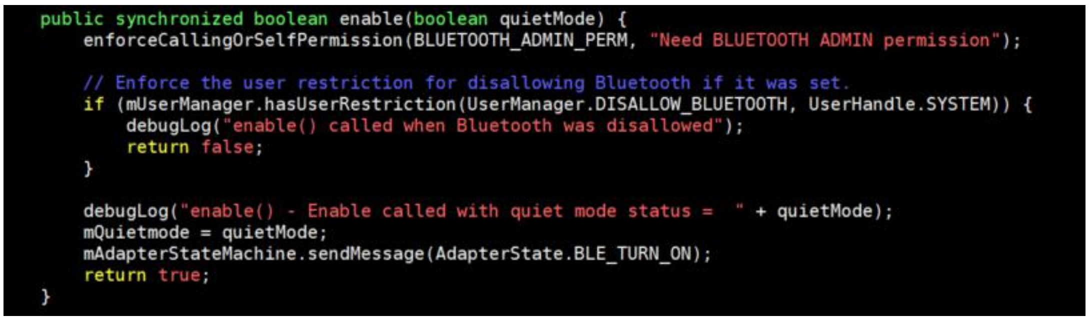
Packages/apps/Bluetooth/src/com/android/bluetooth/btservice/AdapterService.java



```
        @Override
        public boolean enable() {
            if ((Binder.getCallingUid() != Process.SYSTEM_UID) && (!Utils.checkCaller())) {
                Log.w(TAG, "enable() - Not allowed for non-active user and non system user");
                return false;
            }
            AdapterService service = getService();
            if (service == null) {
                return false;
            }
            return service.enable();
        }
```

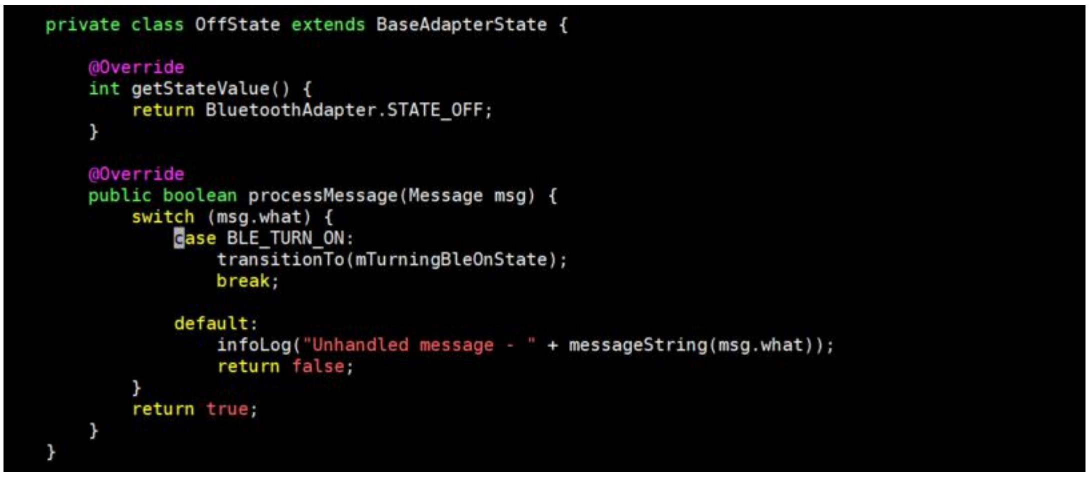


```
    public boolean enable() {
        return enable(false);
    }
```

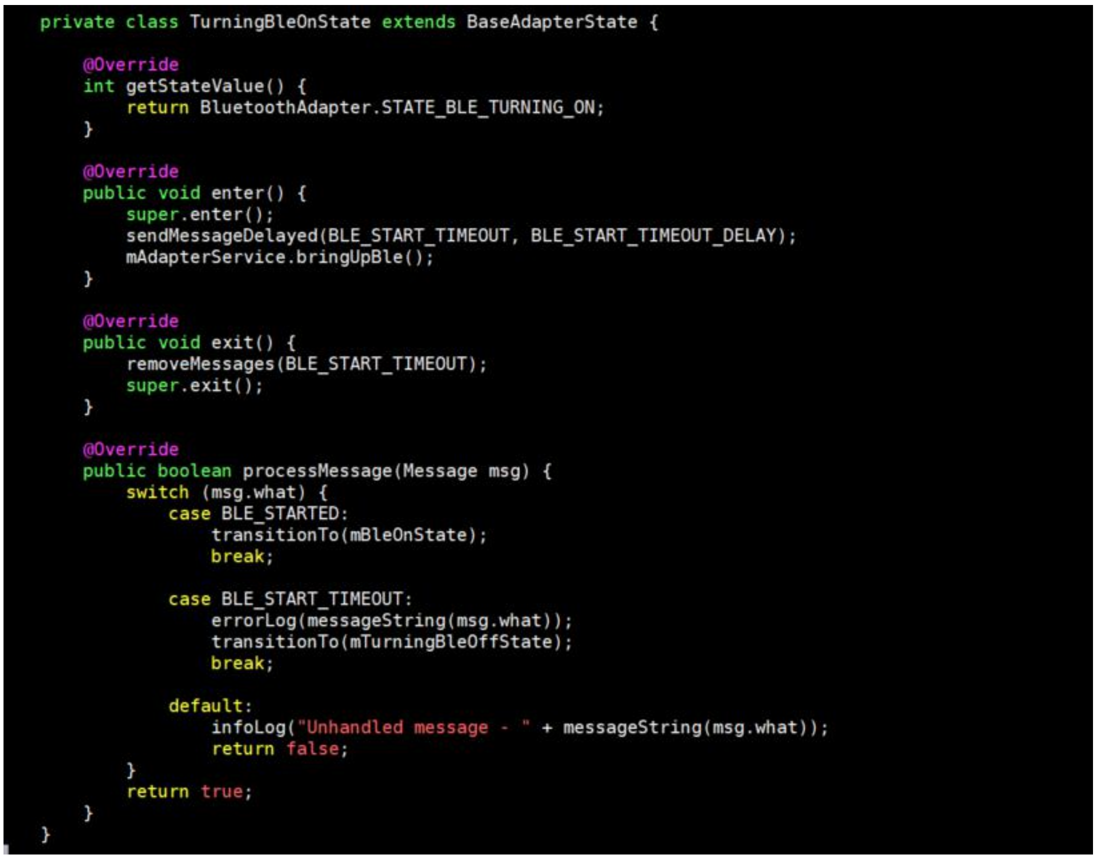


```
    public synchronized boolean enable(boolean quietMode) {
        enforceCallingOrSelfPermission(BLUETOOTH_ADMIN_PERM, "Need BLUETOOTH ADMIN permission");

        // Enforce the user restriction for disallowing Bluetooth if it was set.
        if (mUserManager.hasUserRestriction(UserManager.DISALLOW_BLUETOOTH, UserHandle.SYSTEM)) {
            debugLog("enable() called when Bluetooth was disallowed");
            return false;
        }

        debugLog("enable() - Enable called with quiet mode status =  " + quietMode);
        mQuietmode = quietMode;
        mAdapterStateMachine.sendMessage(AdapterState.BLE_TURN_ON);
        return true;
    }
```

Packages/apps/Bluetooth/src/com/android/bluetooth/btservice/AdapterState.java

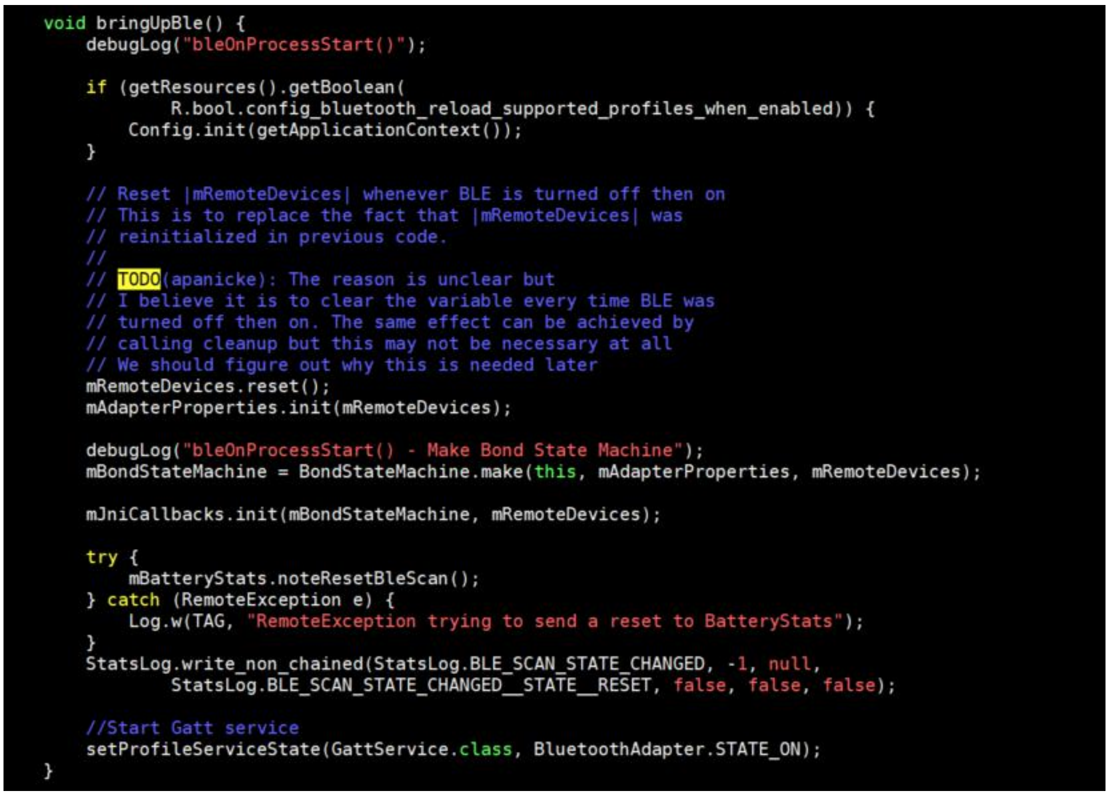


```
    private class OffState extends BaseAdapterState {

        @Override
        int getStateValue() {
            return BluetoothAdapter.STATE_OFF;
        }

        @Override
        public boolean processMessage(Message msg) {
            switch (msg.what) {
                case BLE_TURN_ON:
                    transitionTo(mTurningBleOnState);
                    break;

                default:
                    infoLog("Unhandled message - " + messageString(msg.what));
                    return false;
            }
            return true;
        }
    }
```



```
private class TurningBleOnState extends BaseAdapterState {

    @Override
    int getStateValue() {
        return BluetoothAdapter.STATE_BLE_TURNING_ON;
    }

    @Override
    public void enter() {
        super.enter();
        sendMessageDelayed(BLE_START_TIMEOUT, BLE_START_TIMEOUT_DELAY);
        mAdapterService.bringUpBle();
    }

    @Override
    public void exit() {
        removeMessages(BLE_START_TIMEOUT);
        super.exit();
    }

    @Override
    public boolean processMessage(Message msg) {
        switch (msg.what) {
            case BLE_STARTED:
                transitionTo(mBleOnState);
                break;

            case BLE_START_TIMEOUT:
                errorLog(messageString(msg.what));
                transitionTo(mTurningBleOffState);
                break;

            default:
                infoLog("Unhandled message - " + messageString(msg.what));
                return false;
        }
        return true;
    }
}
```

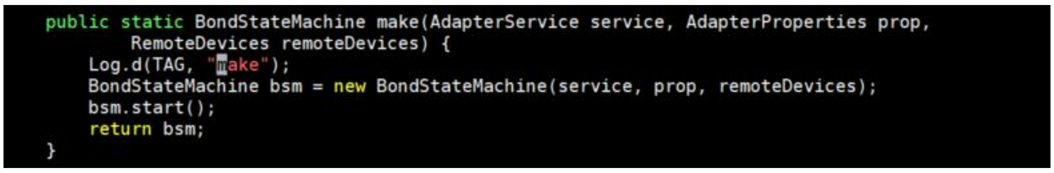
Packages/apps/Bluetooth/src/com/android/bluetooth/btservice/AdapterService.java



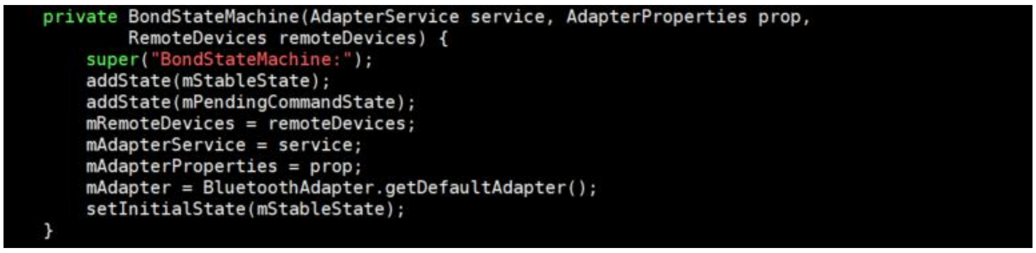
```
void bringUpBle() {
    debugLog("bleOnProcessStart()");

    if (getResources().getBoolean(
            R.bool.config_bluetooth_reload_supported_profiles_when_enabled)) {
        Config.init(getApplicationContext());
    }

    // Reset |mRemoteDevices| whenever BLE is turned off then on
    // This is to replace the fact that |mRemoteDevices| was
    // reinitialized in previous code.
    //
    // TODO(apanicke): The reason is unclear but
    // I believe it is to clear the variable every time BLE was
    // turned off then on. The same effect can be achieved by
    // calling cleanup but this may not be necessary at all
    // We should figure out why this is needed later
    mRemoteDevices.reset();
    mAdapterProperties.init(mRemoteDevices);

    debugLog("bleOnProcessStart() - Make Bond State Machine");
    mBondStateMachine = BondStateMachine.make(this, mAdapterProperties, mRemoteDevices);

    mJniCallbacks.init(mBondStateMachine, mRemoteDevices);

    try {
        mBatteryStats.noteResetBleScan();
    } catch (RemoteException e) {
        Log.w(TAG, "RemoteException trying to send a reset to BatteryStats");
    }
    StatsLog.write_non_chained(StatsLog.BLE_SCAN_STATE_CHANGED, -1, null,
            StatsLog.BLE_SCAN_STATE_CHANGED__STATE__RESET, false, false, false);

    //Start Gatt service
    setProfileServiceState(GattService.class, BluetoothAdapter.STATE_ON);
}
```

BondStateMachine.make(this, mAdapterProperties, mRemoteDevices);启动状态机。

Packages/apps/Bluetooth/src/com/android/bluetooth/btservice/BondStateMachine.java



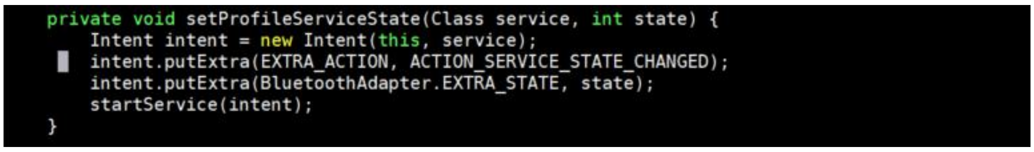
```
    public static BondStateMachine make(AdapterService service, AdapterProperties prop,
            RemoteDevices remoteDevices) {
        Log.d(TAG, "make");
        BondStateMachine bsm = new BondStateMachine(service, prop, remoteDevices);
        bsm.start();
        return bsm;
    }
```



```
    private BondStateMachine(AdapterService service, AdapterProperties prop,
            RemoteDevices remoteDevices) {
        super("BondStateMachine:");
        addState(mStableState);
        addState(mPendingCommandState);
        mRemoteDevices = remoteDevices;
        mAdapterService = service;
        mAdapterProperties = prop;
        mAdapter = BluetoothAdapter.getDefaultAdapter();
        setInitialState(mStableState);
    }
```

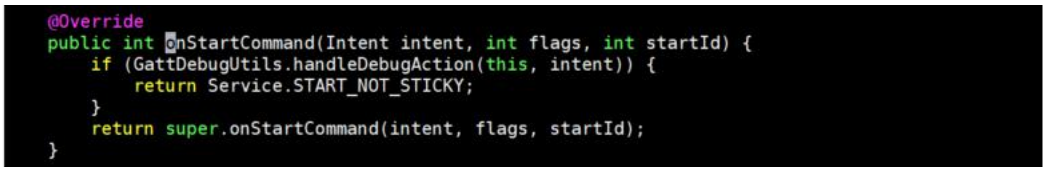
setProfileServiceState(GattService.class, BluetoothAdapter.STATE_ON); 启动 ProfileService 服务。

Packages/apps/Bluetooth/src/com/android/bluetooth/btservice/AdapterService.java



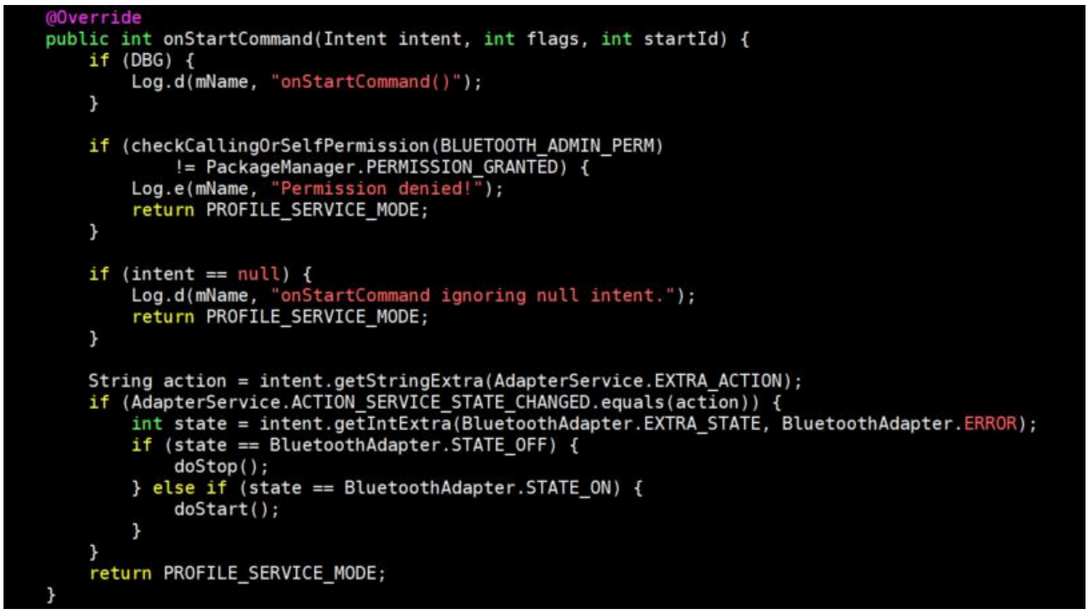
```
    private void setProfileServiceState(Class service, int state) {
        Intent intent = new Intent(this, service);
        intent.putExtra(EXTRA_ACTION, ACTION_SERVICE_STATE_CHANGED);
        intent.putExtra(BluetoothAdapter.EXTRA_STATE, state);
        startService(intent);
    }
```

Packages/apps/Bluetooth/src/com/android/bluetooth/gatt/GattService.java

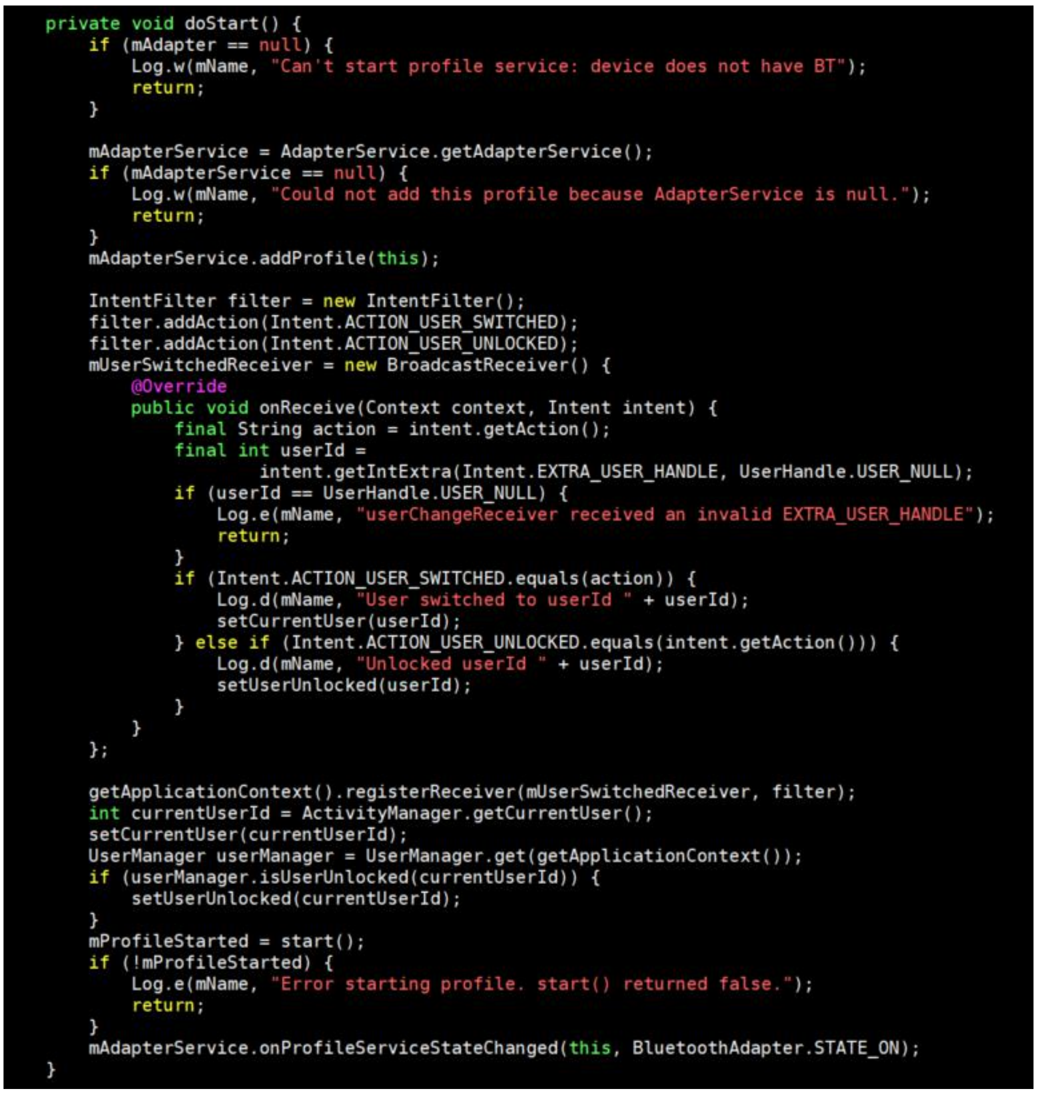


```
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        if (GattDebugUtils.handleDebugAction(this, intent)) {
            return Service.START_NOT_STICKY;
        }
        return super.onStartCommand(intent, flags, startId);
    }
```

Packages/apps/Bluetooth/src/com/android/bluetooth/btservice/ProfileService.java



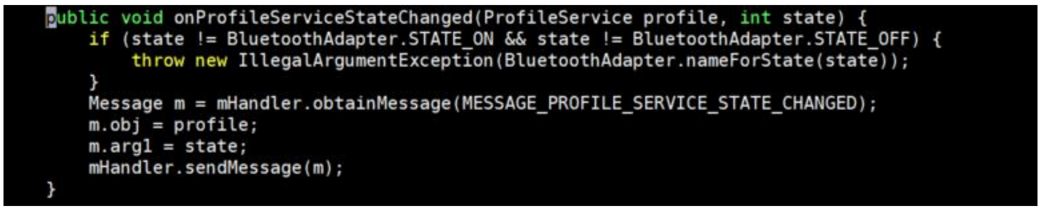
```
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        if (DBG) {
            Log.d(mName, "onStartCommand()");
        }

        if (checkCallingOrSelfPermission(BLUETOOTH_ADMIN_PERM)
                != PackageManager.PERMISSION_GRANTED) {
            Log.e(mName, "Permission denied!");
            return PROFILE_SERVICE_MODE;
        }

        if (intent == null) {
            Log.d(mName, "onStartCommand ignoring null intent.");
            return PROFILE_SERVICE_MODE;
        }

        String action = intent.getStringExtra(AdapterService.EXTRA_ACTION);
        if (AdapterService.ACTION_SERVICE_STATE_CHANGED.equals(action)) {
            int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE, BluetoothAdapter.ERROR);
            if (state == BluetoothAdapter.STATE_OFF) {
                doStop();
            } else if (state == BluetoothAdapter.STATE_ON) {
                doStart();
            }
        }
        return PROFILE_SERVICE_MODE;
    }
```



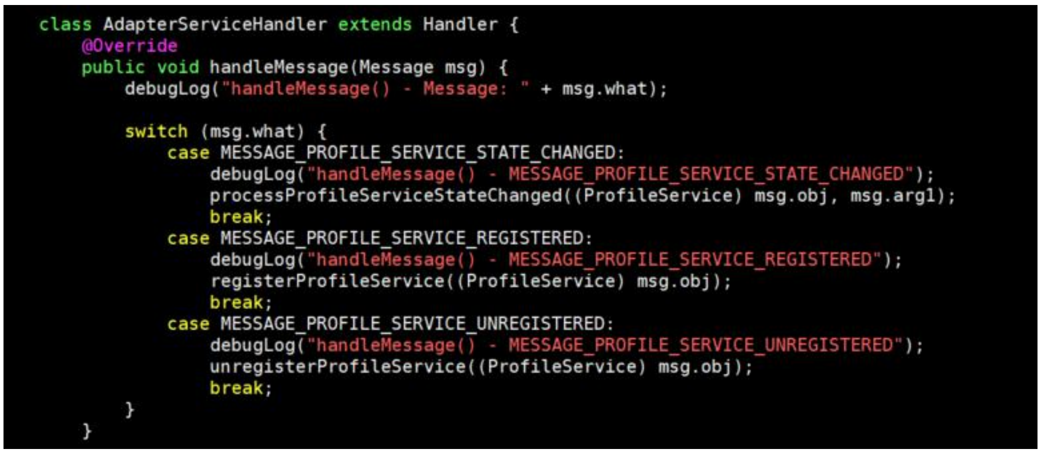
```
    private void doStart() {
        if (mAdapter == null) {
            Log.w(mName, "Can't start profile service: device does not have BT");
            return;
        }

        mAdapterService = AdapterService.getAdapterService();
        if (mAdapterService == null) {
            Log.w(mName, "Could not add this profile because AdapterService is null.");
            return;
        }
        mAdapterService.addProfile(this);

        IntentFilter filter = new IntentFilter();
        filter.addAction(Intent.ACTION_USER_SWITCHED);
        filter.addAction(Intent.ACTION_USER_UNLOCKED);
        mUserSwitchedReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                final String action = intent.getAction();
                final int userId =
                        intent.getIntExtra(Intent.EXTRA_USER_HANDLE, UserHandle.USER_NULL);
                if (userId == UserHandle.USER_NULL) {
                    Log.e(mName, "userChangeReceiver received an invalid EXTRA_USER_HANDLE");
                    return;
                }
                if (Intent.ACTION_USER_SWITCHED.equals(action)) {
                    Log.d(mName, "User switched to userId " + userId);
                    setCurrentUser(userId);
                } else if (Intent.ACTION_USER_UNLOCKED.equals(intent.getAction())) {
                    Log.d(mName, "Unlocked userId " + userId);
                    setUserUnlocked(userId);
                }
            }
        };

        getApplicationContext().registerReceiver(mUserSwitchedReceiver, filter);
        int currentUserId = ActivityManager.getCurrentUser();
        setCurrentUser(currentUserId);
        UserManager userManager = UserManager.get(getApplicationContext());
        if (userManager.isUserUnlocked(currentUserId)) {
            setUserUnlocked(currentUserId);
        }
        mProfileStarted = start();
        if (!mProfileStarted) {
            Log.e(mName, "Error starting profile. start() returned false.");
            return;
        }
        mAdapterService.onProfileServiceStateChanged(this, BluetoothAdapter.STATE_ON);
    }
```

因为启动服务时传入的参数是 ACTION_SERVICE_STATE_CHANGED 和 BluetoothAdapter.STATE_ON，所以调用 doStart。doStart 里面调用了抽象方法 start（实现是在子类 GattService 里面，做了一些预处理，包括 initializeNative、启动一些 manager）。

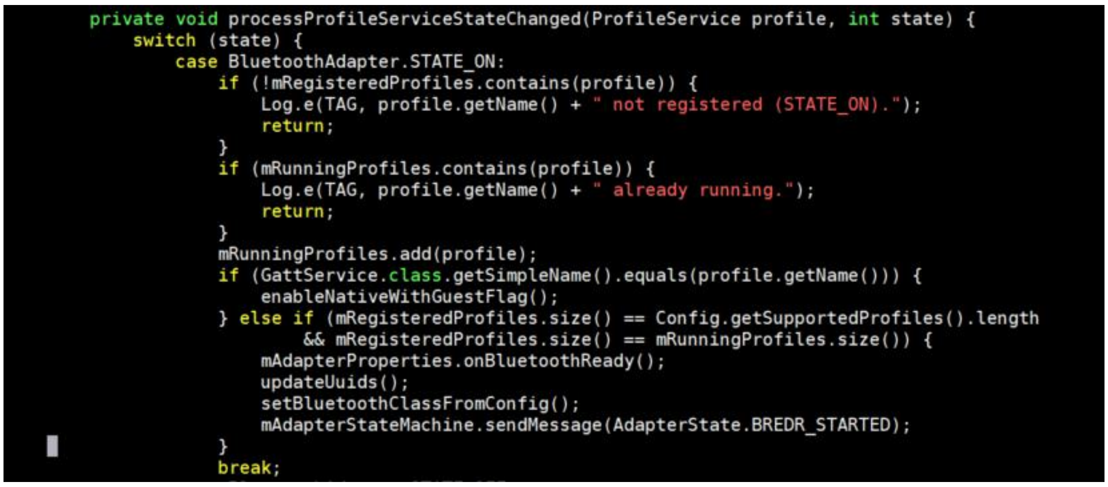
调用 AdapterService.onProfileServiceStateChanged 方法，传递 STATE_ON 消息，该消息会包装在 MESSAGE_PROFILE_SERVICE_STATE_CHANGED 类型里，由 AdapterService 的 handler 方法处理。



```
public void onProfileServiceStateChanged(ProfileService profile, int state) {
    if (state != BluetoothAdapter.STATE_ON && state != BluetoothAdapter.STATE_OFF) {
        throw new IllegalArgumentException(BluetoothAdapter.nameForState(state));
    }
    Message m = mHandler.obtainMessage(MESSAGE_PROFILE_SERVICE_STATE_CHANGED);
    m.obj = profile;
    m.arg1 = state;
    mHandler.sendMessage(m);
}
```

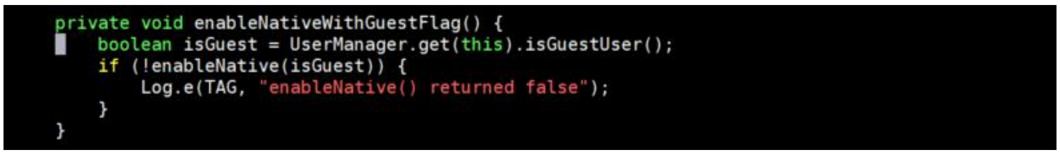


```
class AdapterServiceHandler extends Handler {
    @Override
    public void handleMessage(Message msg) {
        debugLog("handleMessage() - Message: " + msg.what);

        switch (msg.what) {
            case MESSAGE_PROFILE_SERVICE_STATE_CHANGED:
                debugLog("handleMessage() - MESSAGE_PROFILE_SERVICE_STATE_CHANGED");
                processProfileServiceStateChanged((ProfileService) msg.obj, msg.arg1);
                break;
            case MESSAGE_PROFILE_SERVICE_REGISTERED:
                debugLog("handleMessage() - MESSAGE_PROFILE_SERVICE_REGISTERED");
                registerProfileService((ProfileService) msg.obj);
                break;
            case MESSAGE_PROFILE_SERVICE_UNREGISTERED:
                debugLog("handleMessage() - MESSAGE_PROFILE_SERVICE_UNREGISTERED");
                unregisterProfileService((ProfileService) msg.obj);
                break;
        }
    }
```



```
    private void processProfileServiceStateChanged(ProfileService profile, int state) {
        switch (state) {
            case BluetoothAdapter.STATE_ON:
                if (!mRegisteredProfiles.contains(profile)) {
                    Log.e(TAG, profile.getName() + " not registered (STATE_ON).");
                    return;
                }
                if (mRunningProfiles.contains(profile)) {
                    Log.e(TAG, profile.getName() + " already running.");
                    return;
                }
                mRunningProfiles.add(profile);
                if (GattService.class.getSimpleName().equals(profile.getName())) {
                    enableNativeWithGuestFlag();
                } else if (mRegisteredProfiles.size() == Config.getSupportedProfiles().length
                        && mRegisteredProfiles.size() == mRunningProfiles.size()) {
                    mAdapterProperties.onBluetoothReady();
                    updateUuids();
                    setBluetoothClassFromConfig();
                    mAdapterStateMachine.sendMessage(AdapterState.BREDR_STARTED);
                }
                break;
```

会向 AdapterStateMachine 状态机发送 BLE_STARTED 消息，根据之前状态机已经由 OffState 切换成 PendingCommandState，所以消息由 PendingCommandState 状态处理,看 processMessage 的处理



```
private void enableNativeWithGuestFlag() {
    boolean isGuest = UserManager.get(this).isGuestUser();
    if (!enableNative(isGuest)) {
        Log.e(TAG, "enableNative() returned false");
    }
}
```

4.2 Log 打印

```
07-15 14:08:51.888    4841    4841 D BluetoothOppFileProvider: Initialized

07-15 14:08:51.911    4841    4841 V AdapterServiceConfig: Adding A2dpService

07-15 14:08:51.911    4841    4841 V AdapterServiceConfig: Adding A2dpSinkService
```

```
07-15 14:08:51.911   4841   4841 V AdapterServiceConfig: Adding HidHostService

07-15 14:08:51.912   4841   4841 V AdapterServiceConfig: Adding GattService

07-15 14:08:51.912   4841   4841 V AdapterServiceConfig: Adding AvrcpTargetService

07-15 14:08:51.912   4841   4841 V AdapterServiceConfig: Adding AvrcpControllerService

07-15 14:08:51.912   4841   4841 E BluetoothServiceJni: td:
com_android_bluetooth_btservice_AdapterService.cpp classInitNative enter

07-15 14:08:51.912   4841   4841 E             :
[0717/064531.128336:ERROR:com_android_bluetooth_btservice_AdapterService.cpp(613)]
hal_util_load_bt_librarytd:   path=libbluetooth.so

07-15 14:08:51.912   4841   4841 E             :
[0717/064531.147782:ERROR:com_android_bluetooth_btservice_AdapterService.cpp(631)]
hal_util_load_bt_librarytd:   hal_util_load_bt_library

07-15 14:08:51.937   4841   4841 I             :
[0715/140851.937646:INFO:com_android_bluetooth_btservice_AdapterService.cpp(630)]
hal_util_load_bt_library loaded HAL: btinterface=0x908f0244, handle=0xcee83e35

07-15 14:08:51.938   4841   4841 D BluetoothAdapterService: td: AdapterService.java onCreate() enter.

07-15 14:08:51.943   4841   4841 D AdapterState: make() - Creating AdapterState

07-15 14:08:51.945   4841   4841 I bt_btif : init

07-15 14:08:51.945   4841   4856 I AdapterState: OFF : entered

07-15 14:08:51.945   4841   4856 D AdapterProperties: Setting state to OFF

07-15 14:08:51.948   4841   4857 E bt_stack_manager: td: stack_manager.cc event_init_stack enter.

07-15 14:08:51.954   4841   4857 I bt_btif_core: btif_init_bluetooth entered

07-15 14:08:51.957   4841   4862 I bt_osi_thread: run_thread: thread id 4862, thread name bt_jni_workqueue
started

07-15 14:08:51.957   4841   4857 I bt_btif_core: btif_init_bluetooth finished

07-15 14:08:51.957   4841   4857 E bt_stack_manager: td: stack_manager.cc event_init_stack finished


07-15 14:08:51.982   4841   4841 D BluetoothAdapterService: onBind()

07-15 14:08:51.984   3368   3368 D BluetoothManagerService: BluetoothServiceConnection:
com.android.bluetooth.btservice.AdapterService
```

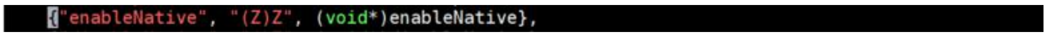
```
07-15 14:08:51.984    3368    3417 D BluetoothManagerService:
MESSAGE_BLUETOOTH_SERVICE_CONNECTED: 1
07-15 14:08:51.985    3368    3417 D BluetoothManagerService: Broadcasting onBluetoothServiceUp() to 4
receiver

07-15 14:08:51.986    4841    4853 E BluetoothAdapterService: td: AdapterService.java enable() invoke
service.enable()
07-15 14:08:51.988    4841    4853 D BluetoothAdapterService: td: AdapterService.java enable() send
AdapterState.BLE_TURN_ON
07-15 14:08:51.988    4841    4856 E AdapterState: td: AdapterState.java
offState->processMesaage->BLE_TURN_ON.

07-15 14:08:51.989    4841    4856 E AdapterState: td: AdapterSate.java TurningBleOnState->enter
07-15 14:08:51.989    4841    4856 D BluetoothAdapterService: bleOnProcessStart()
07-15 14:08:51.989    4841    4856 D BluetoothAdapterService: td: AdapterService.java bringUpBle() enter.
07-15 14:08:51.989    4841    4856 E AdapterProperties: td: AdapterProperties.java init() enter.
07-15 14:08:51.989    4841    4856 I AdapterProperties: init(), maxConnectedAudioDevices, default=5,
propertyOverlayed=5, finalValue=5

07-15 14:08:51.992    4841    4856 D BluetoothBondStateMachine: td: BondStateMachine.java make() invoke
start().
07-15 14:08:51.993    4841    4871 I BluetoothBondStateMachine: StableState(): Entering Off State
07-15 14:08:51.993    4841    4871 E BluetoothBondStateMachine: td: BondStateMachine.java stableState->enter.
07-15 14:08:51.995    4841    4856 E BluetoothAdapterService: td: AdapterService.java setProfileServiceState
state = 12

07-15 14:08:52.008    4841    4841 I BtGatt.JNI: classInitNative(L875): classInitNative: Success!
07-15 14:08:52.008    4841    4841 D GattService: onCreate
07-15 14:08:52.010    4841    4841 E BtGatt.GattService: td:GattService.java onStartCommand() enter.
07-15 14:08:52.010    4841    4841 E GattService: td: ProfileService.java onStartCommand() enter
```

```
07-15 14:08:52.011    4841    4841 E GattService: td: profileService.java doStart() enter.

07-15 14:08:52.011    4841    4841 D BluetoothAdapterService: getAdapterService() - returning

com.android.bluetooth.btservice.AdapterService@a10ca8a

07-15 14:08:52.015    4841    4841 E GattService: td: profileService.java doStart() invoke start().

07-15 14:08:52.015    4841    4841 E BtGatt.GattService: td: GattService.java start()


07-15 14:08:52.030    4841    4841 E BluetoothAdapterService: td: AdapterService.java

onProfileServiceStateChanged() send MESSAGE_PROFILE_SERVICE_STATE_CHANGED

07-15 14:08:52.031    4841    4841 D BluetoothAdapterService: td: AdapterService.java AdapterServiceHandler

handleMessage() - MESSAGE_PROFILE_SERVICE_STATE_CHANGED

07-15 14:08:52.031    4841    4841 E BluetoothAdapterService: td: AdapterService.java

processProfileServiceStateChanged BluetoothAdapter.STATE_ON

07-15 14:08:52.031    4841    4841 E BluetoothAdapterService: td: AdapterService.java

processProfileServiceStateChanged invoke enableNativeWithGuestFlag()

07-15 14:08:52.032    4841    4841 E BluetoothAdapterService: td: AdapterService.java

enableNativeWithGuestFlag() invoke enableNative(isGuest) isGuest=false
```

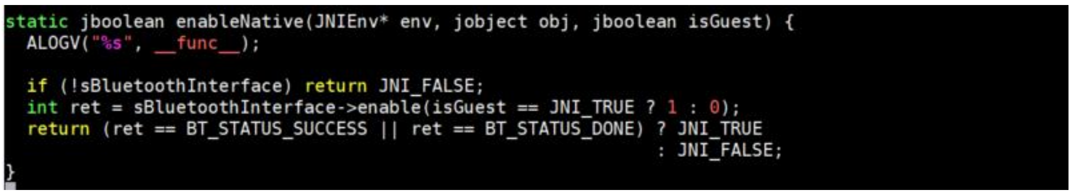
5. C++/C 层

5.1 代码实现

packages/apps/Bluetooth/jni/com_android_bluetooth_btservice_AdapterService.cpp



```
    {"enableNative", "(Z)Z", (void*)enableNative},
```

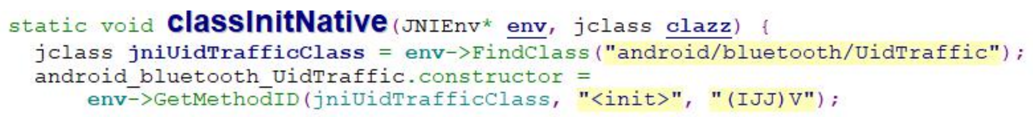


```
static jboolean enableNative(JNIEnv* env, jobject obj, jboolean isGuest) {
  ALOGV("%s", __func__);

  if (!sBluetoothInterface) return JNI_FALSE;
  int ret = sBluetoothInterface->enable(isGuest == JNI_TRUE ? 1 : 0);
  return (ret == BT_STATUS_SUCCESS || ret == BT_STATUS_DONE) ? JNI_TRUE
                                                              : JNI_FALSE;
}
```

通过调用“int ret = sBluetoothInterface->enable()”来驱动底层打开蓝牙开关。

发现 classInitNative 跟其他方法是不一样的，它的第二个参数是一个 clazz:

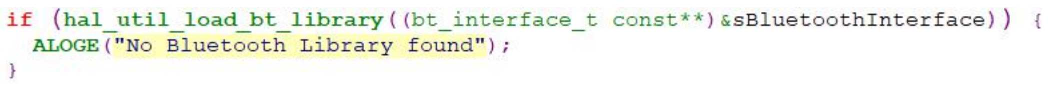
而其他的方法都是 jobject。



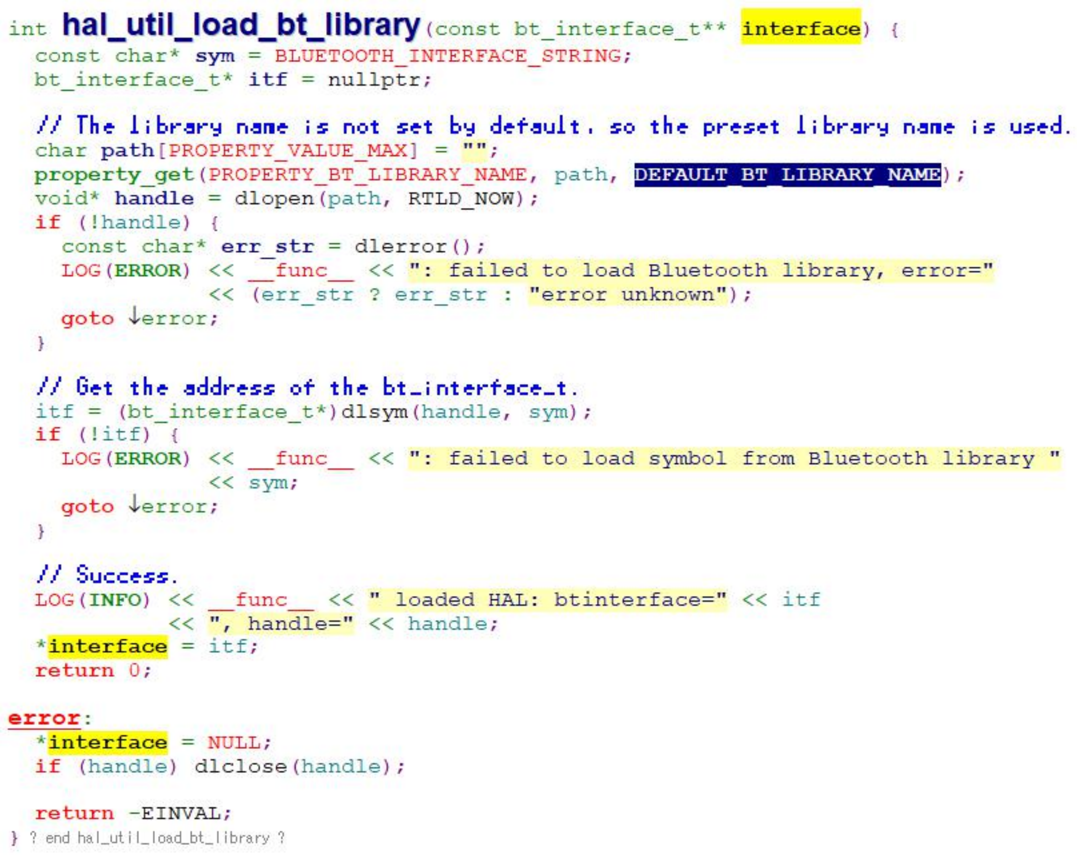
```
static void classInitNative(JNIEnv* env, jclass clazz) {
   jclass jniUidTrafficClass = env->FindClass("android/bluetooth/UidTraffic");
   android_bluetooth_UidTraffic.constructor =
       env->GetMethodID(jniUidTrafficClass, "<init>", "(IJJ)V");
```

通过追溯到 JAVA 层，我们发现前者是一个静态方法，因此它是属于类所有，而非对象：它的调用初始化时间也早于其他所有方法，位于 static 块中：

但是关键是 sBluetoothInterface 是怎么来的呢？这就需要分析刚才提到的 classInitNative 了：

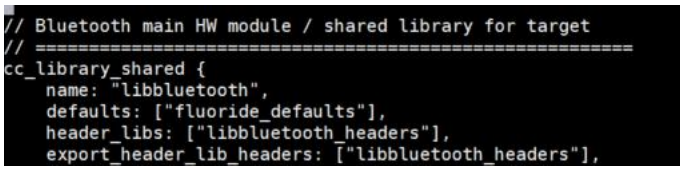


```
  if (hal_util_load_bt_library((bt_interface_t const**)&sBluetoothInterface)) {
    ALOGE("No Bluetooth Library found");
  }
```



```
int hal_util_load_bt_library(const bt_interface_t** interface) {
  const char* sym = BLUETOOTH_INTERFACE_STRING;
  bt_interface_t* itf = nullptr;

  // The library name is not set by default, so the preset library name is used.
  char path[PROPERTY_VALUE_MAX] = "";
  property_get(PROPERTY_BT_LIBRARY_NAME, path, DEFAULT_BT_LIBRARY_NAME);
  void* handle = dlopen(path, RTLD_NOW);
  if (!handle) {
    const char* err_str = dlerror();
    LOG(ERROR) << __func__ << ": failed to load Bluetooth library, error="
               << (err_str ? err_str : "error unknown");
    goto ↓error;
  }

  // Get the address of the bt_interface_t.
  itf = (bt_interface_t*)dlsym(handle, sym);
  if (!itf) {
    LOG(ERROR) << __func__ << ": failed to load symbol from Bluetooth library "
               << sym;
    goto ↓error;
  }

  // Success.
  LOG(INFO) << __func__ << " loaded HAL: btinterface=" << itf
            << ", handle=" << handle;
  *interface = itf;
  return 0;

error:
  *interface = NULL;
  if (handle) dlclose(handle);

  return -EINVAL;
} ? end hal_util_load_bt_library ?
```

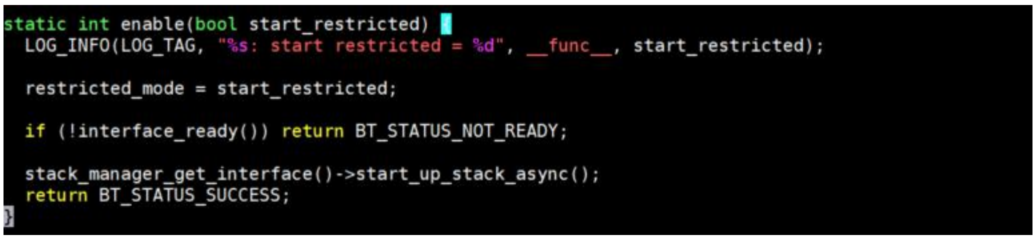
android/system/bt/main$ vi Android.bp



```
// Bluetooth main HW module / shared library for target
// ========================================================
cc_library_shared {
    name: "libbluetooth",
    defaults: ["fluoride_defaults"],
    header_libs: ["libbluetooth_headers"],
    export_header_lib_headers: ["libbluetooth_headers"],
```

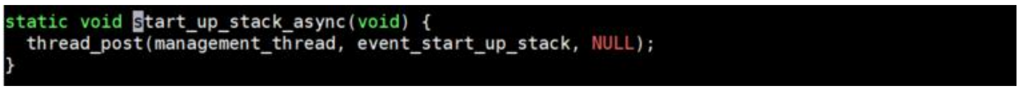
接下来就是 C 里面对打开蓝牙的实现。

system/bt/btif/src/bluetooth.cc

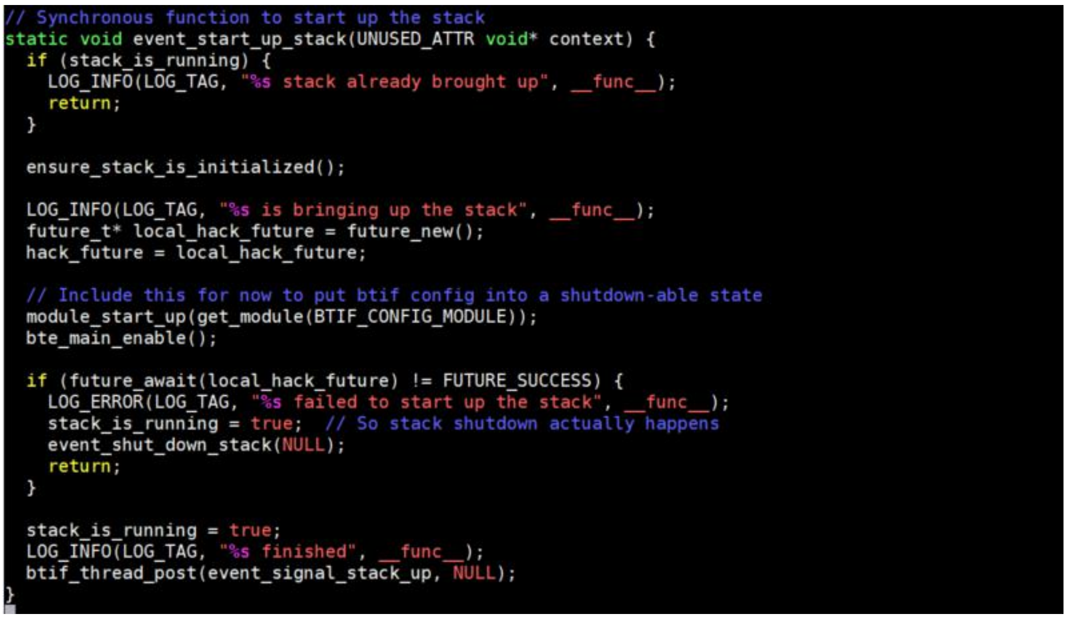


```
static int enable(bool start_restricted) {
  LOG_INFO(LOG_TAG, "%s: start restricted = %d", __func__, start_restricted);

  restricted_mode = start_restricted;

  if (!interface_ready()) return BT_STATUS_NOT_READY;

  stack_manager_get_interface()->start_up_stack_async();
  return BT_STATUS_SUCCESS;
}
```

system/bt/btif/src/stack_manager.cc

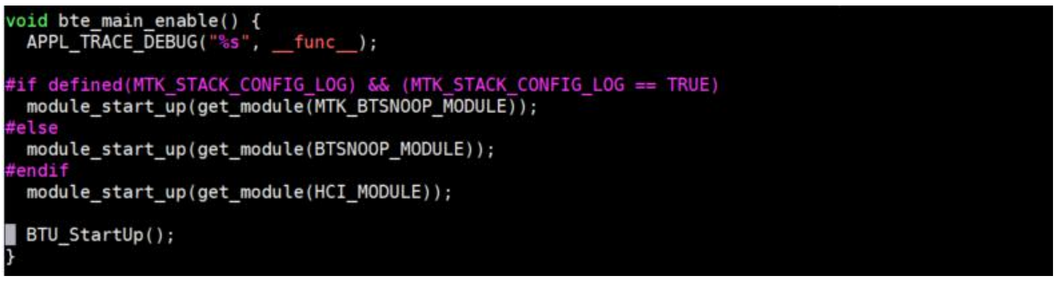


```
static void start_up_stack_async(void) {
  thread_post(management_thread, event_start_up_stack, NULL);
}
```



```
// Synchronous function to start up the stack
static void event_start_up_stack(UNUSED_ATTR void* context) {
  if (stack_is_running) {
    LOG_INFO(LOG_TAG, "%s stack already brought up", __func__);
    return;
  }

  ensure_stack_is_initialized();

  LOG_INFO(LOG_TAG, "%s is bringing up the stack", __func__);
  future_t* local_hack_future = future_new();
  hack_future = local_hack_future;

  // Include this for now to put btif config into a shutdown-able state
  module_start_up(get_module(BTIF_CONFIG_MODULE));
  bte_main_enable();

  if (future_await(local_hack_future) != FUTURE_SUCCESS) {
    LOG_ERROR(LOG_TAG, "%s failed to start up the stack", __func__);
    stack_is_running = true;  // So stack shutdown actually happens
    event_shut_down_stack(NULL);
    return;
  }

  stack_is_running = true;
  LOG_INFO(LOG_TAG, "%s finished", __func__);
  btif_thread_post(event_signal_stack_up, NULL);
}
```

system/bt/main/bte_main.cc



```
void bte_main_enable() {
  APPL_TRACE_DEBUG("%s", __func__);

#if defined(MTK_STACK_CONFIG_LOG) && (MTK_STACK_CONFIG_LOG == TRUE)
  module_start_up(get_module(MTK_BTSNOOP_MODULE));
#else
  module_start_up(get_module(BTSNOOP_MODULE));
#endif
  module_start_up(get_module(HCI_MODULE));

  BTU_StartUp();
}
```

5.2 Log 打印

```
07-15 14:08:52.032    4841    4841 E BluetoothServiceJni: td:

com_android_bluetooth_btservice_AdapterService.cpp enableNative enable()



07-15 14:08:52.032    4841    4841 E bt_btif : td: Bluetooth.cc enable: enable() start restricted = 0

07-15 14:08:52.033    4841    4857 E bt_stack_manager: td: Stack_manager.cc event_start_up_stack enter.



07-15 14:08:52.033    4841    4857 I bt_stack_manager: event_start_up_stack is bringing up the stack

07-15 14:08:52.033    4841    4857 I bt_core_module: module_start_up Starting module "btif_config_module"

07-15 14:08:52.033    4841    4857 I bt_core_module: module_start_up Started module "btif_config_module"

07-15 14:08:52.033    4841    4857 E bt_main : td:Bte_main.cc bte_main_enable enter.
```

```
07-15 14:08:52.033    4841    4857 I bt_core_module: module_start_up Starting module "mtk_btsnoop_module"


07-15 14:08:52.036    4841    4857 I bt_core_module: module_start_up Starting module "hci_module"
07-15 14:08:52.036    4841    4857 I bt_hci    : hci_module_start_up


07-15 14:08:52.036    4841    4880 I bt_osi_thread: run_thread: thread id 4880, thread name hci_thread started
07-15 14:08:52.036    4841    4857 D bt_hci    : hci_module_start_up starting async portion
07-15 14:08:52.036    4841    4880 I bt_hci    : hci_initialize
07-15 14:08:52.040    4841    4880 I bt_hci    : hci_initialize: IBluetoothHci::getService() returned 0xa4354180
(remote)
07-15 14:08:52.043    2583    4881 E android.hardware.bluetooth@1.0-impl: td: BluetoothHci.cc initialize() enter.
07-15 14:08:52.043    2583    4881 E android.hardware.bluetooth@1.0-impl: td: BluetoothHci.cc
VendorInterface::Initialize.
07-15 14:08:52.043    2583    4881 E android.hardware.bluetooth@1.0-impl: td: Vendor_interface.cc Initialize:
07-15 14:08:52.043    2583    4881 E android.hardware.bluetooth@1.0-impl: td: vendor_interface.cc Open enter


07-15 14:08:52.040    2583    4881 I android.hardware.bluetooth@1.0-impl: BluetoothHci::initialize()


07-15 14:08:52.043    2583    4881 D android.hardware.bluetooth@1.0-impl: Open vendor library loaded
07-15 14:08:52.043    2583    4881 E android.hardware.bluetooth@1.0-impl: td: vendor_interface.cc Open
lib_interface_->op
07-15 14:08:52.043    2583    4881 D [BT]        : mtk_bt_op: BT_VND_OP_POWER_CTRL 1
07-15 14:08:52.043    2583    4881 D [BT]        : mtk_bt_op: BT_VND_OP_USERIAL_OPEN


07-15 14:08:53.117    4841    4857 I bt_stack_manager: td: Stack_manager.cc event_start_up_stack finished
07-15 14:08:53.118    4841    4856 I AdapterState: BLE_ON : entered
07-15 14:08:53.118    4841    4856 D AdapterProperties: Setting state to BLE_ON
07-15 14:08:53.118    4841    4856 D BluetoothAdapterService: updateAdapterState() - Broadcasting state
BLE_ON to 1 receivers.
```

```
07-15 14:08:53.118    3368    3417 D BluetoothManagerService: MESSAGE_BLUETOOTH_STATE_CHANGE:
BLE_TURNING_ON > BLE_ON


07-15 14:08:53.121    3368    3417 E BluetoothManagerService: td: BluetoothManagerService.java doBind() enter.
07-15 14:08:53.124    4841    4841 D GattService: onBind
07-15 14:08:53.125    3368    3417 D BluetoothManagerService: Sending BLE State Change:
BLE_TURNING_ON > BLE_ON
07-15 14:08:53.126    3368    3368 D BluetoothManagerService: BluetoothServiceConnection:
com.android.bluetooth.gatt.GattService
07-15 14:08:53.127    3368    3417 D BluetoothManagerService:
MESSAGE_BLUETOOTH_SERVICE_CONNECTED: 2


07-15 14:08:53.128    3368    3417 D BluetoothManagerService: Persisting Bluetooth Setting: 1
07-15 14:08:53.128    4841    4856 I AdapterState: TURNING_ON : entered
07-15 14:08:53.128    4841    4856 D AdapterProperties: Setting state to TURNING_ON


07-15 14:08:53.129    4841    4856 E BluetoothAdapterService: td: AdapterService.java setProfileServiceState
state = 12
07-15 14:08:53.129    3368    3417 D BluetoothManagerService: MESSAGE_BLUETOOTH_STATE_CHANGE:
BLE_ON > TURNING_ON
07-15 14:08:53.129    3368    3417 D BluetoothManagerService: Sending BLE State Change: BLE_ON >
TURNING_ON


07-15 14:08:53.140    4841    4841 D A2dpService: onCreate
07-15 14:08:53.141    4841    4856 E BluetoothAdapterService: td: AdapterService.java setProfileServiceState
state = 12
07-15 14:08:53.141    4841    4841 I A2dpService: create()
07-15 14:08:53.141    4841    4841 D A2dpService: onBind
07-15 14:08:53.152    4841    4856 E BluetoothAdapterService: td: AdapterService.java setProfileServiceState
state = 12
```

```
07-15 14:08:53.152    3368    3368 D BluetoothA2dp: Proxy object connected

07-15 14:08:53.153    4841    4841 E A2dpService: td: ProfileService.java onStartCommand() enter

07-15 14:08:53.154    4841    4841 E A2dpService: td: profileService.java doStart() enter.

07-15 14:08:53.166    4841    4841 E A2dpService: td: profileService.java doStart() invoke start().

07-15 14:08:53.166    4841    4841 I A2dpService: start()

07-15 14:08:53.167    4841    4841 D BluetoothAdapterService: getAdapterService() - returning

com.android.bluetooth.btservice.AdapterService@a10ca8a

07-15 14:08:53.167    4841    4841 I BluetoothA2dpServiceJni: classInitNative: succeeds

07-15 14:08:53.197    4841    4841 I bt_btif_a2dp_source: btif_a2dp_source_init

07-15 14:08:53.198    4841    4895 I bt_btif_a2dp_source: btif_a2dp_source_init_delayed

07-15 14:08:53.198    4841    4885 I bt_bta_av: bta_av_api_register: AVRCP version used for sdp: "avrcp15"

07-15 14:08:53.199    4841    4841 D A2dpService: A2DP offload flag set to false

07-15 14:08:53.204    4841    4841 D A2dpService: setA2dpService(): set to:

com.android.bluetooth.a2dp.A2dpService@94055ea

07-15 14:08:53.205    4841    4841 D A2dpService: setActiveDevice(null): previous is null

07-15 14:08:53.205    4841    4841 D A2dpService: broadcastActiveDevice(null)

07-15 14:08:53.207    4841    4841 E BluetoothAdapterService: td: AdapterService.java

onProfileServiceStateChanged() send MESSAGE_PROFILE_SERVICE_STATE_CHANGED

07-15 14:08:53.208    4841    4841 D A2dpSinkService: onCreate

07-15 14:08:53.209    4841    4841 E A2dpSinkService: td: ProfileService.java onStartCommand() enter

07-15 14:08:53.210    4841    4841 E A2dpSinkService: td: profileService.java doStart() enter.

07-15 14:08:53.210    4841    4841 D BluetoothAdapterService: getAdapterService() - returning

com.android.bluetooth.btservice.AdapterService@a10ca8a

07-15 14:08:53.215    4841    4841 E A2dpSinkService: td: profileService.java doStart() invoke start().

07-15 14:08:53.215    4841    4841 D A2dpSinkService: start()

07-15 14:08:53.225    4841    4841 I BluetoothA2dpSinkServiceJni: classInitNative: succeeds

07-15 14:08:53.225    4841    4841 D A2dpSinkStateMachine: make


07-15 14:08:53.226    4841    4841 I bt_btif : get_profile_interface: id = a2dp_sink

07-15 14:08:53.226    4841    4841 I btif_av : bt_status_t BtifAvSink::Init(btav_sink_callbacks_t *)
```

```
07-15 14:08:53.226    4841    4841 I bt_btif_a2dp_sink: btif_a2dp_sink_init

07-15 14:08:53.227    4841    4885 I bt_bta_av: bta_av_api_register: AVRCP version used for sdp: "avrcp15"

07-15 14:08:53.228    4841    4841 D A2dpSinkService: setA2dpSinkService(): set to:

com.android.bluetooth.a2dpsink.A2dpSinkService@f6c5d90

07-15 14:08:53.228    4841    4896 D A2dpSinkStateMachine: Enter Disconnected: -2

07-15 14:08:53.228    4841    4841 E BluetoothAdapterService: td: AdapterService.java

onProfileServiceStateChanged() send MESSAGE_PROFILE_SERVICE_STATE_CHANGED

07-15 14:08:53.229    4841    4841 D BluetoothAdapterService: handleMessage() - Message: 2

07-15 14:08:53.230    4841    4841 D BluetoothAdapterService: handleMessage() -

MESSAGE_PROFILE_SERVICE_REGISTERED

07-15 14:08:53.230    4841    4864 D BluetoothActiveDeviceManager:

handleMessage(MESSAGE_ADAPTER_ACTION_STATE_CHANGED): newState=11

07-15 14:08:53.230    4841    4841 I BluetoothHidHostServiceJni: classInitNative: succeeds

07-15 14:08:53.231    4841    4841 D HidHostService: onCreate

07-15 14:08:53.232    4841    4841 D HidHostService: onBind

07-15 14:08:53.234    4841    4841 E HidHostService: td: profileService.java doStart() enter

07-15 14:08:53.237    4841    4841 E HidHostService: td: profileService.java doStart() invoke start().

07-15 14:08:53.238    4841    4841 D NewAvrcpTargetService: onCreate

07-15 14:08:53.239    4841    4841 E NewAvrcpTargetService: td: ProfileService.java onStartCommand() enter

07-15 14:08:53.240    4841    4841 E NewAvrcpTargetService: td: profileService.java doStart() enter.


07-15 14:08:53.242    4841    4841 I NewAvrcpTargetService: User unlocked, initializing the service

07-15 14:08:53.242    4841    4841 E NewAvrcpTargetService: td: profileService.java doStart() invoke start().

07-15 14:08:53.242    4841    4841 I NewAvrcpTargetService: Starting the AVRCP Target Service

07-15 14:08:53.246    4841    4841 V NewAvrcpMediaPlayerList: Creating MediaPlayerList

07-15 14:08:53.259    4841    4841 I NewAvrcpTargetJni: classInitNative: AvrcpTargetJni initialized!

07-15 14:08:53.259    4841    4841 D NewAvrcpNativeInterface: Init AvrcpNativeInterface

07-15 14:08:53.259    4841    4841 D NewAvrcpTargetJni: initNative


07-15 14:08:53.259    4841    4885 I bt_stack: [INFO:avrcp_service.cc(379)] AVRCP Target Service started
```

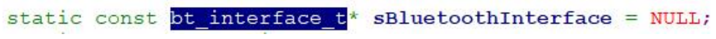
```
07-15 14:08:53.259    4841    4885 I bt_stack: [INFO:connection_handler.cc(198)] Connect to device
ff:ff:ff:ff:ff:ff
07-15 14:08:53.259    4841    4885 I bt_stack: [INFO:connection_handler.cc(219)] virtual bool
bluetooth::avrcp::ConnectionHandler::AvrcpConnect(bool, const RawAddress &): handle=0000 status= 000000
07-15 14:08:53.262    4841    4841 E BluetoothAdapterService: td: AdapterService.java
onProfileServiceStateChanged() send MESSAGE_PROFILE_SERVICE_STATE_CHANGED
07-15 14:08:53.264    4841    4841 I BluetoothAvrcpControllerJni: classInitNative: succeeds
07-15 14:08:53.265    4841    4841 D AvrcpControllerService: onCreate
07-15 14:08:53.266    4841    4841 E AvrcpControllerService: td: ProfileService.java onStartCommand() enter
07-15 14:08:53.267    4841    4841 E AvrcpControllerService: td: profileService.java doStart() enter.
07-15 14:08:53.267    4841    4841 D BluetoothAdapterService: getAdapterService() - returning
com.android.bluetooth.btservice.AdapterService@a10ca8a
07-15 14:08:53.273    4841    4841 E AvrcpControllerService: td: profileService.java doStart() invoke start().
07-15 14:08:53.279    4841    4841 E BluetoothAdapterService: td: AdapterService.java
onProfileServiceStateChanged() send MESSAGE_PROFILE_SERVICE_STATE_CHANGED
07-15 14:08:53.279    4841    4841 D BluetoothAdapterService: handleMessage() - Message: 1


07-15 14:08:53.305    4841    4841 E BluetoothAdapterService: td: AdapterService.java
processProfileServiceStateChanged BluetoothAdapter.STATE_ON
07-15 14:08:53.305    4841    4841 D AdapterProperties: onBluetoothReady, state=TURNING_ON, ScanMode=20
07-15 14:08:53.307    4841    4841 E BluetoothAdapterService: td: AdapterService.java
processProfileServiceStateChanged send AdapterState.BREDR_STARTED
07-15 14:08:53.307    4841    4856 I AdapterState: ON : entered
07-15 14:08:53.307    4841    4841 I BluetoothPhonePolicy: processProfileActiveDeviceChanged,
activeDevice=null, profile=2
07-15 14:08:53.307    4841    4856 D AdapterProperties: Setting state to ON
07-15 14:08:53.308    4841    4856 D BluetoothAdapterService: updateAdapterState() - Broadcasting state ON to 1
receivers.
07-15 14:08:53.308    3368    3417 D BluetoothManagerService: MESSAGE_BLUETOOTH_STATE_CHANGE:
TURNING_ON > ON
```

```
07-15 14:08:53.314    3368    3417 D BluetoothManagerService: Sending BLE State Change: TURNING_ON >
ON
```

Android
9.0蓝牙打开日志信

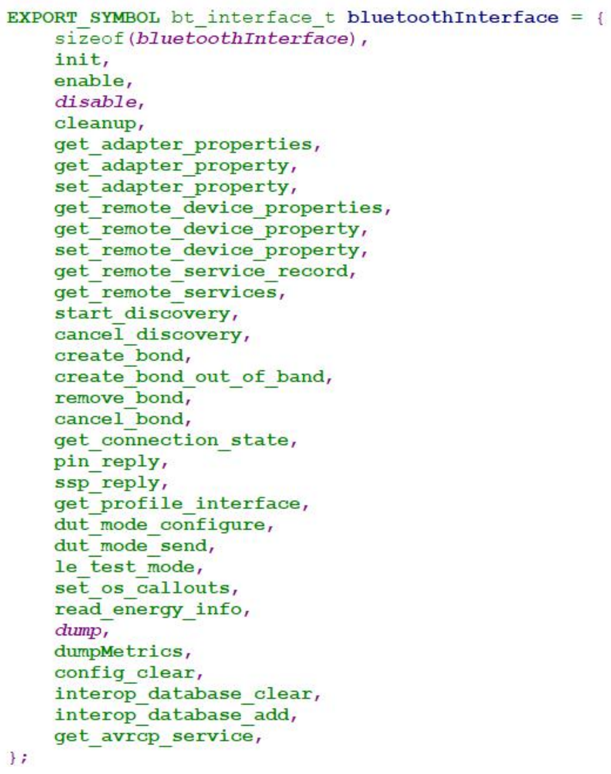
6. 其他关键方法

android\packages\apps\bluetooth\jni\com_android_bluetooth_btservice_AdapterService.cpp

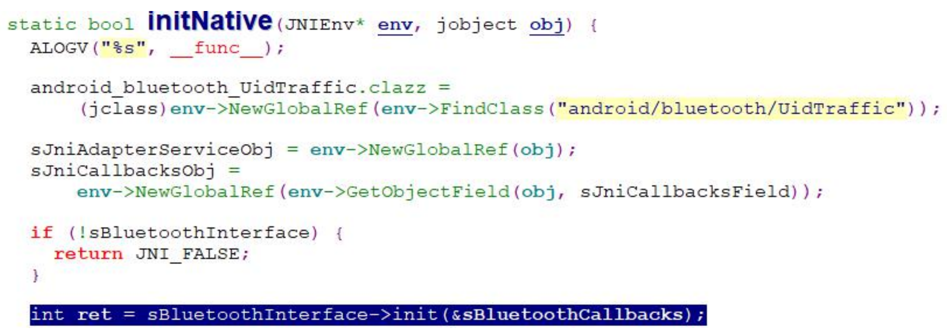


```
static const bt_interface_t* sBluetoothInterface = NULL;
```

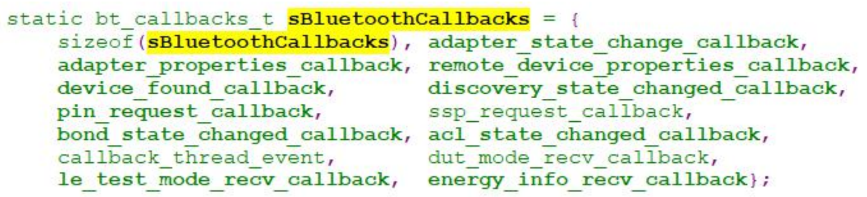
android\system\bt\btif\src\bluetooth.cc



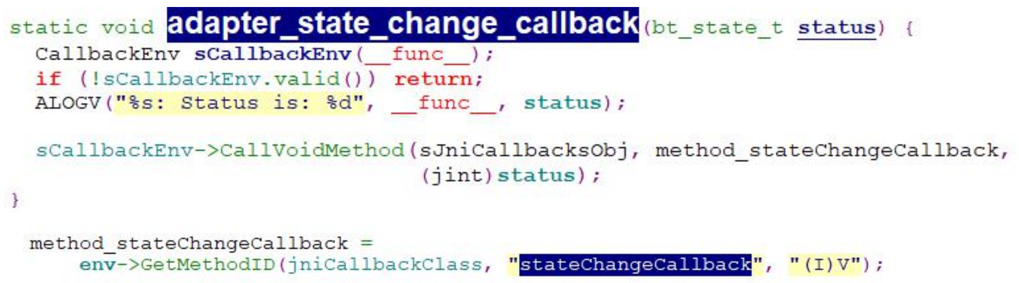
```
EXPORT_SYMBOL bt_interface_t bluetoothInterface = {
    sizeof(bluetoothInterface),
    init,
    enable,
    disable,
    cleanup,
    get_adapter_properties,
    get_adapter_property,
    set_adapter_property,
    get_remote_device_properties,
    get_remote_device_property,
    set_remote_device_property,
    get_remote_service_record,
    get_remote_services,
    start_discovery,
    cancel_discovery,
    create_bond,
    create_bond_out_of_band,
    remove_bond,
    cancel_bond,
    get_connection_state,
    pin_reply,
    ssp_reply,
    get_profile_interface,
    dut_mode_configure,
    dut_mode_send,
    le_test_mode,
    set_os_callouts,
    read_energy_info,
    dump,
    dumpMetrics,
    config_clear,
    interop_database_clear,
    interop_database_add,
    get_avrcp_service,
};
```



```
static bool initNative(JNIEnv* env, jobject obj) {
  ALOGV("%s", __func__);

  android_bluetooth_UidTraffic.clazz =
      (jclass)env->NewGlobalRef(env->FindClass("android/bluetooth/UidTraffic"));

  sJniAdapterServiceObj = env->NewGlobalRef(obj);
  sJniCallbacksObj =
      env->NewGlobalRef(env->GetObjectField(obj, sJniCallbacksField));

  if (!sBluetoothInterface) {
    return JNI_FALSE;
  }

  int ret = sBluetoothInterface->init(&sBluetoothCallbacks);
```

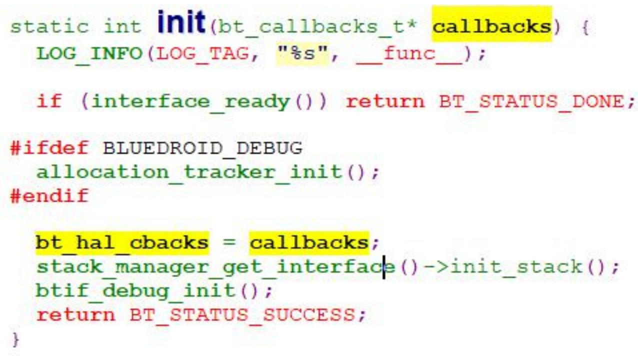


```
static bt_callbacks_t sBluetoothCallbacks = {
    sizeof(sBluetoothCallbacks), adapter_state_change_callback,
    adapter_properties_callback, remote_device_properties_callback,
    device_found_callback,       discovery_state_changed_callback,
    pin_request_callback,        ssp_request_callback,
    bond_state_changed_callback, acl_state_changed_callback,
    callback_thread_event,       dut_mode_recv_callback,
    le_test_mode_recv_callback,  energy_info_recv_callback};
```



```
static void adapter_state_change_callback(bt_state_t status) {
  CallbackEnv sCallbackEnv(__func__);
  if (!sCallbackEnv.valid()) return;
  ALOGV("%s: Status is: %d", __func__, status);

  sCallbackEnv->CallVoidMethod(sJniCallbacksObj, method_stateChangeCallback,
                               (jint)status);
}

  method_stateChangeCallback =
      env->GetMethodID(jniCallbackClass, "stateChangeCallback", "(I)V");
```

initNative 函数的具体实现，通过 bt_interface_t 结构体，调用到 C 中的 init 函数实现。同时传入 sBluetoothCallbacks 回调函数结构体。这个函数结构体比较重要，底层的状态变化都是通过这个回调函数结构体中的函数实现。



```
static int init(bt_callbacks_t* callbacks) {
  LOG_INFO(LOG_TAG, "%s", __func__);

  if (interface_ready()) return BT_STATUS_DONE;

#ifdef BLUEDROID_DEBUG
  allocation_tracker_init();
#endif

  bt_hal_cbacks = callbacks;
  stack_manager_get_interface()->init_stack();
  btif_debug_init();
  return BT_STATUS_SUCCESS;
}
```