BluetoothSettings,java 类 ——》》 onCreateOptionsMenu()创建bluetooth settings 的菜单一级界面

*onCreateOptionsMenu () --- 》》*onOptionsItemSelected（）处理菜单的点击逻辑

UI ： 蓝牙配对

/packages/apps/Settings/src/com/android/settings/bluetooth/DeviceListPreferenceFragment.java中。

DeviceListPreferenceFragment是蓝牙扫描到的设备列表，点击其中一个蓝牙设备，调用onPreferenceTreeClick方法开始蓝牙的配对过程。

onDevicePreferenceClick方法中调用/packages/apps/Settings/src/com/android/settings/bluetooth/BluetoothDevicePreference.java的onClicked方法：

调用的是本地的pair方法：

pair方法会调用/frameworks/base/packages/SettingsLib/src/com/android/settingslib/bluetooth/CachedBluetoothDevice.java中的startPairing，启动配对

startPairing方法会调用createBond

createBond调用/frameworks/base/core/java/android/bluetooth/BluetoothDevice.java  
中的createBond方法：

createBond接着调用IBluetooth的createBond方法，通过aidl方式调用蓝牙远程服务。

(IBluetooth接口的类是AdapterServiceBinder，AdapterServiceBinder实现IBluetooth.Stub接口) 是/packages/apps/Bluetooth/src/com/android/bluetooth/btservice/AdapterService的私有内部类，AdapterServiceBinder收到的操作，都会转交AdapterService处理，所以会调用AdapterService的createBond方法。

AdapterService的createBond方法将配对任务转交mBondStateMachine，由状态机处理该信息

BondStateMachine处理服务发送过来的BondStateMachine.CREATE\_BOND消息 ，在processMessage 中调用 BondStateMachine的createBond 方法:

createBondNative方法实现在/packages/apps/Bluetooth/jni/com\_android\_bluetooth\_btservice\_AdapterService.cpp中：

createBondNative方法调用：

create\_bond这个方法调用到了蓝牙协议栈里面

## 蓝牙协议栈

create\_bond方法位于/system/bt/btif/src/bluetooth.cc：

create\_bond方法调用/system/bt/btif/src/btif\_dm.cc的btif\_dm\_create\_bond方法：

btif\_dm\_create\_bond方法最终调用了本地的btif\_dm\_generic\_evt方法，传入BTIF\_DM\_CB\_CREATE\_BOND事件：

调用本地的btif\_dm\_cb\_create\_bond方法

(btif\_dm\_cb\_create\_bond方法第一次调用会调用)BTA\_DmBondByTransport方法位于\system\bt\bta\dm\bta\_dm\_api.c：

通过do\_in\_bta\_thread调用/system/bt/bta/dm/bta\_dm\_act.cc里面的bta\_dm\_bond方法，进入bta进程：

然后来到\system\bt\stack\btm\btm\_sec.c的BTM\_SecBondByTransport 方法：

调用本地btm\_sec\_bond\_by\_transport方法

调用system/bt/stack/hcic/hcicmds.cc的btsnd\_hcic\_write\_pin\_type方法通过HCI向底层发送命令进行控制

通过和hci层的通信，host告诉controlor蓝牙地址、数据、命令等，从而控制其底层硬件发起配对操作

## 配对状态改变的回传

bta里面调用/system/bt/bta/dm/bta\_dm\_act.cc里面的bta\_dm\_bond方法，进行配对，这个方法里面有这样一段代码：

bta\_dm\_cb.p\_sec\_cback(BTA\_DM\_AUTH\_CMPL\_EVT, &sec\_event);

BTA\_DM\_AUTH\_CMPL\_EVT，根据这个事件标志，我们找到了 /system/bt/btif/src/btif\_dm.cc里面的btif\_dm\_upstreams\_evt方法

论配对成功还是失败，这里都会用 bond\_state\_changed这个方法进行处理

通过HAL\_CBACK(bt\_hal\_cbacks, bond\_state\_changed\_cb, status, &tmp, state);这样的方法进行回调的，bond\_state\_changed\_cb这个函数在bluetooth.h被定义对应的是com\_android\_bluetooth\_btservice\_AdapterService.cpp里的bond\_state\_changed\_callback

sCallbackEnv->CallVoidMethod(sJniCallbacksObj, method\_bondStateChangeCallback, (jint)status, addr.get(), (jint)state);

这里将bondStateChangeCallback方法对应到jni的method\_bondStateChangeCallback方法

jclass jniCallbackClass = env->FindClass("com/android/bluetooth/btservice/JniCallbacks");

........................省略................................

method\_bondStateChangeCallback =

env->GetMethodID(jniCallbackClass, "bondStateChangeCallback", "(I[BI)V");

就找到了JniCallbacks.java里面的bondStateChangeCallback方法

接下来便进入了/packages/apps/Bluetooth/src/com/android/bluetooth/btservice/BondStateMachine.java状态机里面：

状态机里面通过sendMessage进行配对状态的变更