**1**、 first in the scanning activity of the **onCreate ()** inside the initialization of mobile phone.

//（This statement is used to determine whether the device supports low power Bluetooth）

**（1）if** (!getPackageManager().hasSystemFeature(PackageManager.

***FEATURE\_BLUETOOTH\_LE***)) {

Toast.*makeText*(**this**, R.string.***ble\_not\_supported***, Toast.***LENGTH\_SHORT***).show();

finish();

}

//（The statement is used to obtain the BluetoothAdapter object for the back of the scanning device；mBluetoothAdapter.stopLeScan(mLeScanCallback)）

**（2）**BluetoothManager bluetoothManager = (BluetoothManager) getSystemService(Context.***BLUETOOTH\_SERVICE***);

BluetoothAdapter mBluetoothAdapter = bluetoothManager.getAdapter();

//（The statement is used to determine whether the device supports Bluetooth）

**（3）if** (mBluetoothAdapter == **null**) {

Toast.*makeText*(**this**, R.string.***error\_bluetooth\_not\_supported***, Toast.***LENGTH\_SHORT***).show();

finish();

**return**;

}

**2、**In the **onResume ()** method, the following statement is used to call the Bluetooth

**if** (!mBluetoothAdapter.isEnabled()) {

**if** (!mBluetoothAdapter.isEnabled()) {

Intent enableBtIntent = **new** Intent(BluetoothAdapter.***ACTION\_REQUEST\_ENABLE***);

startActivityForResult(enableBtIntent, ***REQUEST\_ENABLE\_BT***);

}

}

**3、**starting to scan the Bluetooth device, it should be used **mBluetoothAdapter.startLeScan (mLeScanCallback)**;to stop the scan using **mBluetoothAdapter.stopLeScan(mLeScanCallback)**, in which the parameters of the mLeScanCallback is scanned to the back of the Bluetooth, we get through **mLeScanCallback = new BluetoothAdapter.LeScanCallback（）.**

**4、**In the scan correction, we want to be scanned into the Bluetooth device into a **iBeacon** object, which contains the devicename, major, minor, UUID, bluetoothAddress, RSSI and other basic properties，The following：

**private** BluetoothAdapter.LeScanCallback mLeScanCallback = **new** BluetoothAdapter.LeScanCallback() {

@Override

**public** **void** onLeScan(**final** BluetoothDevice device, **int** rssi, **byte**[] scanRecord) {

**final** iBeacon ibeacon = iBeaconClass.*fromScanData*(device, rssi, scanRecord);

runOnUiThread(**new** Runnable() {

@Override

**public** **void** run() {

mLeDeviceListAdapter.addDevice(ibeacon);

mLeDeviceListAdapter.notifyDataSetChanged();

}

});

}

};

**5、**Please bind [**com**](eclipse-javadoc:%E2%98%82=iBeaconSDK/src%3Ccom)**.**[**axaet**](eclipse-javadoc:%E2%98%82=iBeaconSDK/src%3Ccom.axaet)**.**[**ibeaconsdk**](eclipse-javadoc:%E2%98%82=iBeaconSDK/src%3Ccom.axaet.ibeaconsdk)**.BluetoothLeService** to use related solutions before bluetooth connection.

**5.1.** Please bind service and set mBluetoothLeService getting as example.

Intent gattServiceIntent = new Intent(this, BluetoothLeService.class);

bindService(gattServiceIntent, mServiceConnection, *BIND\_AUTO\_CREATE*);

private final ServiceConnection mServiceConnection = new ServiceConnection() {

public void onServiceConnected(ComponentName componentName,IBinder service) {

mBluetoothLeService = ((BluetoothLeService.LocalBinder) service).getService();

if (!mBluetoothLeService.initialize()) {

finish();

}

//Bluetooth device connection

mBluetoothLeService.connect(mDeviceAddress);

}

public void onServiceDisconnected(ComponentName componentName) {

mBluetoothLeService = null;

}

};

**5.2.** Please use mBluetoothLeService.connect(address) to connect bluetooth device. Please register BroadcastReceiver and receive information to examine if successful connect.

**5.3.** Please register BroadcastReceiver, registerReceiver(mGattUpdateReceiver, BluetoothLeService.*makeGattUpdateIntentFilter*()); second parameter IntenFilter can be obtained through BluetoothLeService.*makeGattUpdateIntentFilter*()

**5.4.** OnReceive method in BroadcastReceiver, action may be get as following. No receive if action is useless:

**（1）.** BluetoothService.ACTION\_GATT\_CONNECTED (It will receive this broadcast after Ble device successful connect)

**（2）.** BluetoothService.ACTION\_GATT\_DISCONNECTED (It will receive this broadcast when Ble device failed connect or disconnect)

**（3）.** BluetoothService.ACTION\_GATT\_SERVICES\_DISCOVERED (It will receive this broadcast when bluetooth services is found. Rarely use)

**（4）.** BluetoothLeService.ACTION\_EXTRA\_UUID\_DATA (After connection, UUID will be received through this broadcast. Please get it by intent.getStringExtra(BluetoothLeService.*UUID\_DATA*). Return value is string class. )

**（5）.** BluetoothLeService.ACTION\_EXTRA\_OTHER\_DATA (After connection, Major, Minor, Period, TX Power etc will be received through this broadcast. )

int Major = intent.getIntExtra(BluetoothLeService.*MAJOR\_DATA*,-1);

int Minor = intent.getIntExtra(BluetoothLeService.*MINOR\_DATA*, -1);

int Period = intent.getIntExtra(BluetoothLeService.*PERIOD\_DATA*, -1);

int txPower = intent.getIntExtra(BluetoothLeService.*TXPOWER\_DATA*, -1);

**（6）.** BluetoothLeService.ACTION\_EXTRA\_SENIOR\_DATA (To distinguish BluetoothLeService.ACTION\_EXTRA\_OTHER\_DATA above, only VIP can receive this broadcast. After broadcast received, no password needed to modify parameter. Acquisition method is same to Action 5. Rarely use in the development for BLE product upgrades. This broadcast is optional to no receive.)

**（7）.** BluetoothLeService.ACTION\_EXTRA\_PASSWORD\_MISTAKE (Receiving this broadcast, indicates wrong password is entered and parameter will not able to modify. )

**（8）.** BluetoothLeService.ACTION\_EXTRA\_PASSWORD\_RIGHT (Receiving this broadcast, indicates wrong password is entered and parameter is able to modify. )

**5.5.** After mBluetoothLeService.connect(address) programming, broadcast of BluetoothService.ACTION\_GATT\_CONNECTED will be received if successful connect. Then various kinds of operation can be performed. BluetoothLeService.ACTION\_GATT\_DISCONNECTED will be received if connection failed.

**5.6.** Parameter(UUID, Major, Minor, Period, TX power) can be obtained after successful connection. Parameter can be modified as following:

**（1）.** We all need to enter a password for authentication before we make it. Use validatePassword(String password), Receive broadcast by certification BluetoothLeService.ACTION\_EXTRA\_PASSWORD\_RIGHT，Otherwise received BluetoothLeService.ACTION\_EXTRA\_PASSWORD\_MISTAKE

**（2）.** Please use mBluetoothLeService**.** modifyUUID(String uuid) to modify UUID. UUID must be 0~9 or a~f or A~F;

**（3）.** Please use mBluetoothLeService**.** modifyOtherParameter(String major, String minor, String period, int txPower) to modify other parameters(String major, String minor, String period, int txPower). Major and Minor must less than 65535; TX Power can only choose this four value(0, 1, 2, 3). Generally, TX Power is 0 when Spinner is -23; TX Power is 1 when Spinner is -6; TX Power is 2 when Spinner is 0; TX Power is 3 when Spinner is 4.

**（4）.** Method of mBluetoothLeService**.** modifyDeviceName (String deviceName), parameter deviceName must be visible character ascii code; Please use mBluetoothLeService**.**closeBLE() to reset Bluetooth terminal after modification.

**（5）.**changing the password，using modifyNewPassword(String oldPassword, String newPassword), ASCII code visible character, The length of the two parameters must be equal to 6

**5.7.** Attention: If keep fail connect to the bluetooth terminal during connection, please use mBluetoothLeService **.**disconnect() and mBluetoothLeService **.**close() to release resources when disconnect. Otherwise, it will block the connect of other devices.

**5.8.** If you do not understand, the specific details of the use of a reference to our code sample.