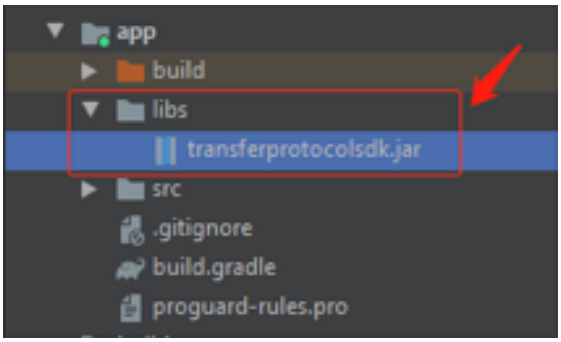
TransferProtocolSDK 통합 가이드

1. 디바이스 연결 및 UUID 사용자 정의 필요

public static final String SERVICE_UUID = "0000ff01-0000-1000-8000-00805f9b34fb";public static final String CHARACTERISTIC_WRITE_UUID = "0000ff02-0000-1000-8000-00805f9b34fb";public static final String CHARACTERISTIC_READ_UUID = "0000ff10-0000-1000-8000-00805f9b34fb";public static final String DESCRIPTOR_UUID = "00002902-0000-1000-8000-00805f9b34fb";

2. SDK 사용 방법

1. libs 디렉터리에 transferprotocolsdk.jar 파일을 추가합니다.



1. AndroidManifest.xml 에 다음 권한을 추가해야 하며, 동적 권한은 별도로 요청해야 합니다

```
<uses-permission
android:name="android.permission.BLUETOOTH" />
<uses-permission
android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

1. 앱의 Application 클래스에서 BleTransferManager를 초기화합니다:

```
public static MainApplication mainApplication;
public static BleTransferManager manager;
public static MainApplication getInstance() {
    return mainApplication;
}

@Override
public void onCreate() {
    super.onCreate();
    mainApplication = this;
    manager = BleTransferManager×initialized(this);
```

}

1. UI Activity 또는 Fragment의 생명주기 메서드에서 다음 콜백을 설정하세요. 필요 시 Application에서 설정해도 됩니다:

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.*activity_main***);**

***manager*.setBTMGattCallBack(this);//블루투스 상태 콜백 설정**
***manager*.setAnalyticalDataCallBack(this);// 블루투스 데이터 전송 콜백 설정**

1. 连接Gatt

***manager*.connectGatt(macAddress, false); // 매개변수1: MAC 주소, 매개변수2: 페어링 여부**

1. 断开Gatt

***manager*.disconnectGatt(macAddress, false); // 매개변수1: MAC 주소, 매개변수2: 페어링 해제 여부 (페어링 해제는 리플렉션으로 숨겨진 API를 호출하는 방식으로, 실패할 수도 있음)**

1. 커스텀 블루투스 Gatt 객체를 사용해 연결하려면 다음을 사용.

***manager*.setBluetoothGatt(BluetoothGatt *bluetoothGatt*);**

1. 데이터 송수신에 관한 더 자세한 내용은 API 문서 및 예제 Demo의 MainActivity 코드를 참고하세요.

--> End

<MainActivity.java>

```
package com.smtlink.transferprotocoldemo;

import static com.smtlink.transferprotocoldemo.MainApplication.manager;

import android.Manifest;
import android.annotation.SuppressLint;
import android.bluetooth.BluetoothGatt;
import android.bluetooth.BluetoothProfile;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
```

```
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.text.TextUtils;
import android.text.method.ScrollingMovementMethod;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.chad.library.adapter.base.BaseQuickAdapter;
import com.chad.library.adapter.base.listener.OnItemChildClickListener;
import com.smtlink.transferprotocoldemo.scannerble.BLEDeviceInfo;
import com.smtlink.transferprotocoldemo.scannerble.ScanningActivity;
import com.smtlink.transferprotocolsdk.Protocols;
import com.smtlink.transferprotocolsdk.bean.AlarmInfo;
import com.smtlink.transferprotocolsdk.bean.ContactsInfo;
import com.smtlink.transferprotocolsdk.bean.ThemeRemindInfo;
import com.smtlink.transferprotocolsdk.ble.AnalyticalDataCallBack;
import com.smtlink.transferprotocolsdk.ble.BTMGattCallBack;
import com.smtlink.transferprotocolsdk.types.MessageType;
import com.smtlink.transferprotocolsdk.utils.BluetoothOpenStateUtil;
import com.smtlink.transferprotocolsdk.utils.SimpleDateFormatUtil;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.io.InputStream;
import java.lang.ref.WeakReference;
import java.nio.charset.StandardCharsets;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Locale;
```

```
public class MainActivity extends AppCompatActivity implements
View.OnClickListener, BTMGattCallBack, AnalyticalDataCallBack {

    public final static int MSG_DISCONNECT = 0;
    public final static int MSG_CONNECTED = 1;
    public final static int MSG_JSON_DATA = 2;
    public final static int MSG_LOW_BATTERY = 3;

    private TextView mMacAddress;
    private Button mConnectAndState, mClearMac;
    private TextView mTransferData;
    private RecyclerView mRecyclerView;
    private TextView mPushState;

    private String address = "10:21:23:C0:02:7F";//00:22:BB:34:26:54,
    private String bleName = "";

    private int isFind = 1;

    private String[] permissions =
{Manifest.permission.ACCESS_FINE_LOCATION};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        manager.setBTMGattCallBack(this);//设置蓝牙状态回调
        manager.setAnalyticalDataCallBack(this);//设置蓝牙数据传输回调

        //获取权限
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S)
            permissions = new String[]
{Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.BLUETOOTH_SCAN,
Manifest.permission.BLUETOOTH_CONNECT};
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED
        || ContextCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH_CONNECT) !=
PackageManager.PERMISSION_GRANTED) {
            requestPermissions(permissions, 100);
        }

        initView();
```

```
    }

    @Override
    protected void onStart() {
        super.onStart();
        if (!BluetoothOpenStateUtil.isBluetoothOpen()) {//检查蓝牙开关状态
            BluetoothOpenStateUtil.openBluetooth(this);
        }

        if (!MainApplication.getInstance().isConnectedState()) {
            if (!TextUtils.isEmpty(mMacAddress.getText().toString().trim()))
                setState(false, Color.BLACK, "连接");
            else setState(false, Color.BLACK, "扫描");
        }
    }

    private void initView() {
        mMacAddress = (TextView) findViewById(R.id.macAddress);
        mMacAddress.setText(address);
        mConnectAndState = (Button) findViewById(R.id.connectAndState);
        mConnectAndState.setOnClickListener(this);
        mClearMac = (Button) findViewById(R.id.clearMac);
        mClearMac.setOnClickListener(this);
        mTransferData = (TextView) findViewById(R.id.transfer_data);

mTransferData.setMovementMethod(ScrollingMovementMethod.getInstance());
        mTransferData.setScrollbarFadingEnabled(false);//一直显示滚动条
        mPushState = (TextView) findViewById(R.id.push_state);

        mRecyclerView = (RecyclerView) findViewById(R.id.recyclerView);
        mRecyclerView.setLayoutManager(new GridLayoutManager(this, 4));

        MainButtonAdapter adapter = new
MainButtonAdapter(R.layout.item_activity_main_button,
ProtocolsHeadUtil.HEADS);
        adapter.setOnItemChildClickListener(onItemChildClickListener);
        adapter.addChildClickViewIds(R.id.button);
        mRecyclerView.setAdapter(adapter);
    }

    private void setState(boolean connectedState, int color, String text) {
        MainApplication.getInstance().setConnectedState(connectedState);
        mConnectAndState.setTextColor(color);
        mConnectAndState.setText(text);
    }

    @Override
```

```
    public void onClick(final View v) {
        if (v == mConnectAndState) {
            if (manager == null) {
                Log.e("gy", "MainActivity onClick manager == null");
                return;
            }
            mPushState.setVisibility(View.GONE);
            if (!MainApplication.getInstance().isConnectedState()) {//未连接, 连接
                String mac = mMacAddress×getText()×toString()×trim();
                if (!TextUtils.isEmpty(mac)) {
                    mConnectAndState.setText("连接中...");
                    manager.connectGatt(mac, false);
                    return;
                }
                Intent intent = new Intent(this, ScanningActivity.class);
                startActivityForResult(intent, 200);
            } else {//已连接, 断开
                manager.disconnectGatt(address, true);
                setState(false, Color.BLACK, "连接");
            }
        } else if (v == mClearMac) {
            if (MainApplication.getInstance().isConnectedState()) {
                Toast.makeText(this, "请先断开连接", Toast.LENGTH_SHORT).show();
                return;
            }
            mMacAddress.setText("");
        }
    }

    @Override
    protected void onActivityResult(final int requestCode, final int resultCode,
@Nullable final Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == RESULT_OK) {
            if (requestCode == 200) {
                if (data != null) {
                    Bundle bundle = data×getExtras();
                    if (bundle != null) {
                        BLEDeviceInfo parcelable =
bundle×getParcelable(ScanningActivity.EXTRA_STRING);
                        if (parcelable != null) {
                            Log.d("gy", "扫描的 Scanning MAC： " + parcelable.address +
" , name： " + parcelable.name);
                            Log.d("gy", "扫描的 Scanning name： " + parcelable.name);
                            bleName = parcelable.name;
                            //英文字母转大写, 否则非法. 然后去连接
                            address = parcelable×address×toUpperCase();
```

```
                    manager.connectGatt(address, false);
                    new Handler().postDelayed(new Runnable() {
                        @Override
                        public void run() {
                            mConnectAndState.setText("连接中...");
                            mMacAddress.setText(address);
                        }
                    }, 200);
                }
            }
        }
    }
}

    private final OnItemChildClickListener onItemChildClickListener = new
OnItemChildClickListener() {
        @Override
        public void onItemChildClick(@NonNull final BaseQuickAdapter<?, ?>
adapter, @NonNull final View view, final int position) {
            Log.i("gy", "MainActivity OnItemChildClickListener cmd: " +
ProtocolsHeadUtil.HEADS.get(position));

            if (manager == null) {
                Log.e("gy", "MainActivity OnItemChildClickListener
BleTransferManager == null");
                return;
            }
            if (!MainApplication.getInstance().isConnectedState()) {//未连接
                Toast.makeText(view.getContext(), "未连接设备！ ",
Toast.LENGTH_SHORT).show();
                return;
            }

            switch (ProtocolsHeadUtil.HEADS.get(position)) {
                case Protocols.GET0://获取设备详细数据
                    manager.cmdGet0();
                    break;
                case Protocols.GET10://获取每天计步总数据
                    manager.cmdGet10();
                    break;
                case Protocols.GET11://获取当天时间段增量(详细)数据, 设备每十分钟记
录一次
                    //方式1, 传一个Date
                    manager.cmdGet11(new Date());
                    //方式2, 传字符串日期和对应的格式化格式
                    //manager.cmdGet11("2022-03-11", "yyyy-MM-dd");
                    break;
```

```
            case Protocols.GET12://获取每天睡眠总数据
                manager.cmdGet12();
                break;
            case Protocols.GET13://获取当天时间段(详细)睡眠数据
                manager.cmdGet13();
                break;
            case Protocols.GET14://获取单次心率测试数据
                manager.cmdGet14();
                break;
            case Protocols.GET15://获取闹钟信息
                manager.cmdGet15();
                break;
            case Protocols.GET16://获取motion开关设置
                manager.cmdGet16();
                break;
            case Protocols.GET17://获取当前心率,睡眠,步数
                manager.cmdGet17();
                break;
            case Protocols.GET18://获取当前步数,距离,卡路里与send,10数据一样
                manager.cmdGet18();
                break;
            case Protocols.GET21://获取时间段睡眠数据(SR08)
                manager.cmdGet21();
                break;
            case Protocols.GET22://获取单次血压测试数据
                manager.cmdGet22();
                break;
            case Protocols.GET23://获取单次血氧测试数据
                manager.cmdGet23();
                break;
            case Protocols.GET61://获取计步历史数据的日期信息
                manager.cmdGet61();
                break;
            case Protocols.GET64://获取运动历史数据(暂未使用)
                //manager.cmdGet64();
                break;
            case Protocols.GET66://询问蓝牙设备连接状态
                manager.cmdGet66();
                break;
            case Protocols.GET69://获取首页气压数据(需要设备有air pressure
Sensor)
                manager.cmdGet69();
                break;
            case Protocols.GET70://获取首页潜水数据(需要设备有GPS Sensor)
                /**
                 * 1. 参数num 使用cmdGet65()获取的数据中运动模式为 潜水
的"serial_number"值
```

```
                 * 2. 直接 num = 100 为要最新的一次
                 */
                manager.cmdGet70(100);
                break;
            case Protocols.GET71://获取首页钓鱼数据(需要设备有GPS和air pressure
Sensor)
                /**
                 * 1. 参数num 使用cmdGet65()获取的数据中运动模式为 钓鱼
的"serial_number"值
                 * 2. 直接 num = 100 为要最新的一次
                 */
                manager.cmdGet71(100);
                break;
            case Protocols.GET72://获取联系人数据
                manager.cmdGet72();
                break;
            case Protocols.GET73://获取已推送的所有表盘序号
                manager.cmdGet73();
                break;
            case Protocols.GET74://获取已推送的所有主题序号
                manager.cmdGet74();
                break;
            case Protocols.GET75://获取已推送的所有应用序号
                manager.cmdGet75();
                break;
            case Protocols.GET77://开始测量心率
                manager.cmdGet77();
                break;
            case Protocols.GET78://获取设备血糖建模天数(状态)YZ01
                manager.cmdGet78();
                break;
            case Protocols.GET79://开始测量血糖
//                manager.cmdGet79();//YZ01使用

                /*
                 * SR08使用
                 * 第一个参数为餐前血糖值，第二个参数为餐后血糖值
                 * 例如：餐前血糖值为3.9，乘以10取整为39
                 *      餐后血糖值为8.0，乘以10取整为80
                 * 备注：测量返回的值也是整数，除以10即为实际值
                 */
                manager.cmdGet79(39, 80);
                break;
            case Protocols.GET80://获取设备半小时一次存储的心率YZ01
                try {
                    //取某天0点整以 秒 为单位的时间戳格式化字符串 例如：2023-07-15
00:00:00（时分秒默认为00）
```

```
                    long date =
SimpleDateFormatUtil.Y_M_dHHmmss().parse("2023-7-15" + "
00:00:00").getTime() / 1000;
                    manager.cmdGet80(date);
                } catch (ParseException e) {
                    e.printStackTrace();
                }
                break;
            case Protocols.GET81://开始测量血氧
                manager.cmdGet81();
                break;
            case Protocols.GET82://开始测量血压
                manager.cmdGet82();
                break;
            case Protocols.GET84://获取心率和血氧(BX03设备)
                manager.cmdGet84();
                break;
            case Protocols.GET85://开始测量体温
                manager.cmdGet85();
                break;
            case Protocols.GET86://获取设备每半小时存储的体温
                manager.cmdGet86();
                break;
            case Protocols.GET87://获取健康监测数据（心率、血氧、血压、血糖、
体温）
                manager.cmdGet87();
                break;
            case Protocols.GET88://获取充电状态
                manager.cmdGet88();
                break;
            case Protocols.GET89://获取GSensor
                manager.cmdGet89();
                break;
            case Protocols.GET90://获取健康监测开关状态
                manager.cmdGet90();
                break;
            case ProtocolsHeadUtil.GET_GROUP_GPS://获取多项运动历史数据(需要
设备有GPS Sensor, 无忽略)
                manager.cmdGet65();//数据获取一般需要2秒左右，可以在自定义列表界
面加个下拉刷新效果，等获取到数据后关闭。
                break;
            case ProtocolsHeadUtil.GET_SINGLE_GPS://获取对应的一次单项运动
GPS数据(需要设备有GPS Sensor)
                manager.cmdGet101(9);//9是cmdGet65()获取的数据中其中一
个"serial_number"值
                //这里数据获取的时长不定，数据越多时间越长。可以加个显示获取进度
的弹窗，在getGpsDataProgress()回调方法中获取进度百分比
                break;
```

```
//////////////////////////////////////////////////

            case Protocols.SET10://设置个人信息
                //依次为: 目标步数,性别(1男,0女),身高,体重,年龄
                manager.cmdSet10(5000, 1, 175, 60, 25);//方式1
                //manager.cmdSet10("5000", "1", "175", "60", "25");//方式2
                break;
            case Protocols.SET11://设置睡眠信息 -- 暂不支持
                //manager.cmdSet11("22:30", "7:30");//开始时间,结束时间
                break;
            case Protocols.SET12://设置久坐提醒信息
                manager.cmdSet12(30);//范围为0到90分钟，以(time_long+=30)阶梯变化，0表示关闭
                break;
            case Protocols.SET13://设置闹钟(W1项目不支持删除,固定3个闹钟)
                List<AlarmInfo> alarmInfos = new ArrayList<>();
                AlarmInfo alarmInfo = new AlarmInfo();
                alarmInfo.time = "12:00";//闹钟时间
                alarmInfo.selected_days_week = "0111110";//共7个字符，周日和周六为0不响铃, 周一到周五为1响铃
                alarmInfo.ring = "0";//铃声 0,1,2,3,4 共5个
                alarmInfo.ring_type = "0";//0仅响铃, 1仅震动, 2响铃与震动
                alarmInfo.alarm_on_off = "1";//闹钟开启状态, 0不开启, 1开启
                alarmInfos.add(alarmInfo);
                manager.cmdSet13(alarmInfos);
                /*
                 * 注意: 设置闹钟信息前先 cmdGet15() 获取同步一下设备中保存的闹钟信息,
                 * 并添加到 alarmInfos 中和新的闹钟信息一起发送给设备, 否则设备会覆盖所有闹钟信息, 只保留当前设置的闹钟信息
                 */
                break;
            case Protocols.SET15://设置APK是否前/后台运行状态
                manager.cmdSet15(1);//1表示APP在前台, 0表示APP在后台
                break;
            case Protocols.SET19://设置喝水提醒
                manager.cmdSet19(1);//1开启喝水提醒, 0关闭
                break;
            case Protocols.SET20://设置手机找手表
                if (isFind == 1) {
                    manager.cmdSet20(1);//1开始查找, 0停止查找
                    isFind = 0;
                } else {
                    manager.cmdSet20(0);
                    isFind = 1;
                }
```

```
                break;
            case Protocols.SET21://设置motion 闹钟静音、抬手亮屏、来电拒接 (W1
项目不用这条协议)
                manager.cmdSet21(1, 1, 1);//1打开, 0关闭
                break;
            case Protocols.SET30://设置女性信息  设置生理期或预产期会互相覆盖
                //设置生理期为: 经期持续7天, 经期间隔28天, 经期开始的时间date转换的
1646755200秒 注意：date默认获取的是毫秒，需要除以1000
                manager.cmdSet30("7", "28", 1646755200);
                //设置预产期为: 计算的预产期时间date转换的1646755200秒
                //manager.cmdSet30(1646755200);
                break;
            case Protocols.SET44://设置语言
                manager.cmdSet44(Locale.getDefault().toLanguageTag());
                //manager.cmdSet44(null);//传null为默认系统语言
                break;
            case Protocols.SET45://设置时间
                manager.cmdSet45(MainActivity.this);
                break;
            case Protocols.SET46://设置单位制式
                manager.cmdSet46(0);// 0公制, 1英制
                break;
            case Protocols.SET68://设置天气
                //详细请参考文档：“协议数据设备返回和发送到设备json格式说明.txt” 中
天气说明
                try {
                    //这里使用assets/weather/weather.json做为示例
                    InputStream inputStream = getAssets()×open("weather/
weather.json");
                    String weatherJson = FileUtils×getAssetsFile(inputStream);
                    Log.i("gy", "weatherJson: " + weatherJson);
                    manager.cmdSet68(weatherJson);
                } catch (IOException e) {
                    Log.e("gye", "weatherJson IOException: " + e.getMessage());
                    e.printStackTrace();
                }
                break;
            case Protocols.SET72://设置联系人
                List<ContactsInfo> contactsInfoList = new ArrayList<>();
                ContactsInfo info = new ContactsInfo();
                info.name = "陈";
                info.number = "13765636986";
                contactsInfoList.add(info);
                manager.cmdSet72(contactsInfoList);
                break;
            case Protocols.SET73://卸载表盘
                manager.cmdSet73("10");//"10" 对应的表盘文件序号
                break;
```

```
            case Protocols.SET74://卸载主题
                manager.cmdSet74("10");//"10" 对应的主题文件序号
                break;
            case Protocols.SET75://卸载应用
                manager.cmdSet75("10");//"10" 对应的应用文件序号
                break;
            case Protocols.SET77://设置音乐信息
                String song_title = "晴天";
                int song_title_length =
song_title.getBytes(StandardCharsets.UTF_8).length;
                manager.cmdSet77(1, song_title_length, song_title);
                //play_status 播放状态 0未播放, 1播放中
                //song_title_length 歌名字节长度
                //song_title 歌名
                break;
            case Protocols.SET78://设置时间制式 0 - 12小时制, 1 - 24小时制
                manager.cmdSet78(1);
                break;
            case Protocols.SET79://设置屏幕常亮 0不常亮, 1常亮
                manager.cmdSet79(1);
                break;
            case Protocols.SET80://设置抬手亮屏 startTime 开始时间, endTime 结
束时间, state 0关闭, 1打开
                manager.cmdSet80("08:00", "22:00", 1);
                break;
            case Protocols.SET81://设置勿扰模式 0关闭, 1打开
                manager.cmdSet81(1);
                break;
            case Protocols.SET82://设置开启震动 0关闭, 1打开
                manager.cmdSet82(1);
                break;
            case Protocols.SET83://设置GPS坐标(需要设备有GPS Sensor)
                //latitude 纬度, lat_direction 纬度方向  0南纬， 1北纬, longitude 经度,
long_direction 经度方向  0西经， 1东经
                //manager.cmdSet83("22.656565", 1, "112.323232", 1);
                break;
            case Protocols.SET84://设置智能睡眠开关  0关闭, 1打开
                manager.cmdSet84(1);
                break;
            case Protocols.SET85://设置开始血糖建模(YZ01)
                manager.cmdSet85();
                break;
            case Protocols.SET87://设置/重置设备(YZ01)
                manager.cmdSet87();
                break;
            case Protocols.SET88://设置自定义数据(Z50-Z51S)
//                  //1.通用协议
```

```
//                manager.cmdSet88(1, "SP1", new byte[]{0x0A, 0x0B});
                //2.设置EQ
                manager.cmdSet88(2, "SP2", new byte[]{0});
                break;
            case Protocols.SET89://设置开启自动健康测量开关
                manager.cmdSet89(1);
                break;
            case Protocols.SET90://设置清除设备数据
                manager.cmdSet90();
                break;
            case Protocols.SET91://设置设备关机
                manager.cmdSet91();
                break;
            case Protocols.SET92://设置健康检测开关(BX03) || 设置设备亮灯红/蓝(SR08)
                manager.cmdSet92(1);
                break;

////////////////////////////////////////////////////

            case ProtocolsHeadUtil.SET_MSG://推送消息通知
                manager.cmdSetMessage(MessageType.OTHER_APP, "今日头条", "俄乌冲突根本问题所在,北约东扩漂亮国转移内部矛盾幕后推动...");
//              //来电提醒
//              manager.cmdSetMessage(MessageType.CALL, "13656336856", "来电通知");
//              //去电提醒(挂断)
//              manager.cmdSetMessage(MessageType.CALL_OVER, "13656336856", "去电通知");
                break;
            case ProtocolsHeadUtil.SET_DIAL://推送大数据(包括表盘,主题,应用(游戏)等)
                mPushState.setText("正在读取数据...");
                mPushState.setVisibility(View.VISIBLE);

                /*
                 * 注意：
                 * 以下示例 G5_360_0001.bin文件针对的是项目名为G5_360的设备，如你的项目没有对应的.bin文件请向SDK提供方索要。
                 * 调试时请把示例对应的所有.bin文件替换为你自己的，不要直接推送与项目不符的.bin文件，会导致设备加载资源出错。
                 */

                //下面使用assets文件夹中文件进行示范。通常情况下APP文件获取应该在服务器下载或手机本地存储中。

                ////////////////////////推送 预制(固定)表盘使用代码 START////////////////////////
```

```
                /*
                 * 设置说明：
                 * 1. 预制(固定)表盘序号取值范围大于“0”小于“2001”， 例:
w1_swatch_1.bin, 取1为序号.
                 */
                String dialPath = "dial/G5_360_0001.bin";//1.预制(固定)表盘文件
                String numS = dialPath×split("_")[2].replace(".", ",").split(",")[0];//取
到0001
                String programObject = String×valueOf(Integer×parseInt(numS));//
把字符0001转为数值1再转字符1
                byte[] bytes = FileUtils×byteArraysFromAssets(MainActivity×this,
dialPath);
                ////////////////////////推送 预制(固定)表盘使用代码
END//////////////////////

                //开始推送
                if (bytes != null) {
                    /*
                     * bytes 推送的数据.
                     * ProgramType.DIAL 推送的数据类型为表盘.
                     * programObject 序号
                     */
                    //manager.cmdGet67(bytes, ProgramType.DIAL,
programObject);//为防止推送示例.bin与当前设备不匹配，默认注释方法调用
                    //注: 推送较大数据时, 设备会有缓存擦除等准备操作, 数据越大操作耗时
越久, 所以触发推送时可添加一个progress提示，例如：正在读取文件数据...
                } else {
                    mPushState.setVisibility(View.GONE);
                    Toast.makeText(MainActivity.this, "读取数据错误",
Toast.LENGTH_SHORT).show();
                }
                break;
            case ProtocolsHeadUtil.GET_THEME_REMIND://获取已推送的主题提醒
                manager.cmdGet76();
                break;
            case ProtocolsHeadUtil.SET_THEME_REMIND://设置主题提醒
                //下面使用assets文件夹中文件进行示范。通常情况下APP文件获取应该在
SD卡(包括但不限于“相册”等手机本地存储)中。

                String time = "08:05";//提醒的时间
                String remindLabel = "080501";//提醒标签序号(为不要冒号的
time(0805)和图片序号(01)的组合, 序号小于10补0所以写为01, 最大序号为99)
                String selected_days_week = "0111110";//共7个字符, 代表周日到周六,
为0不提醒, 为1提醒
                String imgPath = "remind/remind_1.png";//主题提醒图片(此图片为客户
商自己提供。图片宽高、如何在设备显示等请与设备软件开发负责人协商，SDK只做推
送)
```

```
                byte[] imgBytes =
FileUtils×byteArraysFromAssets(MainActivity×this, imgPath);
                Bitmap bitmapFromBytes =
FileUtils×getBitmapFromBytes(imgBytes);
                ThemeRemindInfo themeRemindInfo = new ThemeRemindInfo();
                themeRemindInfo.label = remindLabel;
                themeRemindInfo.alarms = time + "|" + selected_days_week;
                themeRemindInfo.imgBitmap = bitmapFromBytes;

                mPushState.setText("正在读取数据...");
                mPushState.setVisibility(View.VISIBLE);

//                manager.cmdSetThemeRemind(themeRemindInfo);//为防止推送示
例.png与当前设备不匹配，默认注释方法调用
                break;
            case ProtocolsHeadUtil.DELETE_THEME_REMIND://卸载主题提醒
                manager.cmdSet76("080501");
                break;

        }

    }
};

@SuppressLint("MissingPermission")
@Override
public void onConnectionStateChange(final BluetoothGatt gatt, final int
status, final int newState) {
    /*此回调方法用于自定义gatt操作, 一般不用*/
    Log.d("gy", "MainActivity onConnectionStateChange status: " + status + ",
newState: " + newState);
    if (newState == BluetoothProfile.STATE_CONNECTED) {
        Log.i("gy", "MainActivity State Change:  Connect !!!!!! : " +
gatt.getDevice().getName());
        address = gatt.getDevice().getAddress();
        bleName = gatt.getDevice().getName();
    }
}

@SuppressLint("MissingPermission")
@Override
public void onConnected() {
    Log.i("gy", "MainActivity onConnected !!!!!!");
    handler.sendEmptyMessageDelayed(MSG_CONNECTED, 1000);
    //gatt连接成功创建配对
//      if (manager.getBluetoothDevice().getBondState() ==
BluetoothDevice.BOND_NONE)//未绑定，去绑定
```

```
//            manager.getBluetoothDevice().createBond();
    }

    @Override
    public void onDisConnect() {
        Log.e("gy", "MainActivity onDisConnect ******");
        handler.sendEmptyMessageDelayed(MSG_DISCONNECT, 500);
    }

    @Override
    public void jsonObjectData(final String cmdKey, final JSONObject
jsonObject) {
        Log.i("gy", "cmdKey: " + cmdKey + ", jsonString: " + jsonObject.toString());
        JsonInfo jsonInfo = new JsonInfo();
        jsonInfo.cmdKey = cmdKey;
        jsonInfo.jsonObject = jsonObject;
        Message msg = Message×obtain();
        msg×what = MSG_JSON_DATA;
        msg×obj = jsonInfo;
        handler.sendMessageDelayed(msg, 500);
        //get,set,send等回调在子线程队列中进行,不直接更新UI
    }

    @SuppressLint("SetTextI18n")
    @Override
    public void pushDataProgress(final int progress, final int totalProgress) {
        mPushState.setText("推送中: " + progress + " %");
        mPushState.setVisibility(View.VISIBLE);
    }

    @Override
    public void pushDataProgressState(final int code) {
        mPushState.setText("");
        mPushState.setVisibility(View.GONE);
        if (code == 1) {
            Toast.makeText(this, "推送成功", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "推送失败", Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void pushDataNotStartedLowBattery() {
        handler.sendEmptyMessage(MSG_LOW_BATTERY);
    }

    @Override
```

```
    public void getGpsDataProgress(final int progress) {
        Log.d("gy", "getGpsDataProgress progress: " + progress);
        runOnUiThread(new Runnable() {//子线程不能直接更新UI
            @Override
            public void run() {
                String s = "获取中: " + progress + " %";
                int i = View.VISIBLE;
                mPushState.setText(s);
                if (progress == 100) i = View.GONE;
                mPushState.setVisibility(i);
            }
        });
    }

    @Override
    public void onRequestPermissionsResult(final int requestCode, @NonNull
final String[] permissions, @NonNull final int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        Log.d("gy", "onRequestPermissionsResult requestCode: " + requestCode +
", permissions.length= " + permissions.length + ", grantResults.length= " +
grantResults.length);
    }

    public Handler handler = new Handler(new
WeakReference<Handler.Callback>(new Handler.Callback() {
        @Override
        public boolean handleMessage(@NonNull final Message msg) {
            if (msg×what == MSG_DISCONNECT) {
                setState(false, Color.BLACK, "连接");
            } else if (msg×what == MSG_CONNECTED) {
                setState(true, Color.GREEN, "断开连接");
            } else if (msg×what == MSG_JSON_DATA) {
                JsonInfo jsonInfo = (JsonInfo) msg.obj;
                mTransferData.setText(jsonInfo.jsonObject.toString());

                /****************自行json解析 参考 "协议数据设备返回和发送到设备json格
式说明.txt"****************/

                /*
                 * 无论get,set,send 走 AnalyticalDataCallBack.jsonObjectData() 皆为
APP收到设备返回(或主动发送)的数据,
                 * 有具体返回内容的已在.txt文档中说明, 没有的只返回对应的
Protocols(jsonInfo.cmdKey)
                 */

                if (jsonInfo.cmdKey.equals(Protocols.GET0)) {
```

```
                //获取GET,0设备信息成功
            }
            //else if jsonInfo.cmdKey.equals(Protocols.GETXXX) ... 更多GETXXX

            else if (jsonInfo.cmdKey.equals(Protocols.SET14)) {//此协议由设备主动
发起
                try {
                    JSONObject jsonObject = new
JSONObject(jsonInfo.jsonObject.toString());
                    String photo_status = jsonObject.getString("photo_status");
                    //远程拍照状态  photo_status == 0退出拍照, 1打开拍照, 2拍照
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            } else if (jsonInfo.cmdKey.equals(Protocols.SET45)) {
                //备注：SR08接收到回调即设置时间成功，不用再继续解析
                try {
                    JSONObject jsonObject = new
JSONObject(jsonInfo.jsonObject.toString());
                    String sync_time_status =
jsonObject.getString("sync_time_status");
                    //设置时间状态  sync_time_status == 0成功, 1失败(手表未打开同步开
关)
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
            //else if jsonInfo.cmdKey.equals(Protocols.SETXXX) ... 更多SETXXX

            else if (jsonInfo.cmdKey.equals(Protocols.SEND10)) {

            }
            //else if jsonInfo.cmdKey.equals(Protocols.SENDXXX) ... 更多SENDXXX

        } else if (msg×what == MSG_LOW_BATTERY) {
            mPushState.setVisibility(View.GONE);
            mPushState.setText("正在读取数据...");
            Toast.makeText(MainActivity.this, "设备电量过低!",
Toast.LENGTH_SHORT).show();
        }
        return false;
    }
}).get());

}
```

<SR08 Ring FLAG>

(注: 设备连接成功后必须发送一次"GET,66、
SET,15,1"、"SET,46,0"、"SET,45"、"GET,0"这几个指令, 才可获取设备识别以及得到所有的GET和SEND指令相关的数据返回)

APP向设备获取数据的指令头:
GET,0 //获取设备详细数据
GET,10 //获取每天计步总数据
GET,11 //获取当天时间段计步数据
GET,12 //获取每天睡眠总数据
GET,13 //获取当天时间段睡眠数据
GET,14 //获取单次心率测试数据
GET,15 //获取闹钟信息
GET,16 //获取motion开关设置
GET,17 //获取当前心率,睡眠,步数(W1设备)
GET,18 //获取当前步数,距离,卡路里与send,10数据一样(W1设备)
GET,19 //获取历史体温 HB06-T
GET,20 //获取历史呼吸 HB06-T
GET,21 //获取当天时间段睡眠数据(SR08) 注：发送此协议前需要先有发过一次SET,15 协议才会返回数据
GET,22 //获取单次血压测试数据
GET,23 //获取单次血氧测试数据
GET,24 //获取熄屏时间(S666)
GET,61 //获取计步历史数据的日期信息
GET,64 //获取运动历史数据(暂未使用)
GET,65 //获取所有的(每次)单项运动历史汇总数据(需要设备有GPS Sensor, 无忽略)
GET,66 //询问蓝牙设备连接状态
GET,67 //推送数据(表盘,应用(游戏),主题,EPO文件,OTA资源文件等)
GET,69 //获取气压数据(需要设备有air pressure Sensor)
GET,70 //获取潜水数据(需要设备有GPS Sensor)
GET,71 //获取钓鱼数据(需要设备有GPS和air pressure Sensor)
GET,72 //获取联系人数据
GET,73 //获取已推送的所有表盘序号
GET,74 //获取已推送的所有主题序号
GET,75 //获取已推送的所有应用序号
GET,76 //获取已推送的所有主题提醒序号(W1设备)
GET,77 //开始测量心率(YZ01/SR08)
GET,78 //获取设备血糖建模天数(状态)YZ01
GET,79 //开始测量血糖(心率)YZ01/SR08
GET,80 //获取设备半小时存储一次的心率YZ01
GET,81 //开始测量血氧YZ01/SR08
GET,82 //开始测量血压YZ01/SR08
GET,83 //获取自动测量心率、血压、血氧数据(HB05)
GET,84 //获取心率和血氧(BX03)
GET,85 //开始测量体温(SR08)
GET,86 //获取设备每半小时存储的体温(SR08)
GET,87 //获取健康监测数据（心率、血氧、血压、血糖、体温）(SR08)
GET,88 //获取充电状态(SR08)

GET,89 //获取GSensor(SR08)
GET,90 //获取健康监测开关状态(SR08)
GET,92 //获取健康检测开关状态(BX03 暂未加)
GET,101 //获取对应的一次单项运动GPS数据(需要设备有GPS Sensor)

APP向设备设置数据的指令头:
SET,10 //设置个人信息
SET,11 //设置睡眠信息--暂不支持
SET,12 //设置久坐提醒
SET,13 //设置闹钟
SET,14 //由设备发送给APK的指令打开手机相机
SET,15 //设置APK是否前/后台运行状态
SET,19 //设置喝水提醒
SET,20 //设置手机找手表
SET,21 //设置motion 闹钟静音、抬手亮屏、来电拒接
SET,22 //设置表盘序号(BX03)
SET,23 //设置熄屏时间(S666)
SET,30 //设置女性生理期
SET,44 //设置语言
SET,45 //设置时间
SET,46 //设置单位制式
SET,68 //设置天气
SET,72 //设置联系人
SET,73 //卸载表盘
SET,74 //卸载主题
SET,75 //卸载应用
SET,76 //卸载主题提醒(W1设备)
SET,77 //设置音乐信息
SET,78 //设置时间制式
SET,79 //设置屏幕常亮
SET,80 //设置抬手亮屏
SET,81 //设置勿扰模式
SET,82 //设置开启震动
SET,83 //设置GPS坐标(需要设备有GPS Sensor)
SET,84 //设置智能睡眠开关(W1设备)
SET,85 //设置开始血糖建模(YZ01)
SET,87 //设置/重置设备(YZ01)
SET,88 //设置自定义数据，1.通用协议; 2.设置EQ (Z50-Z51S)
SET,89 //设置开启自动健康测量开关(SR08)
SET,90 //设置清除设备数据(SR08)
SET,91 //设置设备关机(SR08)
SET,92 //设置健康检测开关(BX03) / 设置设备亮灯红/蓝(SR08)

设备向APP发送的数据指令头: (注:必须要调用"SET,46"后才能接收到SEND返回的数据)
SEND,10 //发送计步数据
SEND,11 //发送当前时间段增量计步数据(暂未使用)
SEND,12 //发送实时连续心率测试数据

SEND,13 //设备找手机(APP收到指令做响铃或震动等响应)
SEND,14 //发送单次心率测试数据
SEND,15 //发送闹钟(暂未使用)
SEND,16 //发送短信(设备把短信内容和联系人或者号码发送到APP，APP调用API发送短信)
SEND,17 //发送睡眠状态(W1设备)
SEND,18 //发送设备上下键长按状态(W1设备)
SEND,19 //发送设备体温 HB06-T
SEND,20 //发送设备呼吸 HB06-T
SEND,22 //发送单次血压测试数据
SEND,23 //发送单次血氧饱和度测试数据
SEND,24 //发送设备电量
SEND,25 //发送呼叫护士 HB06-T
SEND,26 //发送健康检测数据 HB06-T
SEND,45 //发送请求索要时间
SEND,46 //发送设备单位制式(S6设备)
SEND,62 //发送来电操作(APP收到指令做挂断或接听电话)
SEND,69 //发送请求索要EPO数据(需要设备有GPS Sensor)
SEND,78 //发送时间制式状态
SEND,79 //发送屏幕常亮状态
SEND,81 //发送勿扰模式状态
SEND,82 //发送开启震动状态
SEND,83 //发送向APP索要GPS坐标(需要设备有GPS Sensor)
SEND,84 //发送心率和血氧(BX03设备)