NaviSDK
集成开发指南
（Anroid 版）

北京联创新图科技有限公司

2022-12-28

更新记录

版本	修改人	修改日期	修改说明	备注
V1.0		2022-12-28	初始版本	
V1.1		2022-01-03	1. 增加接口 getNaviPath 2. 修改接口，startCalibrate 3. 修改getCORSStatusRemote返回值 4. 删除setCORSParam 接口	
V1.2		2023-01-31	1. 增加了setNaviPathList接口 2. 增加了setNaviPathIndex接口 3. 增加了getParrLine接口 4. 调整了参数，优化入线	
V1.3		2023-02-14	增加了电机的适配，支持合众思壮电机，参考2.3,XintuEraParamBase说明	
V1.4		2023-02-21	1. 增加了千寻、中国移动、电台差分服务适配 2. 增加了软件(SDK)的激活功能 修改的接口：create 增加的接口： activeDevice isDeviceActivated getDeviceID resetCORSService	
V1.5		2023-03-02	主要增加了车辆跑偏的自动标定功能 修改了接口： XintuEraParamCalibrate startCalibrate	
V1.6		2023-03-10	1. 修复了开始导航时，处理时间过长的问题 2. 增加了在导航的过程中，可以快速设置当前路线	
V1.7		2023-03-17	1. 增加了SIM管理功能（切换内置、外置卡，获取状态） 2. 增加了camera功能	
V1.8		2023-03-28	1. 增加了转角陀螺仪支持，XintuEraDataSteer中返回转角陀螺仪数据 2. 增加了原极614E-P IMU 接口 3. 修复了回调函数没有返回车辆速度的问题	
V1.9		2023-05-24	1. 回滚了转角陀螺仪支持和原极614E-P 接口 2. 在SDK中实现行驶方向的判断 3. 集成了新图组合定位终端	对应SDK1.3.6

V2.0		2023-05-26	增加了获取车辆行驶方向(前进/倒车)的接口	
V2.1		2023-08-24	增加了起止点附近的状态回调	对应 SDK1.3.9
V2.2		2023-12-26	1. 电机状态通知实现方式优化（持续返回电机在线/离线状态） 2. 增加电机测试的接口 3. 返回主天线杆臂换算之前的坐标 4. 增加了电机速度控制增益参数	对应 SDK1.3.10
V2.3		2024-01-03	适配PDS14PIN 版本，修改了SDK创建的接口 XintuEraParamBase 增加了字段tablet_type // 表示平板型号 0：PDS23PIN 1：PDS14PIN	对应 SDK1.3.20
V2.4		2024-03-27	1. XintuEraDataLocation增加时间戳字段(可用于修正系统时间) 2. 适配可移动基站，修改和增加接口： 修改了XintuEraSDKManager.create接口 增加了setDiffData接口 修改了XintuEraParamCORS类 适配移动基站说明： XintuEraParamCORS在现有差分服务类别基础上，增加了两个选项：SERVICE_PROVIDER_RADIO和SERVICE_PROVIDER_RADIO_AUTO。（推荐用户使用AUTO选项） 当选择使用SERVICE_PROVIDER_RADIO_AUTO时，用户通过serialPortName 和 baudRate两个参数，配置差分数据来源的串口号和波特率； 当选择使用 SERVICE_PROVIDER_RADIO 时，则需要用户实现接口（接口为：IXintuEraCallbackDiff）管理差分数据的读取,并通过 SDK 接口 setDiffData 传递到 SDK 中。	对应 SDK1.3.25

本文档仅限授权的集成开发方使用，未经本公司书面许可，任何公司和个人不得以任何形式进行传播。

目录

1 概述

1.1 简介

NaviSDK 封装了农机导航以及差分定位服务功能，开发者在应用中集成 NaviSDK，开发 UI 层界面，即可快速完成农机导航系统的开发工作。

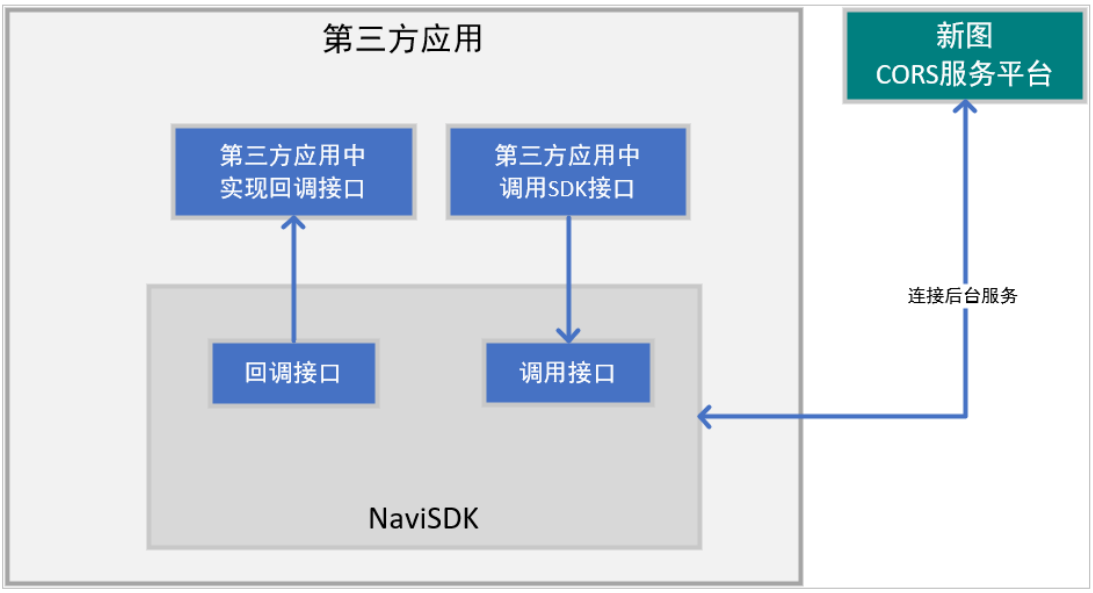
1.2 适用系统

本版本适用于 Android 平台。

1.3 名词解释

名称	解释
APPKEY	集成 SDK 的应用的唯一标识
APPSECRET	与 APPKEY 对应的密码
SERVICEID	新图差分服务平台的账号，设备出厂时内置，或在新图公众号平台进行注册申请
PASSWORD	与 SERVICEID 对应的密码
回调/通知接口	SDK 的接口，需要开发方进行实现

1.4 总结架构



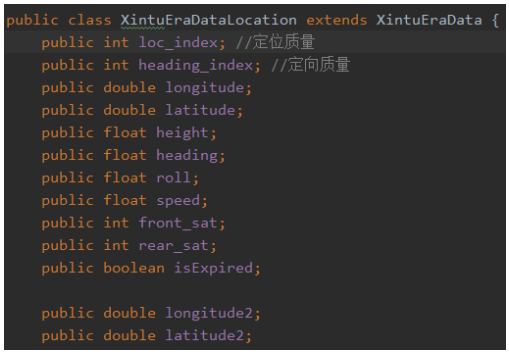
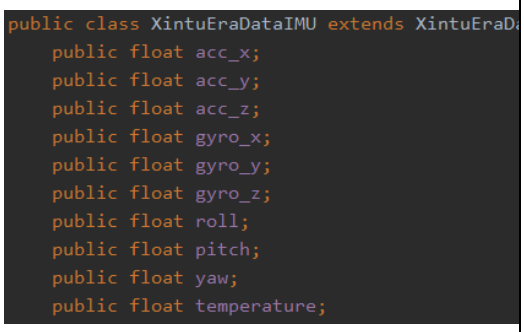
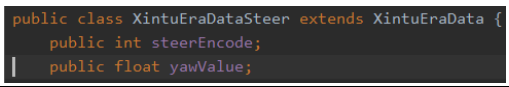
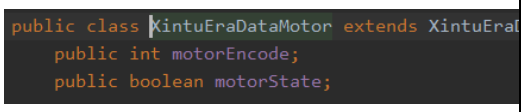
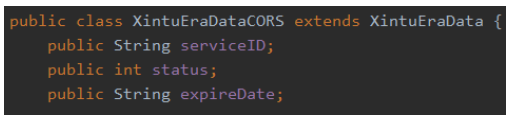
2 SDK 集成

2.1 回调接口

ID	接口	接口说明
1	IXintuEraCallbackData `public interface IXintuEraCallbackData {` `  void onLocationData(XintuEraDataLocation xintuEraDataLocation);` `  void onIMUData(XintuEraDataIMU xintuEraDataIMU);` `  void onSteerData(XintuEraDataSteer xintuEraDataSteer);` `  void onMotorData(XintuEraDataMotor xintuEraDataMotor);`	SDK 通过该回调接口将实时数据通知到应用层 onLocationData 车辆的定位数据 onIMUData IMU 数据 onSteerData 转向相关数据 onMotorData 电机相关数据
2	IXintuEraCallbackNavi `public interface IXintuEraCallbackNavi {` `  void onStart();` `  void onPause();` `  void onResume();` `  void onStop(float compensationValue, int stopReason);` `  void onErrorValue(float value);` `  void onErrorCode(int err_code);` `  void onDrawRefresh();` `  void onNearEndPoint(boolean isStartPoint);` `}`	SDK 通过该回调接口将导航相关数据通知到应用层 onStart 开始导航 onPause 暂停导航 onResume 恢复导航 onStop 停止导航 onErrorValue 当前实时误差值 onErrorCode 当出现错误时的错误编号（待定） onDrawRefresh 如果使用 SDK 中的 DrawView 进行绘制时，SDK 通过该接口通知应用层进行刷新 DrawView（如果应用层不使用 SDK 中的绘制方法，忽略该消息即可） onNearEndPoint(boolean isStartPoint) 车辆接近起点或终点附近 参数 isStartPoint， True，表示起点 False，表示终点
3	IXintuEraCallbackDiff `public interface IXintuEraCallbackDiff {` `  void onStart();` `  void onStop();` `  void onGGAData(String strData);` `}`	SDK 通过该回调接口将 GGA 数据通知到应用层，同时通过该接口，从应用层获取 rtcm 差分数据 onGGAData 返回 GGA 数据 onStart 启动时调用，用户可以实现打开串口等操作 onStop 停止时调用， 用户可以实现关闭串口等操作

2.2 回调接口相关的参数类型

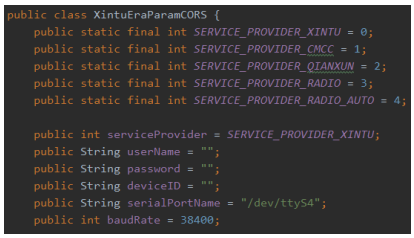
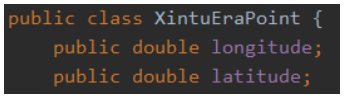
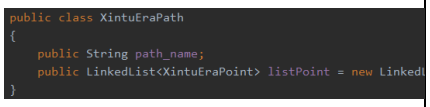
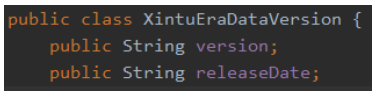
ID	接口	接口说明
1	XintuEraDataLocation	SDK 通过该回调接口将实时定位数据通知到应用层 Timestamp 时间戳 loc_index 定位质量 0：无效解 1：单点解

		2：浮点解 3：固定解 heading_index 定向质量 （取值与 loc_index 相同） longitude 经度 latitude 纬度 height 高程 heading 车头方向（正北为零，顺时针为正） roll 车辆左右倾斜角（度） speed 车速，m/s front_sat 前天线可见星 rear_sat 后天线可见星 isExpired 数据是否已过期 longitude2 主天线经度 latitude2 主天线纬度
2	XintuEraIMUData 	acc_x X 轴加速度 acc_y Y 轴加速度 acc_z Z 轴加速度 gyro_x X 轴角速度 gyro_y Y 轴角速度 gyro_z Z 轴角速度 roll 翻滚角 pitch 俯仰角 yaw 偏航角 temperature 温度
3	XintuEraDataSteer 	steerEncode 前轮转角编码器编码值 yawValue 转角陀螺仪数据
4	XintuEraDataMotor 	motorEncode 电机编码器编码值 motorState 电机状态 true：电机正常 false：电机无返回值
5	XintuEraDataCORS 	serviceID 账号 state 账号状态 -1：账号不存在 0：账号正常 1：没有激活 2：接近过期 3：已过期 expireDate 到期时间
6	XintuEraDataSIM	carrier_id 运营商 ID -1：未知 1：中国移动 2：中国联通

	`` public int carrier_id; public int signal_level; public int network_class; public int service_state; public int data_conn_state; ```	3：中国电信 signal_level 信号质量 0：未知 1：差 2：中 3：好 4：很好 network_class 网络类型（3G,4G…） -1：未知 1：2G 2：3G 3：4G 4：2.75G service_state 服务状态 -1：未知 0：服务正常 1：停止服务 2：仅限紧急呼叫 3：已关机 data_conn_state 数据连接状态 -1：未知 0：断开连接 1：连接着 2：已连接 3：挂起

2.3 SDK 调用参数类说明

ID	类	参数说明
1	基本参数 XintuEraParamBase `public class XintuEraParamBase {` `public String appKey;` `public String appSecret;` `public boolean simuNavi;` `public String storagePath;` `public int motor_type;` `public int tablet_type;`	appKey appSecret simuNavi 导航方式 true：模拟导航 false：真实导航 storagePath: 参数和 log 存储路径 PDS 平板为 /sdcard/ motor_type 电机类型： 0：科亚电机 1：合众思壮电机 tablet_type 平板型号 0：PDS23PIN 1：PDS14PIN
2	车辆参数 XintuEraParamVehicle `public class XintuEraParamVehicle` `{` `public float vehicle_length;` `public float vehicle_height;` `public float max_turn_angle;` `public float antenna_position_lat;` `public float antenna_position_lon;` `public float antenna_direction;` `public float turn_radius;` `public float vehicle_width;`	vehicle_length 车长度（轴距） vehicle_height 车高度，以天线安装位置为主（若有） max_turn_angle 最大转向角 antenna_position_lat gnss 主天线到后轴的距离(横向） antenna_position_lon gnss 主天线到后轴的距离（纵向） antenna_direction 天线安装方向，ant1->ant2,与车头方向夹角，顺时针为正 turn_radius 转弯半径 vehicle_width 车辆后轮轮距
3	控制参数 XintuEraParamControl `public class XintuEraParamControl` `{` `public static final int ANGLE_METHOD_BY_ENCODE = 0;` `public static final int ANGLE_METHOD_BY_GYRO = 1;` `public static final int ANGLE_METHOD_BY_POSE = 2;` `public static final int NAVI_MODE_LINE = 10;` `public static final int NAVI_MODE_POLYLINE = 11;` `public float look_forward_distance;` `public int angle_method;` `public int navi_mode;`	look_forward_distance 前视距离 angle_method 前轮转角获取方式 0：编码器 1：转角陀螺仪 2：车身姿态 navi_mode 导航模式（直线模式/曲线模式）
4	标定参数 XintuEraParamCalibrate `public class XintuEraParamCalibrate` `{` `public float steer_compensation;` `public float heading_compensation;` `public int left_steer_encode;` `public int right_steer_encode;` `public int left_motor_encode;` `public int right_motor_encode;`	steer_compensation 前轮转角编码器补偿角度 heading_compensation 方向补偿 left_steer_encode 方向盘向左打到头时转角编码值 right_steer_encode 方向盘向右打到头时转角编码值 left_motor_encode 方向盘向左打到头时电机编码值 right_motor_encode 方向盘向右打到头时电机编码值 start_point 标定路线起点 end_point 标定路线终点
5	CORS 参数 XintuEraParamCORS	serviceProvider CORS 服务提供方 userName 用户名（账号） passWord 账号密码

		deviceID 设备 ID serialPortName 串口设备号 （SERVICE_PROVIDER_RADIO_AUTO 时有效） baudRate 串口波特率 （SERVICE_PROVIDER_RADIO_AUTO 时有效）
6	坐标点 XintuEraPoint 	longitude 经度 latitude 纬度
7	导航路径 XintuEraPath 	path_name 路线名称 listPoint 坐标列表
8	SDK 版本号 XintuEraDataVersion 	version 版本号 releaseDate 发布时间

2.4 SDK 调用接口

NaviSDK 的调用全部包含在类 XintuEraSDKManager 中，应用中，全局只允许创建一个实例，并使用该实例调用相应的接口。

ID	接口名称	参数和接口功能说明
1	create	功能：创建 SDK 参数： Context context //android context XintuEraParamBase paramBase //参见上文说明 XintuEraParamCORS paramCORS //参见上文说明 IXintuEraCallbackData callbackData //参见上文说明 IXintuEraCallbackNavi callbackNavi //参见上文说明 IXintueraCallbackDiff callbackDiff //参见上文说明 其中，callbackDiff 只在 paramCORS.serviceProvider 配置为 SERVICE_PROVIDER_RADIO 有效 返回值：无
2	destroy	功能：销毁 SDK 参数：无 返回值：无
3	setNaviPath	功能：设置导航路径 参数： XintuEraPath naviPath //参见上文说明 返回值：无
4	getNaviPath	功能：获取导航路径 参数：无 返回值：XintuEraPath naviPath //参见上文说明
5	startNavigation	功能：开始导航 参数：无 返回值：无
7	pauseNavigation	功能：暂停导航 参数：无 返回值：无
7	setSimuNaviDirection	功能：设置**模拟**导航方向（只在模拟导航时有效） 参数： boolean isForward //true：前进，false：后退 返回值：无
9	resumeNavigation	功能：恢复导航 参数：无 返回值：无
0	stopNavigation	功能：停止导航 参数：无 返回值：无
10	startCalibrate	功能：开始标定导航

		参数： Calibrate_mode //0: 方向标定，1：转角编码器标定 返回值：无 注：标定过程主要用于生成前轮转角编码器的误差补偿值。
11	stopCalibrate	功能：停止标定导航 参数：无 返回值：无
12	setVehicleParam	功能：设置车辆参数 参数： XintuEraParamVehicle param //参见上文说明 返回值：无
13	setControlParam	功能：设置控制参数 参数： XintuEraParamControl param //参见上文说明 返回值：无
14	setCalibrateParam	功能：设置标定参数 参数： XintuEraParamCalibrate param //参见上文说明 返回值：无
15	getVehicleParam	功能：获取车辆参数 参数： XintuEraParamVehicle param //参见上文说明 返回值：无
16	getControlParam	功能：获取控制参数 参数： XintuEraParamControl param //参见上文说明 返回值：无
17	getCalibrateParam	功能：获取标定参数 参数： XintuEraParamCalibrate param //参见上文说明 返回值：无
18	restoreGNSSModule	功能： 恢复 GNSS 硬件模块的设置 参数：无 返回值：无 注：设备初始启用或恢复出厂设置时，可以调用该接口进行硬件模块的配置
19	getCORSStatusLocal	功能：获取 CORS 服务状态（本地存储） 参数： XintuEraDataCORS xintuEraDataCORS//参见上文说明 返回值： true: 返回正常 false: 读取错误
20	getCORSStatusRemote	功能：设置 CORS 服务状态（从服务端获取，并同步到本地）

		参数： String serviceID, //账号 String password, //密码 XintuEraDataCORS xintuEraDataCORS //参见上文说明 XintuEraParamCORS param //参见上文说明 返回值： 0：OK -1：network error -2：password error -3：local file read error
21	queryQRCode	功能：获取充值二维码 参数： String serviceID //账号 int month //充值月数 返回值： Bitmap bmp 图片
22	getVersion	功能：获取 SDK 版本 参数： XintuEraDataVersion param //参见上文说明 返回值：无
23	setNaviPathList	功能：设置路线列表 参数： LinkedList<XintuEraPath> listPath //路线列表 返回值： 无
24	setNaviPathIndex	功能：设置路线索引 参数： int index //索引值 boolean needTranspose //是否需要反转 AB 点 返回值： 无
25	getParrLine	功能：获取平行线 参数： XintuEraPoint ptStart //输入，起点坐标 XintuEraPoint ptEnd, //输入，终点坐标 float dis, //输入，平行线间距 boolean isLeft, //输入，是否为左侧平行线 XintuEraPoint ptResultStart, //输出，平行线起点 XintuEraPoint ptResultEnd //输出，平行线终点 返回值： 无
26	activeDevice	功能：激活设备 参数：

		Context context //输入，Android Context String appKey //APP Key, 同 create 时，在 XintuEraParamBase 中传递的 AppKey Sting appSecret //APP Secret 返回值： 激活结果 0：ok -1：本地读 deviceID 失败 -2：网络失败 -3：其他错误 -4：保存 KEY 失败
27	isDeviceActivated	功能：判断当前设备是否激活 参数： Context context //输入，Android Context 返回值： true：已激活 false：未激活
28	getDeviceID	功能：读取设备 ID 参数： 无 返回值： 设备 ID
29	resetCORSService	功能：重新启动 CORS 服务 参数： XintuEraParamCORS param //CORS 服务参数 返回值： 无
30	quickSwitchPath	功能：导航过程中快速切换路线 参数： XintuEraPath naviPath, //路线 boolean isForward //是否正向行驶 返回值： 无
31	getSIMChannel	功能：获取当前使用的卡槽 参数： 无 返回值： 0：外置卡 1：内置卡
32	switchSIMChannel	功能：切换当期使用的卡槽 参数： 0：外置卡 1：内置卡

		返回值： 无
33	getNaviDirection	功能：获取车辆行驶方向（前进/倒车） 参数：无 返回值： -1: 倒车 0：停车 1: 前进
34	startMotorTest	功能：开始电机测试（电机左右各旋转一定角度，2 秒之后停止） 参数：无 返回值：无
35	setControlGain	功能：设置电机控制增益 参数：增益系数：0.5~1.5 之间 返回值： 无
36	setDiffData	功能：传递差分数据 参数： byte[] byteData //存放数据 int len //数据长度 返回值： 无

3 SDK 调用示例

```java
public class MainService extends Service implements IXintuEraCallbackData, IXintuEraCallbackNavi{
    private static final String TAG = "MainService";

    public MainService()
    {

    }
    @Override
    public void onCreate() {
        super.onCreate();
        XintuEraParamBase xintuEraParamBase = new XintuEraParamBase();
        xintuEraParamBase.appKey = BaseConfig.APP_KEY;
        xintuEraParamBase.appSecret = BaseConfig.APP_SECRET;
        xintuEraParamBase.simuNavi = BaseConfig.isSimu;
        xintuEraParamBase.storagePath = BaseConfig.storagePath;
        xintuEraParamBase.motor_type = 1;
        xintuEraParamBase.tablet_type = 1;

        XintuEraParamCORS xintuEraParamCORS = new XintuEraParamCORS();
        readCORSInfo(xintuEraParamCORS);
        sdkManager.create(this, xintuEraParamBase, xintuEraParamCORS,
                this, this);
        sdkManager.setControlGain(1.0f);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        sdkManager.destroy();
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        return mBinder;
    }
    @Override
    public boolean onUnbind(Intent intent)
    {
        Log.d(TAG, "MainService onUnbind()");
        return true;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId)
    {
        Log.d(TAG, "MainService onStartCommand() ");
        //startForeground(10, null);
        return START_NOT_STICKY;
    }
```

```
    @Override
    public void onIMUData(XintuEraDataIMU xintuEraDataIMU) {
        m_xintuEraDataIMU.set(xintuEraDataIMU);
    }

    @Override
    public void onSteerData(XintuEraDataSteer xintuEraDataSteer) {
        m_xintuEraDataSteer.set(xintuEraDataSteer);
    }

    @Override
    public void onMotorData(XintuEraDataMotor xintuEraDataMotor) {
        m_xintuEraDataMotor.set(xintuEraDataMotor);
    }

    @Override
    public void onSIMData(XintuEraDataSIM xintuEraDataSIM) {
        m_xintuEraDataSIM.set(xintuEraDataSIM);
    }

    public void getLocationData(XintuEraDataLocation param)
    {
        m_xintuEraDataLocation.get(param);
    }

    public void getIMUData(XintuEraDataIMU param)
    {
        m_xintuEraDataIMU.get(param);
    }
    public void getSteerData(XintuEraDataSteer param)
    {
        m_xintuEraDataSteer.get(param);
    }
    public void getMotorData(XintuEraDataMotor param)
    {
        m_xintuEraDataMotor.get(param);
    }
    @Override
    public void onSIMData(XintuEraDataSIM xintuEraDataSIM) {
        m_xintuEraDataSIM.set(xintuEraDataSIM);
    }

    public void getLocationData(XintuEraDataLocation param)
    {
        m_xintuEraDataLocation.get(param);
    }

    public void getIMUData(XintuEraDataIMU param)
    {
        m_xintuEraDataIMU.get(param);
    }
    public void getSteerData(XintuEraDataSteer param)
    {
        m_xintuEraDataSteer.get(param);
    }
    public void getMotorData(XintuEraDataMotor param)
    {
        m_xintuEraDataMotor.get(param);
    }

    public void getSIMData(XintuEraDataSIM param)
    {
        m_xintuEraDataSIM.get(param);
    }
```

```
@Override
public void onStart() {
}

@Override
public void onPause() {
    if (m_handler != null) {
        m_handler.sendEmptyMessage(MSG_NAVI_PAUSE);
    }
}

@Override
public void onResume() {
    if (m_handler != null) {
        m_handler.sendEmptyMessage(MSG_NAVI_RESUME);
    }
}

@Override
public void onStop(float compensationValue, int stopReason) {
    if (m_handler != null) {
        Message msg = m_handler.obtainMessage();
        msg.what = MSG_NAVI_STOP;
        msg.arg1 = (int)(compensationValue * 100);
        msg.arg2 = stopReason;
        m_handler.sendMessage(msg);
    }

}

@Override
public void onErrorValue(float value)
{
    if (m_handler != null) {
        Message msg = m_handler.obtainMessage();
        msg.what = MSG_NAVI_ERR_VALUE;
        msg.arg1 = (int)(value * 100);
        m_handler.sendMessage(msg);
    }
}
@Override
public void onErrorCode(int err_code)
{
    if (m_handler != null) {
        Message msg = m_handler.obtainMessage();
        msg.what = MSG_NAVI_ERR_CODE;
        msg.arg1 = err_code;
        m_handler.sendMessage(msg);
    }
}

@Override
public void onDrawRefresh()
{
    if (m_handler != null) {
        Message msg = m_handler.obtainMessage();
        msg.what = MSG_NAVI_DRAW_REFRESH;
        m_handler.sendMessage(msg);
    }
}
```

```
@Override
public void onNearEndPoint(boolean b) {
    if (m_handler != null) {
        Message msg = m_handler.obtainMessage();
        if (b) {
            msg.what = MSG_NAVI_NEAR_ENTRY_POINT;
            Log.d(TAG, "near entry point");
        }
        else {
            msg.what = MSG_NAVI_NEAR_ENDING_POINT;
            Log.d(TAG, "near ending point");
        }
        m_handler.sendMessage(msg);
    }

}

//interface
public void resetCORSService(XintuEraParamCORS xintuEraParamCORS)
{
    //Log.d(TAG, JSON.toJSONString(xintuEraParamCORS));
    sdkManager.resetCORSService(xintuEraParamCORS);
}
public String getDeviceID()
{
    return sdkManager.getDeviceID();
}
public boolean isDeviceActivated()
{
    return sdkManager.isDeviceActivated(this);
}

public int activeDevice()
{
    return sdkManager.activeDevice(this, BaseConfig.APP_KEY);
}

public void setNaviPathList(LinkedList<XintuEraPath> listPath)
{
    sdkManager.setNaviPathList(listPath);
}
public void setNaviPathIndex(int index, boolean needTranspose)
{
    sdkManager.setNaviPathIndex(index, needTranspose);
}

public void getParrLine(XintuEraPoint ptStart, XintuEraPoint ptEnd, float dis, boolean isLeft,
                        XintuEraPoint ptResultStart, XintuEraPoint ptResultEnd)
{
    sdkManager.getParrLine(ptStart, ptEnd, dis, isLeft, ptResultStart, ptResultEnd);
}

public void quickSwitchPath(XintuEraPath naviPath) {
    sdkManager.quickSwitchPath(naviPath);
}
public void setNaviPath(XintuEraPath naviPath)
{
    sdkManager.setNaviPath(naviPath);
}
public XintuEraPath getNaviPath()
{
    return sdkManager.getNaviPath();
}

public void startNavigation()
{
    sdkManager.startNavigation();
}

public void pauseNavigation()
{
    sdkManager.pauseNavigation();
}
```

```
    public void resumeNavigation()
    {
        sdkManager.resumeNavigation();
    }
    public void stopNavigation()
    {
        sdkManager.stopNavigation();
    }
    public void setSimuNaviDirection(boolean isForward)
    {
        sdkManager.setSimuNaviDirection(isForward);
    }

    public void startCalibrate(int mode)
    {
        sdkManager.startCalibrate(mode);
    }

    public void stopCalibrate()
    {
        sdkManager.stopCalibrate();
    }
    public void getVehicleParam(XintuEraParamVehicle param)
    {
        sdkManager.getVehicleParam(param);
    }
    public void setVehicleParam(XintuEraParamVehicle param)
    {
        sdkManager.setVehicleParam(param);
    }

    public void getControlParam(XintuEraParamControl param)
    {
        sdkManager.getControlParam(param);
    }
    public void setControlParam(XintuEraParamControl param)
    {
        sdkManager.setControlParam(param);
    }
    public void getCalibrateParam(XintuEraParamCalibrate param)
    {
        sdkManager.getCalibrateParam(param);
    }
    public void setCalibrateParam(XintuEraParamCalibrate param)
    {
        sdkManager.setCalibrateParam(param);
    }
    public void restoreGNSSModule()
    {
        sdkManager.restoreGNSSModule();
    }

    public boolean getCORSStatusLocal(XintuEraDataCORS xintuEraDataCORS)
    {
        return sdkManager.getCORSStatusLocal(xintuEraDataCORS);
    }

    public int getCORSStatusRemote(String serviceID, String password,
                                   XintuEraDataCORS xintuEraDataCORS)
    {
        return sdkManager.getCORSStatusRemote(serviceID, password, xintuEraDataCORS);
    }

    public Bitmap queryQRCode(String serviceID, int month)
    {
        return sdkManager.queryQRCode(serviceID, month);
    }

    public void startMotorTest()
    {
        sdkManager.startMotorTest();
    }

    public void restoreGnssModule() {
        sdkManager.restoreGNSSModule();
    }
}
```