# 千寻差分数据
# 嵌入式 SDK

# 开发指南

V2.0.4

**千寻位置网络有限公司**

**2017 年 12 月·上海**

# 法律声明



**千寻位置网络有限公司**

**联系邮箱：service@qxwz.com**

**官方网站：www.qxwz.com**

# 目 录

# 第一章 概述

## 1.1 产品简介

千寻位置网差分数据嵌入式 SDK 为基于嵌入式平台的应用提供自动注册差分账号、自动获取差分数据的服务接口包，专注于为广大开发者提供便捷的高精度位置服务，这里简称为**差分数据 SDK**。

该 SDK 免费对外开放，适用于大批量高精度终端的差分账号智能分配和差分数据获取的繁琐操作。使用该 SDK 获取差分数据后，差分数据传输、芯片写入工作完全交由开发者完成，为开发者提供最大灵活性。用户在获取这些服务的同时，不用考虑数据存储，数据安全和繁琐的网络交互等细节。

千寻位置网差分数据 SDK 主要提供以下功能：

用户使用自己的高精度芯片差分算法， 使用 SDK 可以获取差分数据，用户可以使用这些差分数据灌进相关定位芯片或算法。

# 第二章 开发指南

## 2.1 开发准备

### 2.1.1 注册为开发者

开发者需要入驻千寻位置网，在千寻位置官网（qxwz.com）注册为千寻位置开发者，注册成功并通过个人/企业用户认证后，便可成为千寻位置网的开发者。详细信息请见千寻官网帮助文档。

### 2.1.2 购买千寻 FindM 或 FindCM 服务

### 2.1.3 进入管理控制台-控制中心-千寻跬步 / 千寻知寸-服务实例，选择服务实例对应管理进行开发者配置工作


1. 选择服务实例接入方式为 SDK；
2. 选择设备服务号绑定方式：自动绑定或手动绑定；

   若选择手动绑定，则需对需要连接千寻服务的设备进行绑定操作；
3. 选择设备服务号激活方式：自动激活或手动激活或终端激活方式；

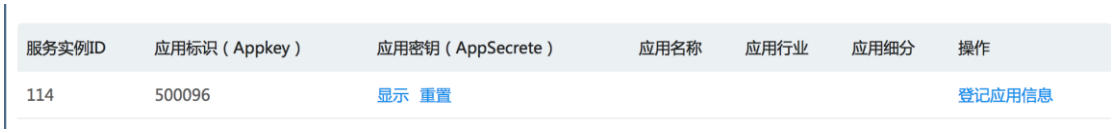
   若选择手动激活，则需在千寻官网管理控制台中进行激活后使用；

   若选择终端激活方式，则需在终端调用千寻激活接口后进行激活后使用。

### 2.1.4 进入管理控制台-控制中心-千寻跬步 / 千寻知寸-设备服务号，创建设备服务号

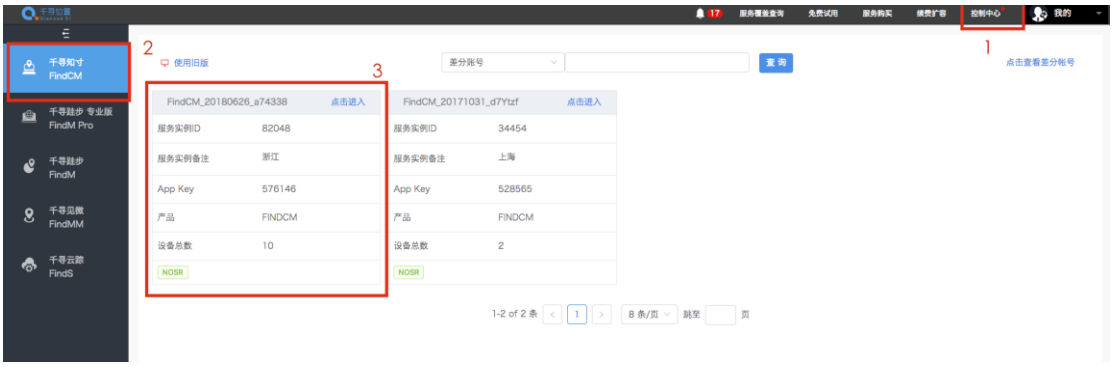

### 2.1.5 前往控制中心-千寻跬步 / 千寻知寸-应用列表获取 AppKey, AppSecret

官网旧版界面




官网新版界面

1） 登录千寻官网，进入控制中心，找到对应的服务，比如 FindCM。




2）点击进入服务实例-服务详情-账号信息，即可获取应用标识（AppKey）和应用密钥（AppSecret）


### 2.1.6 集成 SDK，写入 AppKey, AppSecret, DeviceID（设备 SN），DeviceType（设备类型）

### 2.1.7 绑定与激活操作（可选）

1. 若服务实例配置为需要绑定设备，则需要在控制中心-千寻跬步 / 千寻知寸-设备服务号进行绑定工作，写入 SDK 的 DeviceID 和 DeviceType


2. 若服务实例配置为需要激活，前往控制中心-千寻跬步 / 千寻知寸-设备服务号对当前设备进行激活操作。


## 2.2 接入流程图

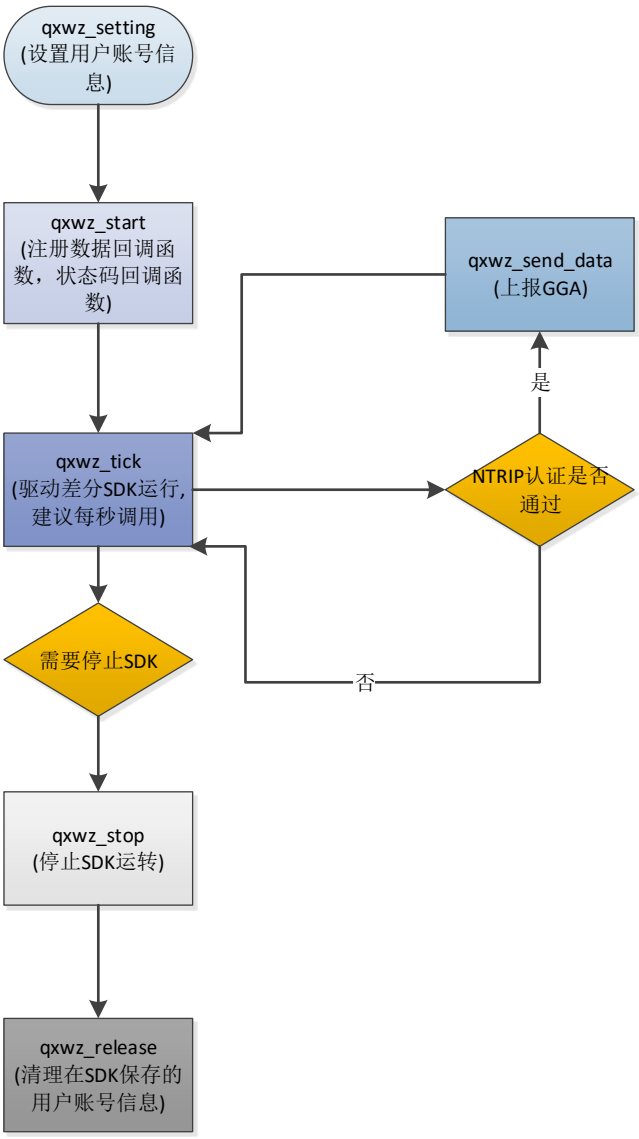
SDK 使用流程大体如下：

1，工程引入 SDK lib 包，调用对应的 API 接口完成操作。

2，由于 API 有先后依赖关系， 列出相关 API 的调用流程。

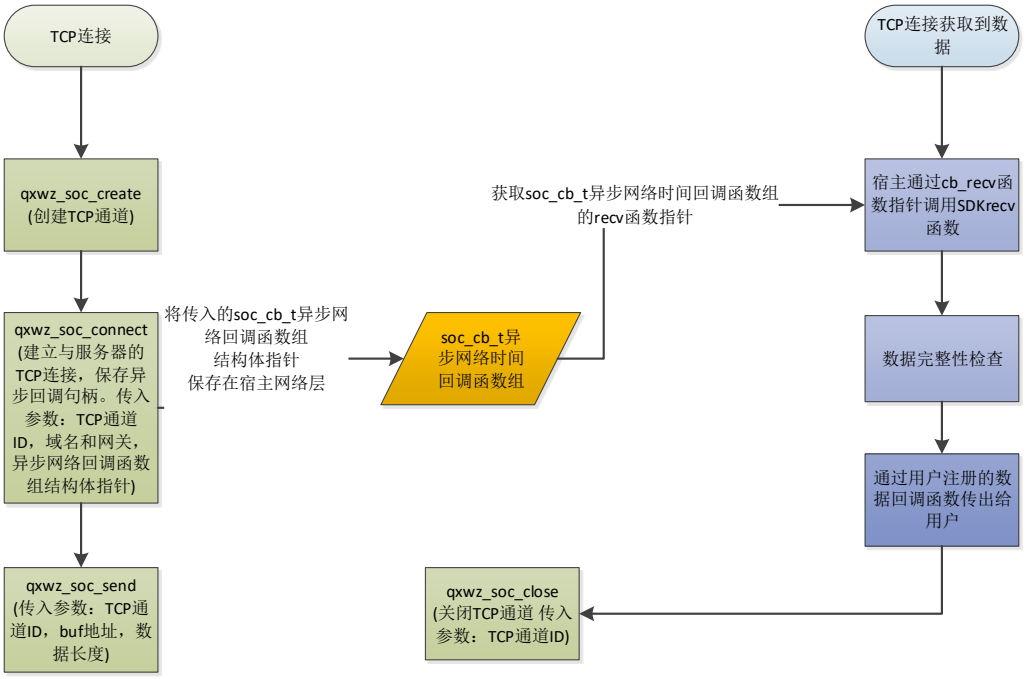
qxwz_setting→qxwz_start->qxwz_tick->qxwz_stop->qxwz_release。

如下图所示：




3，列出网络层相关 API 的调用流程。

如下图所示：




上图中，绿色背景的函数是需要宿主来实现提供给 SDK 调用 。黄色背景的结构体需要保存在宿主网络层。

# 2.3 配置工程

## 2.3.1 lib 和头文件引入

**1 ,**准备千寻位置网服务邮箱提供的 lib 包。例如 “ QXSI-RTK_1.2.0.20170801.MTK.lib”

2，工程引入头文件，“qxwz_sdk.h” ，“qxwz_socket.h” ，“qxwz_status.h” ，“qxwz_types.h”

# 2.4 网络模块交互设计

## 2.4.1 设计流程

差分数据 SDK 的网络请求和数据回传，外部需重写(参考 qxwz_socket.h 头文件)，由于 SDK 将网络模块的功能剥离出来，因此需要开发者自己实现网络模块功能。（备注：

SDK 在设计的时候已经是多通道设计，但是开发者在实现网络的时候可以按照单通道来实现）。网络层实现可以参考《千寻 SDK 网络层接口指南》。

### 2.4.2 网络模块实现流程

1，由开发者实现 qxwz_soc_create 方法，该方法主要是用来创建一个 TCP，如果宿主是单通道设计的，那么可以返回一个固定的大于等于 1 的数字。该方法定义于 qxwz_socket.h 头文件上。定义为

qxwz_s32_t qxwz_soc_create(qxwz_u32_t nwk_id)

参数 nwk_id 是指网络服务标识，例如：0 表示默认连接，1 表示中国电信，2 表示中国移动，3 表示中国联通，4 表示 wifi，5 表示有线连接，6 表示其他网络连接方式。

函数的返回值有 2 种情况 :创建成功返回一个固定的大于等于 1 的数字 ,创建失败返回-1。

2， 由开发者实现 qxwz_soc_connect 方法，该方法主要是由 SDK 调用来发起 TCP 连接，

连接服务端。该方法定义于 qxwz_socket.h 头文件上。函数定义为：

qxwz_s32_t qxwz_soc_connect(qxwz_s32_t soc, host_info_t *host, soc_cb_t *cbs)

该函数的返回值有 3 种情况：成功返回 0，失败返回-1，异步模式返回-2。

参数 qxwz_s32_t soc 表示网络句柄。唯一对应一个网络应用程序

参数 host_info_t*host 表示服务器信息，内含域名和端口，需要底层网络进行域名解析。

参数 soc_cb_t * cbs 表示异步网络层事件通知函数组。异步网络模式下，需要将异步网络层事件通知函数注册进入网络层。异步网络层事件通知函数以函数指针的形式包含在自定义类型 soc_cb_t 结构体内。

异步模式下，当网络层发生某个网络事件时，需要调用对应的通知函数通知 SDK 的网络应用。而 SDK 每个网络应用都需要注册所需的事件函数。

同步模式下，异步事件通知函数组可以不必提供。

如果宿主使用的是异步 socket(非循环等待 AT 指令应答的实现)，当宿主检测到连接完成，调用异步事件通知函数组 soc_cb_t 结构体变量里的异步事件通知函数，即为 qxwz_s32_t ( *cb_connect ) (qxwz_s32_t soc)函数指针，将异步 connect 事件通知 SDK 做上层应用处理。

3， 由开发者实现 qxwz_soc_send 方法，该方法主要是由 SDK 调用来将数据发送给服务端。该方法定义于 qxwz_socket.h 头文件上。函数定义为：

qxwz_s32_t qxwz_soc_send(qxwz_s32_t soc, const qxwz_s8_t *buf, qxwz_s32_t len)

该函数的返回值有 3 种情况：成功返回发送数据长度，失败返回-1，异步模式返回-2。

参数 qxwz_s32_t soc 表示网络句柄。唯一对应一个网络应用程序。

参数 qxwz_s8_t *buf 表示数据缓存空间，缓存了待发送数据。

参数 qxwz_s32_t len 表示待发送数据的长度。

如果宿主使用的是异步 socket(非循环等待 AT 指令应答的实现)。当宿主检测到网络层资源可以发送数据，调用异步事件通知函数组 soc_cb_t 结构体变量里的异步事件通知函数，即为 qxwz_s32_t ( *cb_send ) (qxwz_s32_t soc)函数指针， 将异步 send 事件通知 SDK 做上层应用处理。

4， 由开发者实现 qxwz_soc_recv 方法，该方法主要是由 SDK 调用来接收数据。该

函数定义于 qxwz_socket.h 头文件上。函数定义为：

qxwz_s32_t qxwz_soc_recv(qxwz_s32_t soc, qxwz_s8_t *buf, qxwz_s32_t len)

异步模式下该函数不必实现。同步模式下，SDK 会调用该函数来获取网络层接收到的数据，获取成功返回本次获取的数据长度值，失败返回-1。

参数 qxwz_s32_t soc 表示网络句柄。唯一对应一个网络应用程序。

参数 qxwz_s8_t *buf 表示数据缓存空间，存储待接收的数据。

参数 qxwz_s32_t len 表示待接收数据的最大长度。

如果宿主使用的是异步 socket(非循环等待 AT 指令应答的实现)当宿主获取到数据的时候，通过调用异步事件通知函数组 soc_cb_t 结构体变量里的异步事件通知函数，即为 qxwz_s32_t ( *cb_recv ) (qxwz_s32_t soc, const qxwz_s8_t *buf, qxwz_u32_t len) 函数指针，将网络层（socket 层）信息传给函数指针所指向的 SDK 内部的函数，获取数据传递给 SDK。其中，参数 buf 指针所指向的是网络层的 data buffer。

5，由开发者实现 qxwz_soc_close 方法，该方法主要是由 SDK 调用来关闭 TCP 连接。该函数定义于 qxwz_socket.h 头文件上。函数定义为：

qxwz_s32_t qxwz_soc_close(qxwz_s32_t soc)

该函数的返回值有 3 种情况：成功返回 0，失败返回-1，异步模式返回-2。

参数 qxwz_s32_t soc 表示网络句柄。唯一对应一个网络应用程序。

6，当宿主检测到对端服务器关闭了链路时，需要调用异步事件通知函数组 soc_cb_t 结构体变量里的异步事件通知函数，即为 cb_close，通告 SDK 链路管道关闭了。

7，任何出现的 socket 异常，比如：接收数据失败、TCP 连接意外断开等，在异步模式下，可以调用全局 soc_cb_t 结构体变量里的回调函数指针，即为 cb_status 方法,以便通知 SDK TCP 连接出现异常。

# 第三章 SDK 接口说明

## 3.1 配置 SDK

### 3.1.1 接口定义

```
/**
  * 设置rtcm配置
  * @param config  config 包括 appKey、secret、device_ID、device_Type，auth_Servconfig, Servconfig，appKey和appsecret到官网申请到的,device_ID是可以填写mac地址、商品编号，device_Type是设备类型，可以填入类似sony、hitargt等，这四个参数必须填入。auth_server_config 专网环境用来配置鉴权专网ip和port。ntrip_server_config 专网环境用来配置差分专网ip和port。
  * @param isRealtime 用户可根据自身平台做出选择，若是可以提供系统绝对时间，则isRealtime选择TRUE； 若是只能提供相对时间，则isRealtime选择FALSE
  * @return:
  *   0 成功返回0
  *  -1 失败返回-1
  /**
  qxwz_s32_t qxwz_setting(const qxwz_usr_config_t* config, qxwz_bool_t isRealtime);
```

### 3.1.2 示例代码

```
qxwz_config config;
config.appkey = "/*申请的 appKey*/";
config.appsecret= "/*申请的 secret*/";
config.device_ID = "00:11:67:44:41:6A"; //设备的唯一 ID，建议使用设备的序列号
config.device_Type = "Test_device";
qxwz_setting(&config,true);
```

# 3.2 启动 RTCM

## 3.2.1 接口定义

```
/**
 * @param[in]   data_rsp:   RTCM数据获取函数回调指针
 * @param[in]   status_rsp: SDK状态获取函数回调指针
 *
 * @return:
 *    0 成功返回0
 *   -1 失败返回-1
 */
      qxwz_s32_t qxwz_start(qxwz_data_response_t * data_rsp, qxwz_status_response_t * status_rsp)
```

## 3.2.2 示例代码

```
          qxwz_data_response_t data_res = {
                  receive_iprtcm,
                  NULL
          };
          qxwz_status_response_t status_res = {
                  receive_status
          };
          qxwz_start(&data_res,&status_res) ;
```

# 3.3 获取 RTCM 数据

## 3.3.1 接口定义

```
/**
* 当收到RTCM数据时，向外部程序通知收到的RTCM数据
*
*   @param rtcm         rtcm数据
*   @param length       rtcm数据长度
*   @param length       rtcm数据类型，差分数据用户选择为RTCM_TYPE_RAW类型
*/

qxwz_void_t (*qxwz_sdk_data_response)(qxwz_void_t *data, qxwz_u32_t length, qxwz_data_type_e type);
```

## 3.4 获取 SDK 返回状态

### 3.4.1 类型接口定义

```
/**
 *  RTCM服务状态码回调函数
 *  @param qxwz_status_response
 */
qxwz_void_t (*qxwz_sdk_status_response) (qxwz_s32_t status);
```

备注：返回状态码见 3.10 状态码表

## 3.5 获取用户账号状态信息

### 3.5.1 接口定义

```
    /**
    *
    * 查询用户账号信息
    *
    * @return   qxwz_account_info*
    *
    */——
    const qxwz_account_info*   getqxwzAccount(void);
```

通过 getqxwzAccount 接口获取用户状态信息，当距离账号过期还有 10 天时，我们会通过状态信息回调的方式，通知用户，状态码为 2011。当用户账号已过期时，相应的状态码为 2010。

```
    /**
    * SDK 账号信息
    */
    typedef struct {
        char *appkey;
        char *deviceID;
        char *deviceType;
        time_t expire_time;   /*自1970年1月1日的秒数*/
        char *NtripUserName;
    } qxwz_account_info;
```

示例代码：

```
qxwz_account_info *p_account_info = NULL;
void   get_qxwz_sdk_account_info(void)
{
  p_account_info = getqxwzAccount();
  if(p_account_info->appkey != NULL) {
       printf("appkey=%s\n",p_account_info->appkey);
  }
  if(p_account_info->NtripUserName != NULL) {
       printf("NtripUserName=%s\n",p_account_info->NtripUserName);
  }
  printf("expire_time=%d\n",p_account_info->expire_time);
}
```

# 3.6 发送 GGA

## 3.6.1 接口定义

```
/**
 *   向ntrip服务器发送GGA字符串用来获取rtcm数据
 *   @param data NMEA协议里面GGA语句
 *   @param data字符串的size大小
 *   @param  用户发送的data数据类型，一般选择为UDATA_GGA类型
 */

qxwz_s32_t qxwz_send_data(const void *data, qxwz_u32_t size, qxwz_udata_type_e type);
```

备注：每隔一段时间（建议根据设备使用场景中 1 秒移动的距离，时间可以是 2 秒，建议不要大于 1 分钟）调用 qxwz_send_data 方法，将原始的 GGA 字符串传给 SDK。

## 3.6.2 示例代码

```
char* gga = "$GPGGA,000001,3112.518576,N,12127.901251,E,1,8,1,0,M,-32,M,3,0*4B\r\n");
qxwz_send_data(gga,128,UDATA_GGA);
```

## 3.7 固定频率使 SDK 工作

### 3.7.1 接口定义

```
/**
* 可在此方法内执行具体的任务，由外部程序定时调用，由于网路超时等情况，外部程序可能不能及时调用
* *   @param system_time UTC 时间（计于 1970 年 1 月 1 号 0 零时）或者相对时间节拍
*/
      qxwz_s32_t qxwz_tick(qxwz_u32_t system_time);
```

备注：相对时间节拍是指每隔一段时间（建议为秒级）调用 qxwz_tick 方法，并且传入一个表示应用启动到调用 qxwz_tick 为止的一个秒级的时间，（例如：第一次调用传入 0，第二次调用传入 1，第三次传入 2，以此类推）。qxwz_tick 的返回值是 SDK 状态，当返回值是 0 时表示 SDK 已经停止工作。

## 3.8 停止 SDK 工作

### 3.8.1 接口定义

```
/**
* 停止 SDK 工作，释放 SDK 资源，不再接收 rtcm 差分数据和状态码
*
*/
      qxwz_void_t qxwz_stop(void) ;
```

备注： qxwz_stop 调用后，SDK 已经停止工作，不能再接收差分数据和返回状态码信息。若要再次启动，需要调用 qxwz_start 函数启动 SDK。qxwz_stop 调用后，qxwz_tick 不能停止，应该继续提供时间节拍。

## 3.9 释放 SDK 用户信息

### 3.9.1 接口定义

```
/**
* 释放保存在 SDK 的用户账户信息
*
*/
        qxwz_void_t qxwz_release(void);
```

备注：qxwz_release 调用后，SDK 已经释放保存在 SDK 的用户账户信息。若要再次启动，需要调用 qxwz_setting 函数重新配置用户账户信息。

## 3.10 获取 SDK 版本

### 3.10.1 接口定义

```
/**
* 获取 SDK 当前版本信息
*
*/
        qxwz_u8_t* qxwz_sdk_version(void);
```

备注：可以获取当前运行的 SDK 版本的版本号。

## 3.11 返回状态码

| 状态码 | 状态说明 |
| --- | --- |
| 1000 | ntrip 连接到服务器 |
| 1001 | ntrip 断开服务器 |

| | |
|---|---|
| 1002 | APP KEY 认证失败 |
| 1003 | APP KEY 认证成功 |
| 1004 | 网络异常 |
| 1005 | NTRIP 用户已经达到上限 |
| 1006 | NTRIP 用户不存在 |
| 1011 | 非法 GGA |
| 1013 | 正在连接 ntrip 服务器 |
| 1014 | ntrip 正在接受数据 |
| 1015 | 非法 APP KEY |
| 1016 | 非法 APP SECRET |
| 1017 | deviceType 错误 |
| 1018 | deviceid 错误 |
| 2001 | 缺少参数 |
| 2002 | 账号不存在 |
| 2003 | 账号重复 |
| 2004 | 错误密码 |
| 2005 | 账号未激活 |
| 2006 | 没有有效的账号 |
| 2007 | POPUser 不存在 |
| 2008 | 服务端内部错误 |
| 2010 | 账号已过期，需续费 |
| 2011 | 账号即将过期 |

| | |
|---|---|
| 2012 | 目前的账号没有绑定 |
| 2013 | 鉴权参数错误 |
| 2014 | SSOToken 错误 |
| 2015 | 账号数不够 |
| 1020 | SDK 内部错误 |
| 1021 | Ntrip 播发数据正常 |
| 1022 | Ntrip 认证失败 |
| 1035 | setting 错误 |
| 1036 | 没有调用 setting 函数 |

# 第四章 常见问题(FAQ)

## 4.1 用户使用过程中的问题

Q1 :文档只给出了相应的函数调用 ,终端实现高精度定位的一个具体的流程，请给出说明？

A1：终端实现高精度定位的一个具体的流程:大致思路是，

STEP1：去官网申请 appkey 和 appsecret，完成用户认证，申请配额，订阅相关服务，绑定设备 sn。具体可参考官网帮助文档。

STEP2：调用 qxwz_s32_t qxwz_setting(const qxwz_usr_config_t* config, qxwz_bool_t isRealtime)这个方法，将用户配置信息传入 SDK，参考接入文档 3.1.2

STEP3:调用 qxwz_s32_t qxwz_start(qxwz_data_response_t * data_rsp, qxwz_status_response_t * status_rsp)方法，该方法的两个参数是回调函数的函数指针，这两个回调函数需由用户实现。

STEP4：每秒调用 qxwz_tick()方法，并传入系统时间，具体请参考接入文档 3.6。

此外，每秒将接收到的 GGA 数据发送给 SDK，由 SDK 发送给千寻服务器，然后服务器下发 RTCM 差分数据包，用户将接收到的数据包回写到 GPS 模块,然后硬件会做纠偏，后面再吐出的 GGA 等卫星数据就是纠偏过的，更加精准的。

Q2: 是不是我每一秒调用一次 void qxwz_tick(int32_tsystem_time)这个函数，SDK 就可以正常的工作了？

A2: 每一秒调用一次 void qxwz_tick(int32_tsystem_time)这个函数，这个是必要条件，此外 SDK 正常工作还需要网络模块和 GPS 模块正常工作，以及每秒发送正确的 GGA 数据给 SDK

Q3: 为什么接入后程序跑飞？

A3:SDK 需要 4K 字节以上的 STACK，需要用户设置，要不然可能会造成内存踩踏，导致跑飞。

Q4：当收到 RTCM 数据时，向外部程序通知收到的 RTCM 数据差分数据回调函数

qxwz_void_t (*qxwz_sdk_data_response)(qxwz_void_t *data, qxwz_u32_t length, qxwz_data_type_e type)

错误事件通知/运行状态通知 RTCM 服务状态码回调函数

qxwz_void_t (*qxwz_sdk_status_response)(qxwz_s32_t status)

发送 GGA 字符串，用来获取 rtcm 数据

qxwz_s32_t qxwz_send_data(const void *data, qxwz_u32_t size, qxwz_udata_type_e type)

1. 这三个函数的用途
2. 是否需要由我们来编写

A4:前两个函数需要用户实现函数体，第三个用户只需要每秒调用即可。

Q5：获取差分数据和发送 GGA 的时间间隔可以调整不?

A5: 获取差分数据的时间是根据您的需求变化的，如果是 FindM 服务，那么可能是 15 秒播发或者 1 秒播发，如果是 FindCM 服务，一般就是每秒播发，发送 GGA 的时间可以调整，但最好不低于 5 秒一次。

Q6:如何实现 SDK 的 LOG 信息输出?

A6:目前 SDK 中的 LOG 输出是需要用户实现的。SDK 提供了一个函数接口 extern int qxwz_printf(const char *fmt,...)给用户。用户需要实现这一函数来实现 LOG 输出。

差分数据回调函数上报数据为二进制格式，无法以字符串方式打印，如需查看内容，请以十六进制格式打印。

Q7：通过集成千寻的 SDK，连接到服务器的主要流程以及会收到哪些数据？

A7:目前连接到千寻服务器主要有以下流程:

STEP1：第一个 HTTP 请求（短连接），获取千寻服务器时间戳记。

STEP2：第二个 HTTP 请求（短连接），主要是对用户账号信息等进行鉴权、获取服务器配置、以及获取 ntrip 差分账号等。

STEP3：Ntrip 连接（长连接），主要是使用上一步骤获取的差分账号信息连接 Ntrip 服务器

STEP4：定时发送 GGA 到 Ntrip 服务器，Ntrip 会依据用户的位置信息，播发 RTCM 差分数据。

Q8：无法获得差分数据是什么原因？

A8:　无法获得差分数据，除了鉴权失败以外，还可能有以下几种原因：

1. 发送 GGA 格式不合法

正常启动差分 SDK 后，差分 SDK 返回 1007（NTRIP 认证成功），调用 qxwz_send_data 方法后，拿不到差分数据，请检查发送的 GGA 字符串格式是否正确，数据帧结束符<CR><LF>需保留。

2. 差分区域不支持

正常启动差分 SDK 后，差分 SDK 返回 1007（NTRIP 认证成功），调用 sendGGA 方法后，拿不到差分数据，是因为你发送的经纬度所在地区暂时没有提供差分服务。您可在千寻官网查询服务覆盖范围：https://www.qxwz.com/map-cover.html。如果您有疑问请联系千寻客服。

3. 原来可以获取，后来无法获取

正常启动差分 SDK 后，开始能正常拿到差分数据，过段时间后拿不到差分数据，这是因为差分 SDK 必须在 120 秒内向服务器传递最新位置，如果在 120 秒内没有给服务器最新位置，服务器将不会播发差分数据。为了正常拿到差分数据，建议您每秒向差分 SDK 发送最新位置。

Q8：如何验证是否使用了差分数据？

A8: 根据 NMEA 语句判断当前是否为高精度解算位置。开发者可根据定位模块吐出的 NMEA 语句进行判断。如下：

$GPGGA,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,M,<10>,M,<11>,<12>*xx，当第<6>个字段表示 GPS 状态，出现以下三类之一时：2（码差分），4（固定解），5（浮点解），代表当前是高精度解算位置。如果长时间

未进入高精度定位，请确认设备在开阔无遮挡空间测试，确定 GNSS 信号强度足够。

## 4.2 用户账号申请的问题

1.注意事项，device_Type 与 device_ID 填写时不要出现空格。

2.官网上的设备 sn 对应的是 deviceID，在用户程序中填写的必须与官网的保持一致。

## 4.3 用户账号的问题

（1）SDK 返回 1015 表示 appKey 无效，请输入正确的 appKey/appSecret。

（2）差分 SDK 返回 2002 表示设备未绑定账号，请绑定当前使用的 deviceId 和 deviceType；

（3）差分 SDK 返回 2005 表示账号需要手动激活，请激活该 appKey 下的差分账号；

（4）差分 SDK 返回 2006 表示没有可用的账号，appKey 用户量已经达到上限，请在 appKey 账号里面申请扩容，允许更多用户接入差分服务；

（5）差分 SDK 返回 2012 表示设备服务号在该 sik 下只能手动绑定。

（6）差分 SDK 返回 2010 表示账号已过期。

（7）差分 SDK 会返回 2010 (账号已过期需续费)，需开发者登陆千寻位置官网进行续费操作。