千寻云踪Web API

开发指南

（Version 1.4.2）

千寻位置网络有限公司

2016 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

千寻云踪 Web API 是千寻位置网面向开发者提供的一套位置开发调用接口。基于高并发、分布式、流式计算等技术，提供海量终端位置数据的上传、存储、轨迹抽稀、地理围栏、统计报表等各种功能；打造满足不同行业及应用需求的云服务，使得位置数据的接入及后续处理变得『易如反掌』。

该 Web API 适用于管理各种人、车、物等实体，例如商用车、乘用车、手机、可穿戴设备等。基于该 Web API 进行二次开发，通过简单的实体管理、位置管理、配置管理，便可快速搭建功能强大的业务系统。

要使用该 Web API，需要申请 appKey 和 appSecret，请在千寻位置网(qxwz.com)注册账号，并按照帮助文档完成注册、认证、应用创建、appKey 和 appSecret 获取工作，具体流程可见开发指南的准备工作章节。

1.2 功能特点

1.2.1 实体管理

（1）实体操作：支持实体的创建、删除、更新、查询

（2）自定义扩展：支持自定义扩展字段，满足个性化的业务诉求

1.2.2 位置管理

（1）自定义扩展：轨迹点支持自定义扩展字段，满足个性化的业务诉求

（2）位置上传：支持上传单个轨迹点、批量上传轨迹点

（3）实时位置：支持查看指定实体列表的实时位置

（4）历史轨迹：支持查询指定实体连续 10 天的历史轨迹

1.2.3 配置管理

（1）抽稀配置：可以选择是否抽稀、设置抽稀延迟、抽稀程度。轨迹抽稀可以在保持轨迹基本形态前提下，

降低存储成本、提高查询效率

（2）存储配置：支持自定义数据存储周期，最短 1 天，最长 3 年，默认存储半年。

（3）地理围栏配置：全局控制是否开启地理围栏服务

1.2.5 事件管理

（1）事件类型：支持车道偏离预警、前车碰撞预警等，支持用户自定义事件

（2）事件操作：支持事件的创建、删除、更新、查询

（3）模块配合：可将查询事件与查询历史轨迹配合操作

1.2.6 行程管理

（1）行程操作：支持地理围栏的创建、删除、更新、查询

（2）模块配合：可将查询行程与查询历史轨迹配合操作

1.2.7 地理围栏

（1）地理围栏类型：目前仅支持多边形，后续将支持圆形、行政区划、行驶路线等多种类型的地理围栏

（2）地理围栏操作：支持地理围栏的创建、删除、更新、查询

（3）地理围栏事件：支持进地理围栏、出地理围栏报警

（4）地理围栏统计报表：支持按照指定的时间段，查询某个实体进出地理围栏的统计情况

（5）模块配合：配合规则引擎、消息通道、统计报表，可支持实时报警和统计报表功能

1.2.8 规则引擎

（1）规则引擎类型： 规则引擎为开发者提供极大的灵活性，本期仅开放地理围栏的规则。后续陆续开放更多规则，来满足和适应用户多样化的业务需求

（2）规则引擎操作：支持规则引擎的创建、删除、更新、查询

（3）配合消息通道：配合消息通道，可将由规则产生的数据实时写入消息通道，用于实时报警服务

（4）配合统计报表：配合统计报表，可将由规则产生的数据落入统计报表，用于统计报表服务

1.2.9 消息通道

（1）消息通道类型：当前支持阿里云 MNS，后续将支持钉钉、微信、短信、邮件等多种消息通道

（2）消息通道操作：支持消息通道的创建、删除、更新、查询

1.2.10 关系管理

（1）关系管理：支持绑定、解绑以及绑定关系的查询

（2）关系类型：当前支持规则与应用进行全局绑定，一个地理围栏与一个实体绑定

1.2.11 统计报表

（1）实体数量报表：查询所有实体的总量、新增、活跃的数量统计，支持日、月两个级别的时间维度

（2）时长报表：查询所有实体和单个实体的时长统计，支持日、月两个级别的时间维度

（3）里程报表：查询所有实体和单个实体的里程统计，支持日、月两个级别的时间维度

（4）事件点报表：支持查询进出地理围栏的事件点列表

（6）区域分布报表：支持查询某个业务类别下实体的区域分布统计

1.2.12 网关协议-JT808

（1）支持交通部的 JT808 协议，支持终端注册、注销、鉴权和位置数据上传。

（2）实体管理：支持创建 JT808 类别的实体

1.3 配套工具

为了更好地支持开发者使用千寻云踪服务，千寻位置已为您提供移动版的配套工具，Web 版的配套工具正在紧张研发中。通过配套工具，您可以

（1） 查看任意一个应用下的实体列表、多个实体的实时位置和单个实体的历史轨迹（Web 版和移动版）

（2） 增、删、改、查实体（仅限 Web 版，移动版仅提供查询操作）

（3） 增、删、改、查地理围栏，建立地理围栏与实体的绑定关系（仅限 Web 版）

（4） 增、删、改、查规则，建立规则与应用、规则与实体的绑定关系（仅限 Web 版）

（5） 提供查看基础的统计报表，帮助您快速掌握整体概况（Web 版&移动版）

（6） 配置管理，支持开发者进行抽稀配置、存储配置、是否开启地理围栏配置（Web 版）

整体而言，移动端只提供查询的操作，对于增、删、改的操作请在 Web 版中完成；配套工具主要用于帮助开发者更好地管理和体验服务，自身不承载业务逻辑。如果您需要搭建业务系统，请基于千寻云踪 API&SDK 进行二次开发。下面对千寻云踪 APP 和千寻云踪 Web 版的功能进行简单介绍

1.3.1 千寻云踪 APP（iOS 版）

千寻云踪 APP（iOS 版）是千寻云踪 Web API 的配套工具，适用于 iOS7.0 及其以上版本，您可扫描如下二维码下载千寻云踪 APP：

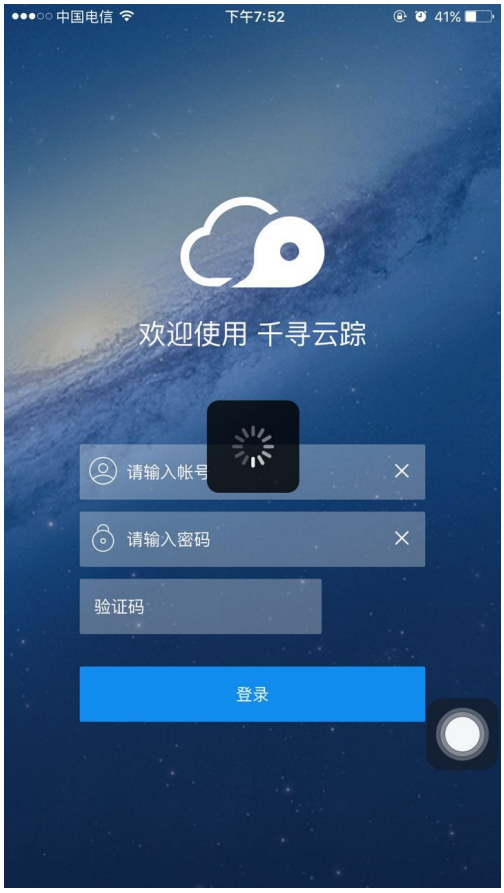
扫一扫下载 APP

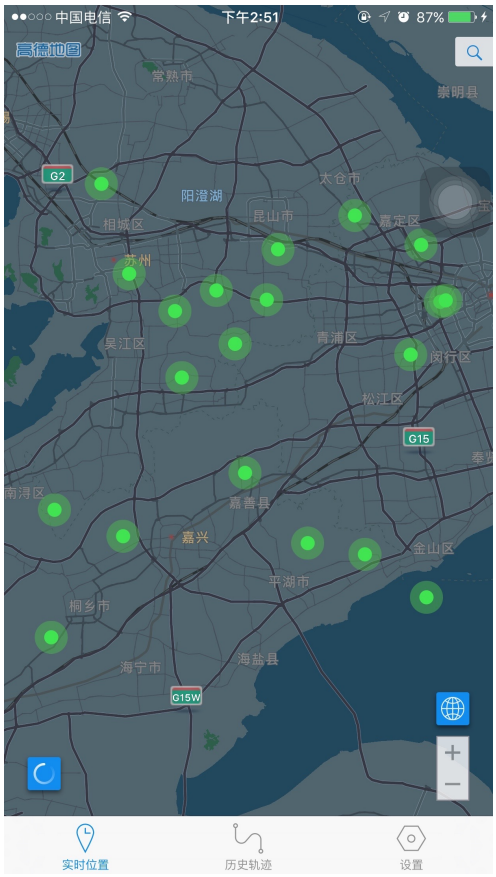
主要提供如下功能：

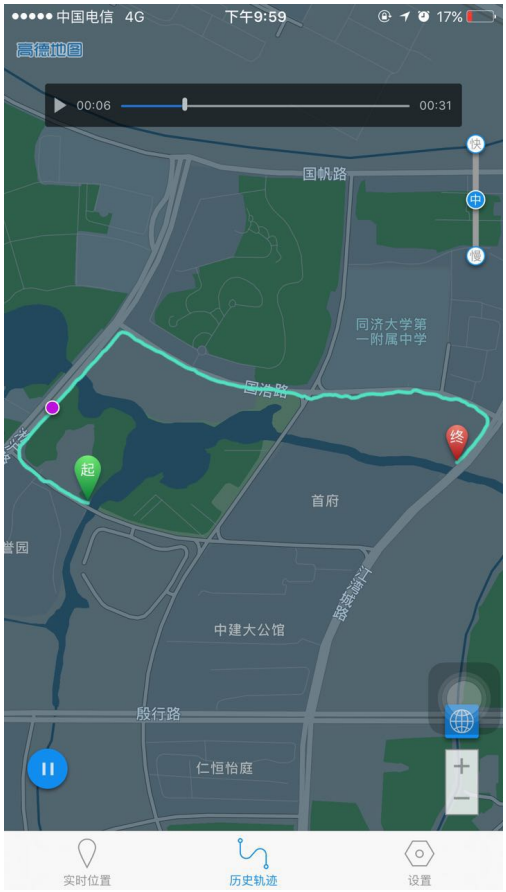
功能	简介
用户管理	1. 使用千寻位置网的用户账号进行登录 2. 登录成功后便可查看已创建的应用列表 3. 选择某个应用，便可查询该应用下的实体列表、多个实体的实时位置和单个实体历

	史轨迹
实时位置	1. 查看指定实体列表的实时位置 2. 查询单个实体的状态信息，包括位置、速度信息，支持跳转到历史轨迹
历史轨迹	1. 查询的单个实体的历史轨迹 2. 支持轨迹回放，可设置快、中、慢三个档次
体验云踪	1. 支持将 iOS 手机当做一个实体注册在某个应用下，形成体验闭环 2. 用户可开启/关闭上传位置通道，开启后 2 个小时终止上传位置。
新手引导	1.引导用户体验千寻云踪应用

附：功能截图




中国电信
下午2:51
87%
高德地图
常熟市
崇明县
G2
阳澄湖
太仓市
相城区
昆山市
嘉定区
苏州
吴江区
青浦区
闵行区
松江区
G15
南浔区
嘉善县
嘉兴
金山区
平湖市
桐乡市
海宁市
海盐县
G15W
实时位置
历史轨迹
设置


中国电信 4G
下午9:59
17%
高德地图
00:06
00:31
国帆路
同济大学第一附属中学
国浩路
起
终
首府
中建大公馆
殷行路
仁恒怡庭
实时位置
历史轨迹
设置


中国电信 4G
下午9:59
16%
体验云踪
应用
默认分组
实体名称
正在记录位置数据，两小时后将停止记录
停止记录

1.3.2 千寻云踪 Web 版

正在紧张研发中，敬请期待。基于 Web 版的工具您可以完成对实体管理、位置管理、配置管理、消息通道管理和关系管理，并提供基本的统计报表功能。

1.4 HTTPS 支持

千寻云踪全面支持 http/https，您可以按需使用。如选用 https 协议，安全性高但会影响访问效率。

第二章 开发指南

2.1 准备工作

2.1.1 入驻千寻

开发者需要入驻千寻位置网，才能够使用千寻云踪 Web API。请在千寻官网(qxwz.com)注册用户，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。

2.1.2 创建应用

在千寻位置网官网的管理控制台成功创建应用后，会自动生成该应用的 appKey&appSerect；appKey 和 appSecret 是应用的凭证信息，也是调用 API/SDK 的唯一凭证，请务必妥善保管。

1.

2.2 新手上路

新手上路为您提供最基本的使用流程，您只需要完成下面三步，便可快速搭建起一个最简单的业务系统。

2.2.1 创建实体

首先您需要创建一个实体，这个实体可以是任意的人、车、物，例如商用车、乘用车、可穿戴设备、GPS 跟踪器等。在同一个应用下面，实体名称全局唯一。

相关接口：

（1） 创建实体：gpsp.entity.createEntity

2.2.2 上传位置

您获取该实体的位置后，便可调用接口上传到云端进行存储和处理。

相关接口：

（1） 上传单个轨迹点：gpsp.point.addTrackPoint

（2） 批量上传轨迹点：gpsp.point.addTrackPoints

2.2.3 查询实时位置和历史轨迹

将实体的位置源源不断地上传到云平台上后，调用查看实时位置和历史轨迹的接口，配合地图上便可显示实体的实时位置和历史轨迹。

相关接口：

（1） 查看实时位置：gpsp.point.locate

（2） 查看历史轨迹：gpsp.point.queryPoints

2.3 接口列表

在新手上路章节，您搭建了一个最简单的业务原型。为了丰富拓展您的业务系统，您需要了解更多的功能接口。

2.3.1 概览

2.3.1.1URL 地址

URL 定义规范：http://api.qxwz.com/rest/api_name/appKey/0?_sign=xxxxxxxxxxx，其中

（1） http://api.qxwz.com/rest 为千寻 Web API 服务的统一地址前缀

（2） api_name 为接口名称

（3） appKey 为应用的唯一标识

（4） 0 为保留值

（5） _sign 为加签值，根据参数、appSecret、时间戳共同生成。有关_sign 的生成算法请查看下文的签名机制章节。

说明：

（1） Web API 支持 HTTP 协议，以 POST、GET 方式进行请求

（2） 参数需要以标准的 URL 请求格式构造，若参数为对象，需要进行 JSON 序列化。例如创建实体的 URL 为：

http://api.qxwz.com/rest/gpsp.entity.createEntity/10000/0?_sign=xxxxxxxxxxxxx&entity={"entityName":"沪 A1234","entityType":"电动车","attributes":{"color":"黑色","driver":"张三"}}

2.3.1.2 签名机制

签名算法使用遵循 RFC 2104 规范的 HMAC-SHA256 算法，要签名的元素是请求自身的 URL、参数和时间戳（毫秒级），由于每个 API 请求基本不同，所以签名的结果也不尽相同。RFC2104 规范请参考：http://www.ietf.org/rfc/rfc2104.txt

2.3.1.2.1 加签原理

以 appSecret 为种子，对一次请求的 URL+Param+Timestamp 进行加签。

1）从协议域“/rest”开始，至传递参数符“？”结束（不包括？），该部分为需加签的 URL 部分。

2）从传递参数符“？”开始至参数结束(包括在 postContent 中的参数) ,该部分为需加签的 Param 部分。请注意，该部分所有参数需要按照参数名字典序进行从小到大排列。

3）协议头部中 wz-acs-timestamp 为需加签的 timestamp 部分。

4）计算完成后的得到 HexString（十六进制字符串），以“_sign=xxxxxxxxx”的参数形式缀于 URL 中。

2.3.1.2.2 加签算法示例（JAVA 代码版）

说明：您可在千寻位置网官网（qxwz.com）的千寻云踪部分下载示例代码。

如下代码为可直接运行的加签示例，包括加签算法、发送 HTTP 请求、处理 HTTP 响应结果等。您需

要注意如下几点内容：

（1）请填写您在千寻官网上创建的应用自动生成 appKey 和 appSecret。

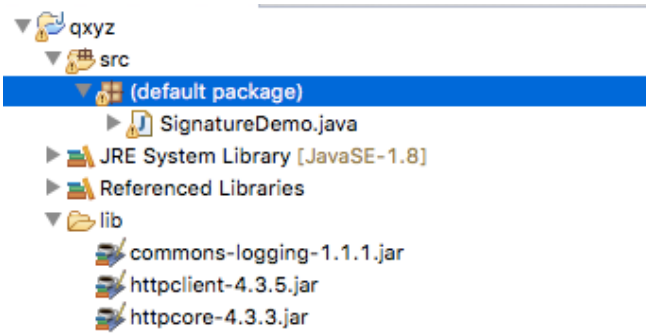
（2）本示例依赖如下包，用于 http 请求

1) commons-logging-1.1.1.jar

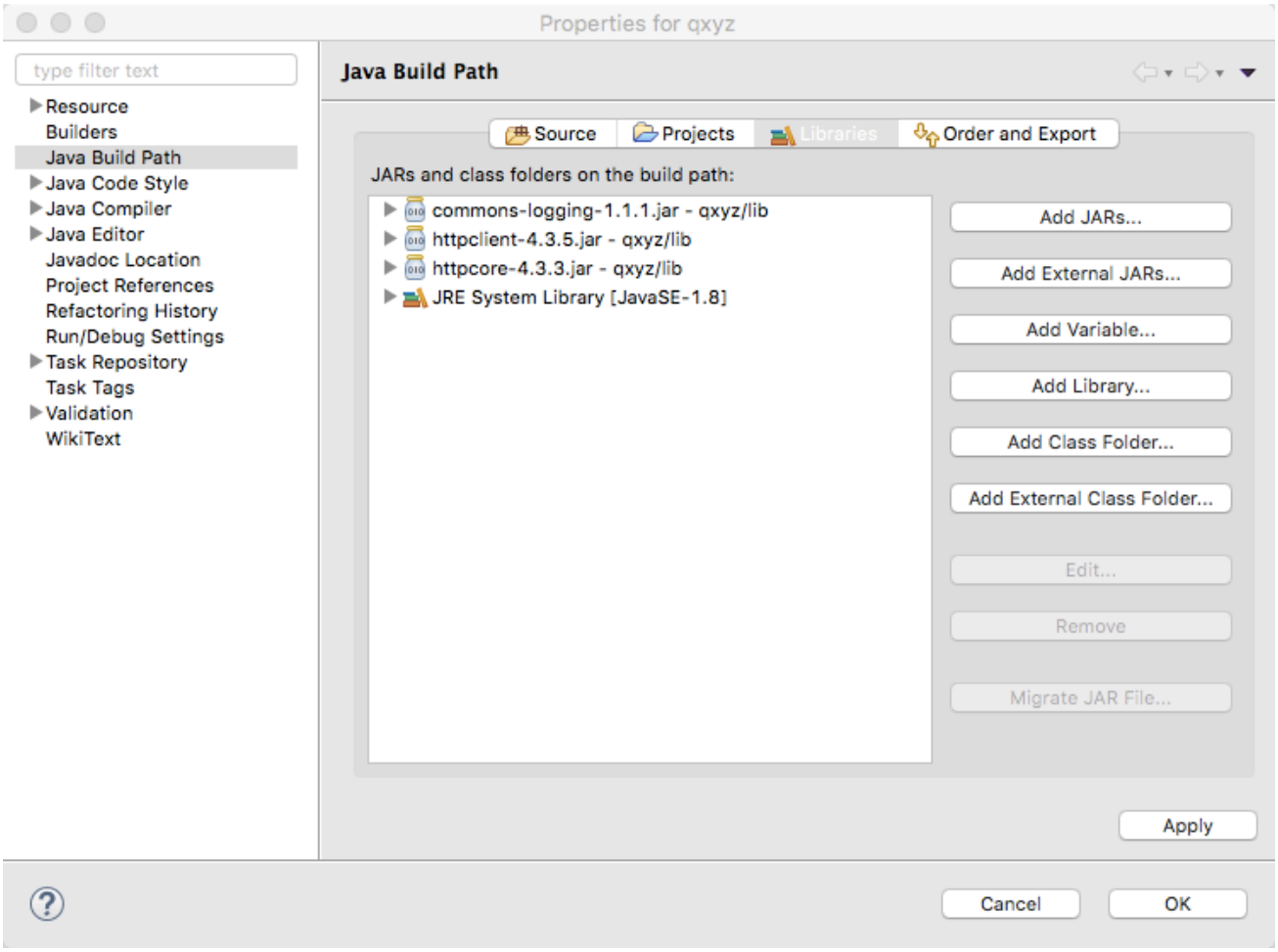
2) httpclient-4.3.4.jar

3）httpcore-4.3.3.jar

附图 1：需要引用的三个包



附图 2：eclipse 中项目的 Properties->Java Build Path->Libraries 中 Add JARs。



（3） 签名算法为 doHmacSHA2，注意时间戳为当前系统时间，单位为毫秒

（4） 需要在请求的 Header 加入 wz-acs-timestamp

（5） 请注意中文编码问题，转换成 utf-8，规避潜在的乱码问题。

```
import java.nio.charset.Charset;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
```

```
import org.apache.http.HttpEntity;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.util.EntityUtils;

public class SignatureDemo {
    public static void main(String[] args) throws Exception{
        String appKey = "appkey_xxx";// 请填写您申请的appkey
        String appSecret = "appSecret_xxx";// 请填写您申请的appSecret
        String apiName = "gpsp.entity.createEntity";      // 访问的API接口名称
        String apiPath = "/rest/" + apiName + "/" + appKey + "/0";// API路径，注意没有问号
        // 参数列表(加签的时候需要进行字典序升序排序)
        Map<String, String> paramMap = new HashMap<String, String>();
         String entity="{\"entityName\":\"沪ABCD\",\"entityType\":\"小汽车\",\"attributes\":{\"color\":\"黑色\",\"driver\":\"千小寻\"}}";
        paramMap.put("entity", entity);
        // 毫秒级时间戳，一定是毫秒级！重要！！！
        String timestamp = String.valueOf(System.currentTimeMillis());
        String signatureStr = SignatureDemo.doHmacSHA2(apiPath, paramMap, appSecret, timestamp);
        System.out.println("加签值："+signatureStr);//打印sign值

        String qxwzUrl = "http://api.qxwz.com";//千寻的服务器地址
        String queryUrl = qxwzUrl + apiPath + "?_sign=" + signatureStr;

        //发送http请求，获取返回值
        doHttpPost(queryUrl,paramMap,timestamp);
    }

    public static void doHttpPost(String url,Map<String, String> paramMap, String timestamp) throws Exception {
        CloseableHttpClient httpClient = HttpClients.createDefault();
        HttpPost httpPost = new HttpPost(url);
        httpPost.setHeader("Accept-Encoding", "gzip, deflate");
        httpPost.setHeader("wz-acs-timestamp", timestamp);//此处注意加上时间戳，否则http请求将无效
        List<NameValuePair> nvps = new ArrayList<NameValuePair>();
        for (Map.Entry<String, String> entry : paramMap.entrySet()){
            if (entry.getValue() == null) continue;
                nvps.add(new BasicNameValuePair(entry.getKey(), entry.getValue()));
```

```
        }
        httpPost.setEntity(new UrlEncodedFormEntity(nvps, "UTF-8"));//处理中文编码的问题

        try {
            CloseableHttpResponse response = (CloseableHttpResponse) httpClient.execute(httpPost);
            try {
                HttpEntity entity = response.getEntity();
                if (entity != null) {
                    System.out.println("返回值："+EntityUtils.toString(entity));
                    /*
                     * 此处针对返回结果进行不同的处理
                     */
                    EntityUtils.consume(entity);
                }
            } finally {
                response.close();
            }

        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            if (httpPost != null) {
                httpPost.releaseConnection();//释放资源
            }
            if(httpClient != null){
                httpClient.close();//释放资源
            }
        }
    }

    /*
     * 将字节数组转换成16进制字符串
     * */
    public static String encodeHexStr(final byte[] bytes){
        if (bytes == null) {
            return null;
        }
        char[] digital    = "0123456789ABCDEF".toCharArray();
        char[] result = new char[bytes.length * 2];
        for (int i = 0; i < bytes.length; i++) {
            result[i * 2] = digital[(bytes[i] & 0xf0) >> 4];
            result[i * 2 + 1] = digital[bytes[i] & 0x0f];
        }
        return new String(result);
```

```
    }

    /*
     * 加签算法
     * */
    public static <T> String doHmacSHA2(String path, Map<String, T> params, String key, String timestamp) {
        List<Map.Entry<String, T>> parameters = new ArrayList<Map.Entry<String,T>>(params.entrySet());
        SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(), "HmacSHA256");
        Charset CHARSET_UTF8 = Charset.forName("UTF-8");
        Mac mac;
        try {
            mac = Mac.getInstance("HmacSHA256");
            mac.init(signingKey);
        } catch (NoSuchAlgorithmException e) {
            throw new IllegalStateException(e.getMessage(), e);
        } catch (InvalidKeyException e) {
            throw new IllegalStateException(e.getMessage(), e);
        }
        if(path != null && path.length() > 0){
            mac.update(path.getBytes(CHARSET_UTF8));
        }
        if(parameters != null){
            Collections.sort(parameters, new MapEntryComparator<String, T>());
            for (Map.Entry<String, T> parameter : parameters) {
                byte[] name = parameter.getKey().getBytes(CHARSET_UTF8);
                Object value = parameter.getValue();
                if(value instanceof Collection){
                    for (Object o : (Collection)value){
                        mac.update(name);
                        if(o != null){
                            mac.update(o.toString().getBytes(CHARSET_UTF8));
                        }
                    }
                }else{
                    mac.update(name);
                    if(value != null){
                        mac.update(value.toString().getBytes(CHARSET_UTF8));
                    }
                }
            }
        }
        if(timestamp != null && timestamp.length() > 0){
            mac.update(timestamp.toString().getBytes(CHARSET_UTF8));
        }
```

```
            return encodeHexStr(mac.doFinal());
        }
}
/*
 * Map参数排序类
 * */
class MapEntryComparator<K, V> implements Comparator<Map.Entry<K, V>> {
        @Override
        public int compare(Entry<K, V> o1, Entry<K, V> o2) {
            if (o1 == o2) {
                return 0;
            }
            final String k1 = o1.getKey().toString();
            final String k2 = o2.getKey().toString();
            int l1 = k1.length();
            int l2a = k2.length();
            for (int i = 0; i < l1; i++) {
                char c1 = k1.charAt(i);
                char c2;
                if (i < l2a) {
                    c2 = k2.charAt(i);
                } else {
                    return 1;
                }
                if (c1 > c2) {
                    return 1;
                } else if (c1 < c2) {
                    return -1;
                }
            }
            if (l1 < l2a) {
               return -1;
            }else if(l1 == l2a) {
               return 0;
            }else {
               return -1;
            }
        }
        }
```

2.3.1.2.3 加签算法示例（Node.JS 代码版）

```
var crypto = require('crypto');
var http = require('http');
var querystring = require('querystring');
```

```
//准备签名数据
var api_name="gpsp.entity.createEntity";
var appKey = "appkey_XXXX";//此处请填写您自己的 appkey
var appSecret="appSecret_XXXX";//此处请填写你自己的 appSecret
var signUrl="/rest/"+api_name+"/"+appKey+"/0";
var signParamObj={
   "entity":'{"entityName":"沪  A1234","entityType":"电动车","attributes":{"color":"黑色","driver":"张三"}}'/*,
   "aTestName":'aTestValue'*/ //如果有多个参数，请以逗号为分隔符进行填写
};
var signTimestamp=new Date().getTime();

//生成签名
var sign = crypto.createHmac("sha256",appSecret);
//1.部分 URL
sign.update(signUrl,'utf8');//必须使用 utf8 编码
//2.参数
var paramNams=Object.keys(signParamObj);
paramNams.sort();//将 url 参数按照参数名称进行升序排序
paramNams.forEach(function (name) {
   sign.update(name,'utf8');
   sign.update(signParamObj[name],'utf8');
   console.log("参数:",name,"=",signParamObj[name]);
})
//3.时间戳
sign.update(signTimestamp+"",'utf8');
var sig = sign.digest('hex');
console.log("签名:",sig);
//至此签名已生成，下文主要描述如何调用 API

//调用 API 示例
var hostname='api.qxwz.com';
var port=80;
function callApi() {
   var signParamData = querystring.stringify(signParamObj);
   var options = {
      hostname: hostname,
      port: port,
      path: signUrl+"?"+"_sign="+sig+"&"+signParamData,
      method: 'GET',
      headers: {
         "wz-acs-method":"HMAC-SHA256",
         "wz-acs-timestamp":signTimestamp
      }
   };
```

```
    console.log("path:",options.path);

    var req = http.request(options, function (res) {
        var buffers = [];
        var nread = 0;
        res.on('data', function (chunk) {
            buffers.push(chunk);
            nread += chunk.length;
        });
        res.on('end', function () {
            var buffer = null;
            switch(buffers.length) {
                case 0: buffer = new Buffer(0);
                    break;
                case 1: buffer = buffers[0];
                    break;
                default:
                    buffer = new Buffer(nread);
                    for (var i = 0, pos = 0, l = buffers.length; i < l; i++) {
                        var chunk = buffers[i];
                        chunk.copy(buffer, pos);
                        pos += chunk.length;
                    }
                    break;
            }
            var data=buffer.toString();
            console.log("结果:",data);
        })
    });
    req.on('error', function (e) {
        console.log('grid problem with request: ' + e.message);
    });
    req.end();
}
callApi();
```

2.3.1.3 返回值

返回值是一个 JSON 对象，包含如下三部分内容。

字段名	含义	值
Code	错误码	0，调用成功，调用结果符合预期

		非 0，调用失败，调用结果不符合预期
Message	错误信息	code 非 0 的时候，会返回一段文字描述，对 code 进行说明。
Data	返回的数据	code 为 0 的时候，data 是实际的返回数据。data 的类型可以是对象（复合类型）或者基本类型（数字、字符串） code 非 0 的时候，data 必须设置为 null。调用方可选择忽略此值。

2.3.1.4 状态码列表

Code	Message	说明
10001	Invalid params	无效参数
12001	Internal error	内部错误
20101	Missing Param	缺少参数
20201	Entity has been created already	实体已存在，请不要重复创建
20202	Entity does not exist	实体不存在
20301	Run out of quota	配额达到上限

2.3.1.5 配额详情

为了更好地为用户提供服务，千寻云踪对每个应用下配额数做了适当限制。如您需要更高的配额，请联系我们申请，服务邮箱是 service@qxwz.com。

名称	配额数	详细说明
实体总数	20	每个应用下，默认提供 20 个实体配额
实体表的自定义字段数	10	每个应用下，实体表支持最多自定义 10 个字段。

轨迹表的自定义字段数	10	每个应用下，轨迹表支持最多自定义 10 个字段。
地理围栏	20	每个应用下，默认提供 20 个地理围栏配额
规则引擎	10	每个应用下，默认提供 10 个规则引擎配额
消息通道	3	每个应用下，默认提供 3 个消息通道配额

2.3.1.6 预设字段

千寻云踪提供一些预设字段，用于为用户提供更多的增值服务；开发者需要确保自定义字段不能如下预设字段重复：

（1）位置点的预设字段：包括经度、纬度、坐标类型、gnss 时间、速度、方向、高程、服务器时间，见于上报位置点部分。

（2）bizType 为预定义字段，目前支持 general、jt808 两个类型，见于创建实体部分。

（3） 以 qx_开始的字段均为千寻内部预设字段，用户的自定义字段不允许以 qx_开头。

2.3.1.7 Lucence 语法

Lucence 是 apache 软件基金会一个开放源代码的全文检索引擎工具包 ,是一个全文检索引擎的架构，提供了完整的查询引擎和索引引擎，部分文本分析引擎。

千寻云踪采用 Lucence 语法来适应多样的查询需求。您可以对各个字段进行各种条件的组合查询，类似于数据库中的查询，当前支持 AND、TO 语法，更多语法将陆续开放。

语法示例：

fieldA:valueA AND fieldB:[range1 TO range2]，其中支持并查询(AND)、支持范围查询(TO)。

参考资料：

http://lucene.apache.org/core/6_2_1/queryparser/org/apache/lucene/queryparser/classic/package-summary.html#package.description

2.3.2 实体管理

2.3.2.1 创建实体

接口名称：gpsp.entity.createEntity

接口描述：创建一个实体，实体可能是任意一个人、车、物，如货车、电动车、可穿戴设备、GPS 跟踪器等。其中 entityName、entityType 为预设字段，同时支持用户自定义扩展字段，满足多样化的业务需要。配额有关内容请参考配额 2.3.1 概览-配额详情部分。

创建实体接口中，业务类型参数 bizType 默认为 general。同一个应用下，实体名称（entityName）+业务类型（bizType）全局唯一。

接口参数：

参数名称	参数类型	参数说明	示例
entity	JSON	实体对象，其中 1.entityName 为实体名称 ,同一个应用下全局唯一，必选。 2.entityType 为实体类型 ,例如“电动车”、“可穿戴设备”、“货车”等，一旦设定不支持更改，必选。 3.attributes 为自定义扩展属性,不允许以 qx_开头。	entity={"entityName":"实体 A","entityType":"电动车","attributes":{"color":"黑色","driver":"张三"}}

返回结果：

返回类型	示例
JSON（正常）	{"code":0,"data":"实体 A"}
JSON（异常）	参照状态码列表

2.3.2.2 删除实体

接口名称：gpsp.entity.deleteEntity

接口描述：删除一个实体

请求参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
entityName	string	实体名称，必选	entityName=实体 A

返回结果：

返回类型	示例
JSON（正常）	{"code":0}
JSON（异常）	参照状态码列表

2.3.2.3 更新实体

接口名称：gpsp.entity.updateEntity

接口描述：更新一个实体的属性信息

请求参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
entityName	string	实体名称，必选	entityName =实体 A
attributes	JSON	属性，必选	attributes={"color":"白色","driver":"李四"}

返回结果：

返回类型	示例
JSON（正常）	{"code":0}
JSON（异常）	参照状态码列表

2.3.2.4 查询单个实体

接口名称：gpsp.entity.queryEntity

接口描述：查询单个实体，返回所有属性

请求参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
entiyName	string	实体名称，必选	entityName=实体 A

返回结果：

返回类型	示例
JSON（正常）	{"code":0,"data":{"entityName":"沪 A1234","entityType":"电动车", "createdTime": 1474180931222,"modifiedTime": 1474180931222, "attributes":{"color":"黑色","driver":"张三"}}}
JSON（异常）	参照状态码列表

2.3.2.5 查询多个实体

接口名称：gpsp.entity.queryEntities

接口描述：查询多个实体，返回每个实体的所有属性

请求参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
entityNames	JSON	实体列表，必选	entityNames=["实体 A","实体 B"]

返回结果：

返回类型	示例
JSON（正常）	{"code":0,"data":[{"entityName":"实体 A","entityType":"电动车","attributes":{"color":"黑色","driver":"张三"}},{"entityName":"实体 B","entityType":"电动车","attributes":{"color":"白色","driver":"李四"}}]}
JSON（异常）	参照状态码列表

2.3.2.6 查询所有实体

接口名称：gpsp.entity.queryEntitiesByApp

接口描述：查询某个业务类型下所有的实体列表，返回每个实体的所有属性。默认按创建时间递减排序。

请求参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持"general"、"jt808"，默认为 general，可选	bizType=general
pageNum	int	当前页数，如果 pageNum<=0，则查询所有的实体，可选	pageNum=2

pageSize	int	分页大小	pageSize=20

返回结果：

返回类型	示例
JSON（正常）	{"code":0,"data":[{"entityName":"沪 A1234","entityType":"电动车", createdTime": 1474180931222,"modifiedTime": 1474180931222, "attributes":{"color":"黑色","driver":"张三"}},{"entityName":"沪 B5678","entityType":"电动车", createdTime": 1474180931222,"modifiedTime": 1474180931222, "attributes":{"color":"白色","driver":"李四"}}]}
JSON（异常）	参照状态码列表

2.3.3 位置管理

2.3.3.1 上传单个轨迹点

接口名称：gpsp.point.addTrackPoint

接口描述：上传单个轨迹点，轨迹点所对应的实体对象必须存在。其中 lon（经度）、lat（纬度）、coordinate（坐标类型）、altitude（高程）、gnssTime（gnss 时间）、speed（速度）为预设字段，同时支持用户自定义扩展字段，满足多样化的业务需要。配额有关内容请参考配额 2.3.1 概览-配额详情部分。此处仅支持 bizType=general 的下的实体上传位置。

接口参数：

参数名称	参数类型	参数说明	示例

entityName	string	实体名称	entityName=实体 A
point	JSON	轨迹点，其中 （1）lon 为经度，必选 （2）lat 为纬度，必选 （3）coordinate 为坐标类型，目前支持 WGS84、GCJ02，默认为 WGS84，可选 （4）gnssTime 为 gnss 时间，Unix 时间戳，单位为毫秒，必选 （5）altitude 为高程，单位为 m，可选 （6）speed 为速度，单位为 m/s，速度大于等于 0，可选 （7）direction 为方向，正北方向，顺时针 0-360 度，单位为度，可选。 （8）attributes 为自定义扩展字段，不允许为预设字段，不允许以 qx_开头。可选	point={"lon":120.12, "lat":40.34, "coordinate":"WGS84", "gnssTime":1464147173225， "altitude":433.01, "speed":4.01, "direction":5.01, attributes:{"temperature":16} }

返回结果：

返回类型	示例
JSON（正确）	{"code":0}
JSON（错误）	参照状态码列表

2.3.3.2 批量上传轨迹点

接口名称：gpsp.point.addTrackPoints

接口描述：批量上传轨迹点，每个轨迹点可以自定义属性并属于属于同一个实体。此处仅支持 bizType=general 的下的实体上传位置。

接口参数：

参数名称	参数类型	参数说明	示例
entityName	string	实体名称，必选。	entityName=实体 A
points	JSON	轨迹点列表，轨迹点参数说明与 addTrackPoint 接口一致。	[{"lon":120.12,"lat":40.34, "coordinate":"WGS84","altitude":433.01,"gnssTime": 1464147173225 ,"speed":4.01,"direction":5.01,attributes:{"temperature":16}},{"lon":120.43,"lat":40.32, ,"coordinate":"WGS84","gnssTime":1464147173225 ,"altitude":233.01,"speed":84.01,"direction":3.0,attributes:{"temperature":26}}]

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0}
JSON（异常）	参照状态码列表

2.3.3.3 查询实时位置

接口名称：gpsp.point.locate

接口描述：查看指定实体列表的实时位置，支持设置活跃时间，返回活跃时间内有位置更新的点。指定的实体列表需要在同一个业务类别（bizType）下。

接口参数：

参数名称	参数类型	参数说明	示例

bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
entityNames	JSON	实体名称列表	entityNames=["实体 A","实体 B"]
activeTime	int	活跃时间，单位为秒，返回活跃时间内有位置更新的点。如果小于等于 0，返回查询存储周期内的最后一个点，必选。 例如 activeTime 为 180 ,那么实体如果在 180s 内没有上传过点，则将不返回该实体的位置。	activeTime=180

返回结果：

返回类型	**示例**
JSON(正确)	{ "code": 0 ,"data": [{"entityName": "实体 A","lon": 120.12, "lat": 40.34,"coordinate": "WGS84", "speed": 4.01, "altitude": 433.01, "direction": 5.01, "gnssTime": 1464147173225, "serverTime": 1464576439150,"attributes": { "temperature": 16}}]} 其中 serverTime 为点存储到服务器的时间。
JSON（错误）	参照状态码列表

2.3.3.4 查询历史轨迹

接口名称：gpsp.point.queryPoints

接口描述：查询指定实体的历史轨迹，支持分页，支持查询存储周期内连续 10 天的历史轨迹。默认按

gnssTime 递增排序。

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
entityName	string	实体名称，必选	entityName=实体 A
beginTime	long	开始时间，Unix 时间戳，单位为毫秒，必填。	beginTime=1464147173220
endTime	long	结束时间，Unix 时间戳，单位为毫秒，必填。	endTime=1464149173325
pageNum	int	当前页数，如果 pageNum<=0，则查询所有的轨迹点，但最大只支持 1000，必选。	pagNum = 2
pageSize	int	分页大小,最大为 1000	pageSize =20

返回结果：

返回类型	示例
JSON(正确)	{ "code": 0 ,"data": [{ "lon": 120.12, ,"lat": 40.34,"coordinate": "WGS84", "speed": 4.01, "altitude": 433.01, "direction":5.01, "gnssTime": 1464147173225, "serverTime": 1464156844094,"attributes": { " temperature ": 16} }, { "lon": 120.34, ,"lat": 40.32,"coordinate": "WGS84", "speed": 44.01, "altitude": 533.01, "direction":23.01, "gnssTime": 1464148273225, "serverTime": 1464148273327,"attributes": { " temperature ": 22} }, { "lon": 120.12, ,"lat": 40.34,"coordinate": "WGS84", "speed": 80.01, "altitude":

	342.01, "direction":35.01, "gnssTime": 1464149273225, "serverTime": 1464149274225,"attributes": { " temperature ": 29} }]}
JSON(错误)	参照状态码列表

2.3.4 配置管理

2.3.4.1 抽稀配置

接口名称：gpsp.config.setSimplifyConfig

接口描述：可以选择是否抽稀、设置抽稀延迟，抽稀程度。轨迹抽稀可以在保持轨迹基本形态前提下，降低存储成本、提高查询效率，但会去掉部分数据。

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808， 默认为 general，可选	bizType=general
simplify	bool	是否选择抽稀，默认不选择抽稀，如果选择不抽稀，后两项设置不会生效，可选	simplify=true
simplifyDelay	int	抽稀延迟天数，目前只支持 1 天、3 天、7 天，该天数后轨迹数据将进行抽稀操作。默认是 3 天，可选	simplifyDelay=3
simplifyLevel	Int	抽稀程度，目前支持 1、2、3，分别对应低、中、高，默认是 3，可选	simplifyLevel=3

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0}
JSON（异常）	参照状态码列表

2.3.4.2 存储配置

接口名称：gpsp.config.setStoreConfig

接口描述：自定义数据存储周期，默认存储 180 天

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
storeCycle	int	存储周期 ,以天为单位 ;默认为 180 天，可选	storeCycle=180

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0 }
JSON（异常）	参照状态码列表

2.3.4.3 地理围栏配置

接口名称：gpsp.config.setFenceConfig

接口描述：配置是否启动地理围栏功能

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
fence	bool	是否启动地理围栏功能，默认为 true，	fence=true

		可选	

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0 }
JSON（异常）	参照状态码列表

2.3.4.4 获取配置表

接口名称：gpsp.config.getConfig

接口描述：获取配置表信息，包括存储配置和抽稀配置。

接口参数：无

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general

返回结果：

返回类型	示例
JSON（正常）	{"code":0,"data":{ "appKey":1000,"simplify":true,"simplifyDelay": 1,"simplifyLevel":2, "storeCycle":180, "fence":true, "quotaPointAttributes": 10,"quotaEntity": 20,"quotaFence": 20,"quotaEntityAttributes": 10,"quotaRule": 10,"quotaChannel": 3}} 其中 （1） quotaEntity 为实体配额数 （2） quotaEntityAttributes，实体可自定义的字段配额数量

	（3） quotaPointAttributes，轨迹点可自定义的字段配额数量 （4） quotaFence，可创建的地理围栏配额数量 （5） quotaRule，可创建的规则配额数量 （6） quotaChannel，可创建的消息通道配额数量
JSON（异常）	参照状态码列表

2.3.5 事件管理

千寻云踪支持用户自主上传基于位置的事件，例如变道、前车碰撞预警、行人碰撞预警等各类事件。你可以结合位置管理进行综合分析。

2.3.5.1 创建事件

接口名称：gpsp.event.createEvent

接口描述：创建一个事件，用来记录一个实体在某个时间某个地点发生了某件事。

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
entityName	string	实体名称，必选	entityName=实体 A
eventName	string	事件名称，可选	eventName=偏离 007
eventType	string	事件类型，必选，当前支持下列事件： （1） qx_ldw_left：车道左偏离警告 （2） qx_ldw_right：车道右偏离警告	eventType=qx_ldw_left

		（3） qx_fcw：前车碰撞预警 （4） qx_pcw：行人碰撞预警 （5） qx_ rapidAcceleration：急加速 （6） qx_rapidDeceleration：急减速 （7） qx_sharpTurn：急转弯 （8） qx_overSpeed：超速 支持用户自定义事件，但不允许以 qx_开头。	
eventTime	long	事件时间，Unix 时间戳，单位为毫秒，必选	eventTime= 1464147173225
attributes	JSON	自定义属性，支持可扩展字段，扩展字段不得以 qx_开头。其中 （1） qx_lon 为经度，预设字段，必选 （2） qx_lat 为纬度，预设字段，必选 （3） qx_coordinate 为坐标类型，预设字段，目前支持 WGS84 和 GCJ02 两种。默认为 WGS84，可选	attributes={"qx_lon":121 .468434,"qx_lat":31.2065 75,"address":"上海市黄浦 区徐家路日月光中心"}

返回结果：

返回类型	**示例**
JSON（正常）	{"code":0,"data": 1000079 }返回行程的唯一标识符 eventId，数据类型为 long
JSON（异常）	参照状态码列表

2.3.5.2 删除事件

接口名称：gpsp.event.deleteEvent

接口描述：删除一个事件

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
eventId	long	事件唯一标识符，必选	eventId=1000079

返回结果：

返回类型	示例
JSON（正常）	{"code":0}
JSON（异常）	参照状态码列表

2.3.5.3 更新事件

接口名称：gpsp.event.updateEvent

接口描述：更新一个事件

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
eventId	long	事件 Id，必选	eventId=1000079
eventName	string	事件名称，可选	eventName=偏离 007
eventType	string	事件类型，可选，当前支持下列事件： （1）qx_ldw_left：车道左偏离警告	eventType=qx_ldw_right

		（2） qx_ldw_right：车道右偏离警告 （3） qx_fcw：前车碰撞预警 （4） qx_pcw：行人碰撞预警 （5） qx_ rapidAcceleration：急加速 （6） qx_rapidDeceleration：急减速 （7） qx_sharpTurn：急转弯 （8） qx_overSpeed：超速 支持用户自定义事件，但不允许以 qx_开头。	
eventTime	long	事件时间，Unix 时间戳，单位为毫秒，可选	eventTime= 1464147173225
attributes	JSON	自定义属性，支持可扩展字段，扩展字段不得以 qx_开头。其中 （4） qx_lon 为经度，预设字段，可选 （5） qx_lat 为纬度，预设字段，可选 qx_coordinate 为坐标类型，预设字段，目前支持 WGS84 和 GCJ02 两种。默认为 WGS84，可选	attributes={"qx_lon":121.468434,"qx_lat":31.206575,"address":"上海市黄浦区徐家路日月光中心"}

3. 返回结果：

返回类型	**示例**
JSON（正常）	{"code":0}
JSON（异常）	参照状态码列表

2.3.5.4 查询单个事件

接口名称：gpsp.event.queryEvent

接口描述：更新一个事件

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
eventId	long	事件 Id，必选	eventId=1000079

4. 返回结果：

返回类型	示例
JSON（正常）	{"code":0,"data": {"bizType":"general","eventId": 1000079, "eventName":"偏离007","entityName":"实体A","eventType":"qx_ldw_right" ,"eventTime":"1464147173225","attributes":{"qx_lon":121.468434,"qx_lat":31.206575,"address":"上海市黄浦区徐家路日月光中心"}}}
JSON（异常）	参照状态码列表

3.3.5.5 查询事件列表

接口名称：gpsp.event.queryEvents

接口描述：查询指定时间段的事件列表，支持分页，默认按创建时间递减排序。

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general

queryText	string	查询条件，该参数使用 Lucene 定义的语法来表述查询条件。关于 Lucence 的详情，请查看 2.3.1 概览-Lucence 章节了解详情，当前仅支持 AND、TO。 查询条件支持的参数如下： （1） entityName，string 类型，实体名称，可选 （2） eventName，string 类型，事件名称，可选 （3） eventType，string 类型，事件类型，可选 （4） eventTime，string 类型，事件时间段，Unix 时间戳，单位为毫秒，可选。	queryText=entityName:实体 A AND eventName:偏离 007 AND eventType:qx_ldw_right AND eventTime:[1464147173220 TO 1464149173325]
pageNum	int	当前页数，如果 pageNum<=0，则查询所有的行程。	pageNum=2
pageSize	int	分页大小	pageSize=20

返回结果：

返回类型	**示例**
JSON（正常）	{"code":0, "data":[{"bizType":"general","eventId": 1000079, "eventName":"偏离 007","entityName":"实体 A","eventType":"qx_ldw_right" ,"eventTime":"1464147173225","attributes":{"qx_lon":121.468434,"qx_lat":31.206575,"address":"上海市黄浦区徐家路日月

	光中心"}},{"bizType":"general","eventId": 1000080, "eventName":"偏离008","entityName":"实体A","eventType":"qx_ldw_left" ,"eventTime":"1464147174225","attributes":{"qx_lon":121.468534,"qx_lat":31.206585,"address":"上海市黄浦区徐家路日月光中心"}}]}
JSON（异常）	参照状态码列表

2.3.6 行程管理

通过行程管理，您可以对轨迹进行分时段管理，比如记录一次骑行、一次跑步，一次行车等。结合位置管理，您可以

（1） 查询指定实体的某个行程的起始时间

（2） 根据该实体及其起始时间查询轨迹

（3） 对该段轨迹进行展示、统计分析，也可将统计分析的结果自行更新到行程的扩展字段中。

2.3.6.1 创建行程

接口名称：gpsp.trip.createTrip

接口描述：创建一个行程，用于记录一次骑行、一次跑步，一次行车等。

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
entityName	string	实体名称，必选	entityName=实体 A
tripName	string	行程名称，可选	tripName=跑步 007

startTime	long	开始时间，Unix 时间戳，单位为毫秒，必填。	startTime=1464147173225
endTime	long	结束时间，Unix 时间戳，单位为毫秒，必填。	endTime=1464147373225
attributes	JSON	自定义属性，支持可扩展字段，扩展字段不得以 qx_开头。其中 （1）qx_lon_start 为起点的经度，预设字段，必选 （2）qx_lat_start 为起点的纬度，预设字段，必选 （3）qx_coordinate_start 为起点的坐标类型，预设字段，目前支持 WGS84 和 GCJ02 两种，默认为 WGS84，可选 （4）qx_lon_end 为终点的经度，预设字段，必选 （5）qx_lat_end 为终点纬度，预设字段，必选 （6）qx_coordinate_end 为终点的坐标类型，预设字段，目前支持 WGS84 和 GCJ02 两种，默认为 WGS84，可选	attributes ={"qx_lon_start":121.468434, "qx_lat_start":31.206575,"qx_coordinate_start":"WGS84","address_start":"上海市黄浦区徐家路日月光中心","qx_lon_end":121.514158,"qx_lat_end":31.299059,"qx_coordinate_end":"WGS84","address_end":"上海市杨浦区五角场"}

返回结果：

返回类型	**示例**

JSON（正常）	{"code":0, "data":1000023}，返回行程的唯一标识符 tripId，数据类型为 long
JSON（异常）	参照状态码列表

2.3.6.2 根据名称创建行程

接口名称：gpsp.trip.createUniqueTrip

接口描述：根据名称创建一个行程，用于记录一次骑行、一次跑步，一次行车等；适用于需保持行程名称唯一的用户。如果行程名称已存在，则无法创建成功。

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
entityName	string	实体名称，必选	entityName=实体 A
tripName	string	行程名称，必选。	tripName=跑步 007
startTime	long	开始时间，Unix 时间戳，单位为毫秒，必填。	startTime=1464147173225
endTime	long	结束时间，Unix 时间戳，单位为毫秒，必填。	endTime=1464147373225
attributes	JSON	自定义属性，支持可扩展字段，扩展字段不得以 qx_开头。其中 （1）qx_lon_start 为起点的经度，预设字段，必选 （2）qx_lat_start 为起点的纬度，预设	attributes ={"qx_lon_start":121.468434, "qx_lat_start":31.206575,"qx_coordinate_start":"WGS84","address_start":"上海市黄浦区徐家路

		字段，必选 （3）qx_coordinate_start 为起点的坐标类型，预设字段，目前支持 WGS84 和 GCJ02 两种，默认为 WGS84，可选 （4）qx_lon_end 为终点的经度，预设字段，必选 （5）qx_lat_end 为终点纬度，预设字段，必选 （6）qx_coordinate_end 为终点的坐标类型，预设字段，目前支持 WGS84 和 GCJ02 两种，默认为 WGS84，可选	日月光中心","qx_lon_end":121.514158,"qx_lat_end":31.299059,"qx_coordinate_end":"WGS84","address_end":"上海市杨浦区五角场"}

返回结果：

返回类型	示例
JSON（正常）	{"code":0, "data":1000024}，返回行程唯一标识符 tripId。
JSON（异常）	参照状态码列表

2.3.6.3 删除行程

接口名称：gpsp.trip.deleteTrip

接口描述：删除一个行程

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general

tripId	long	行程 Id，必选。	tripId=1000023

返回结果：

返回类型	示例
JSON（正常）	{"code":0}
JSON（异常）	参照状态码列表

2.3.6.4 根据名称删除行程

接口名称：gpsp.trip.deleteTripByName

接口描述：删除一个行程

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808， 默认为 general，可选	bizType=general
tripName	string	行程名称，必选。	tripName=跑步 007

返回结果：

返回类型	示例
JSON（正常）	{"code":0}
JSON（异常）	参照状态码列表

2.3.6.5 更新行程

接口名称：gpsp.trip.updateTrip

接口描述：更新一个行程

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
tripId	long	行程 ID，必选	tripId=1000023
tripName	string	行程名称，可选	tripName=跑步 008
startTime	long	开始时间，Unix 时间戳，单位为毫秒，可选。	startTime=1464147173225
endTime	long	结束时间，Unix 时间戳，单位为毫秒，可选。	endTime=1464147373225
attributes	JSON	自定义属性，支持可扩展字段，扩展字段不得以 qx_开头。其中 （1）qx_lon_start 为起点的经度，预设字段，可选 （2）qx_lat_start 为起点的纬度，预设字段，可选 （3）qx_coordinate_start 为起点的坐标类型，预设字段，目前支持 WGS84 和 GCJ02 两种，默认为 WGS84，可选 （4）qx_lon_end 为终点的经度，预设字段，可选 （5）qx_lat_end 为终点纬度，预设字段，可选	attributes ={"qx_lon_start":121.468434, "qx_lat_start":31.206575,"qx_coordinate_start":"WGS84","address_start":"上海市黄浦区徐家路日月光中心","qx_lon_end":121.514158,"qx_lat_end":31.299059,"qx_coordinate_end":"WGS84","address_end":"上海市杨浦区五角场"}

		（6）qx_coordinate_end 为终点的坐标类型，预设字段，目前支持 WGS84 和 GCJ02 两种，默认为 WGS84，可选	

返回结果：

返回类型	示例
JSON（正常）	{"code":0}
JSON（异常）	参照状态码列表

2.3.6.6 根据名称更新行程

接口名称：gpsp.trip.updateTripByName

接口描述：更新一个行程

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
tripName	string	行程名称，必选	tripName=跑步 007
startTime	long	开始时间，Unix 时间戳，单位为毫秒，可选。	startTime=1464147173225
endTime	long	结束时间，Unix 时间戳，单位为毫秒，可选。	endTime=1464147373225
attributes	JSON	自定义属性，支持可扩展字段，扩展字段不得以 qx_开头。其中 （1）qx_lon_start 为起点的经度，预设	attributes ={"qx_lon_start":121.468434, "qx_lat_start":31.206575,"qx_co

		字段，可选 （2）qx_lat_start 为起点的纬度，预设字段，可选 （3）qx_coordinate_start 为起点的坐标类型，预设字段，目前支持 WGS84 和 GCJ02 两种，默认为 WGS84，可选 （4）qx_lon_end 为终点的经度，预设字段，可选 （5）qx_lat_end 为终点纬度，预设字段，可选 （6）qx_coordinate_end 为终点的坐标类型，预设字段，目前支持 WGS84 和 GCJ02 两种，默认为 WGS84，可选	ordinate_start":"WGS84","address_start":"上海市黄浦区徐家路日月光中心","qx_lon_end":121.514158,"qx_lat_end":31.299059,"qx_coordinate_end":"WGS84","address_end":"上海市杨浦区五角场"}

返回结果：

返回类型	示例
JSON（正常）	{"code":0}
JSON（异常）	参照状态码列表

2.3.6.7 查询单个行程

接口名称：gpsp.trip.queryTrip

接口描述：查询一个行程

接口参数：

参数名称	参数类型	参数说明	示例

bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
tripId	long	行程 Id，必选	tripId=1000023

返回结果：

返回类型	示例
JSON（正常）	{"code":0, "data":{"bizType":"general","tripId":1000023,"entityName":"实体A","tripName":"跑步007","startTime":"1464147173225","endTime":"1464147373225","attributes":{"qx_lon_start":121.468434,"qx_lat_start":31.206575,"qx_coordinate_start":"WGS84","address_start":"上海市黄浦区徐家路日月光中心","qx_lon_end":121.514158,"qx_lat_end":31.299059,"qx_coordinate_end":"WGS84","address_end":"上海市杨浦区五角场"}}}
JSON（异常）	参照状态码列表

2.3.6.8 根据名称查询单个行程

接口名称：gpsp.trip.queryTripByName

接口描述：查询一个行程

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
tripName	string	行程名称，必选	tripName=跑步 007

返回结果：

返回类型	示例
JSON（正常）	{"code":0, "data":{"bizType":"general","tripId":1000023,"entityName":"实体A","tripName":"跑步007","startTime":"1464147173225","endTime":"1464147373225","attributes":{"qx_lon_start":121.468434,"qx_lat_start":31.206575,"qx_coordinate_start":"WGS84","address_start":"上海市黄浦区徐家路日月光中心","qx_lon_end":121.514158,"qx_lat_end":31.299059,"qx_coordinate_end":"WGS84","address_end":"上海市杨浦区五角场"}}}
JSON（异常）	参照状态码列表

2.3.6.9 查询行程列表

接口名称：gpsp.trip.queryTrips

接口描述：查询行程列表，支持分页。默认按创建时间递减排序。

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型 ,目前支持 general、jt808 ,默认为 general，可选	bizType=general
queryText	string	查询条件，该参数使用 Lucene 定义的语法来表述查询条件。关于 Lucence 的详情，请查看 2.3.1 概览-Lucence 章节了解详情，当前仅支持 AND、TO。 查询条件支持的参数如下： （1）entityName，string 类型，实体名称，可选	示例一： queryText=entityName:实体 A 示例二： queryText=beginTime:[146414717

		（2）tripName，string 类型，行程名称，可选 （3）startTime，long 类型，开始时间，Unix 时间戳，单位为毫秒，可选。 （4）endTime，long 类型，结束时间，Unix 时间戳，单位为毫秒，可选。	3220 TO 1464149173325]
pageNum	int	当前页数，如果 pageNum<=0，则查询所有的行程。	pageNum=2
pageSize	int	分页大小	pageSize=20

返回结果：

返回类型	示例
JSON（正常）	{"code":0, "data":[{"bizType":"general","tripId":1000023,"entityName":"实体 A","tripName":"跑步 007","startTime":"1464147173225","endTime":"1464147373225","attributes":{"qx_lon_start":121.468434,"qx_lat_start":31.206575,"qx_coordinate_start":"WGS84","address_start":"上海市黄浦区徐家路日月光中心","qx_lon_end":121.514158,"qx_lat_end":31.299059,"qx_coordinate_end":"WGS84","address_end":"上海市杨浦区五角场"}},{"bizType":"general","tripId":1000024,"entityName":"实体 A","tripName":"跑步 008","startTime":"1464147173225","endTime":"1464147373225","attributes":{"qx_lon_start":121.468434,"qx_lat_start":31.206575,"qx_coordinate_start":"WGS84","address_start":"上海市黄浦区徐家路日月光中心","qx_lon_end":121.514158,"qx_lat_end":31.299059,"qx_coordinate_end":"WGS84","address_end":"上海市杨浦区五角场"}}]}

JSON（异常）	参照状态码列表

2.3.7 地理围栏

地理围栏是位置服务的基础服务，如需使用地理围栏服务，需要完成如下操作：

（1） 实体管理：创建实体

（2） 地理围栏：创建地理围栏

（3） 关系管理：将实体与地理围栏绑定

（4） 规则引擎：创建规则，并将规则与应用绑定

（5） 配置管理：在地理围栏设置中，将地理围栏开关打开

2.3.7.1 创建地理围栏

接口名称：gpsp.fence.createFence

接口描述：创建地理围栏，支持多边形、圆形、行政区划等多种类型，当前版本仅只支持多边形。配额有关内容请参考配额 2.3.1 概览-配额详情部分。

接口参数：

参数名称	参数类型	参数说明	示例
fenceName	string	地理围栏名称，同一个应用下全局唯一，必选	fenceName=地理围栏 A
fenceType	string	地理围栏类型，当前仅支持多边形(POLYGON)	fenceType=POLYGON
fenceDetail	JSON	地理围栏详情，针对不同的类型参数详情如下, 1.多边形（POLYGON） （1）geometry：图形对象，参照 WKT 格式，目前仅支持 POLYGON。 （2）coordinate：坐标类型，目前支持 WGS84、GCJ02，默认为 WGS84。	fenceDetail={"geometry":"POLYGON((120.10 40.20, 120.20 40.20, 120.20 40.10,120.10 40.10, 120.10 40.20))","coordinate":

			"WGS84"}}

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0, "data":"地理围栏 A"}
JSON（异常）	参照状态码列表

2.3.7.2 删除地理围栏

接口名称：gpsp.fence.deleteFence

接口描述：删除一个地理围栏，删除之前需要先解除与实体的绑定关系，否则无法删除。

接口参数：

参数名称	参数类型	参数说明	示例
fenceName	string	地理围栏名称，同一个应用下全局唯一，必选	fenceName=地理围栏 A

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0}
JSON（异常）	参照状态码列表

2.3.7.3 更新地理围栏

接口名称：gpsp.fence.updateFence

接口描述：更新一个地理围栏，如果当前地理围栏已经绑定，无法更新。

接口参数：

参数名称	参数类型	参数说明	示例

fenceName	string	地理围栏名称，同一个应用下全局唯一，必选	fenceName=地理围栏 A
fenceType	string	地理围栏类型，当前仅支持多边形(POLYGON)	fenceType=POLYGON
fenceDetail	JSON	地理围栏详情，针对不同的类型参数详情如下, 1.多边形（POLYGON） （1）geometry：图形对象，参照 WKT 格式，目前仅支持 POLYGON。 （2）coordinate：坐标类型，目前支持 WGS84、GCJ02。 备注：如果某个参数未填写，则保持原来数据不变。	fenceDetail={"geometry":"POLYGON((121.10 39.20, 121.20 39.20, 121.20 39.10,121.10 39.10, 121.10 39.20))" ,"coordinate": "WGS84"}}

返回结果：

返回类型	**示例**
JSON（正常）	{"code":0 }
JSON（异常）	参照状态码列表

2.3.7.4 查询单个地理围栏

接口名称：gpsp.fence.queryFence

接口描述：查询单个地理围栏

接口参数：

参数名称	**参数类型**	**参数说明**	**示例**
fenceName	String	地理围栏名称，同一个应用下全局唯一，必选	fenceName=地理围栏 A

返回结果：

返回类型	示例
JSON（正常）	{"code":0, "data":{ "fenceName":"地理围栏 A","fenceType":"POLYGON", "fenceDetail":{"geometry":"POLYGON((120.10 40.20, 120.20 40.20, 120.20 40.10,120.10 40.10, 120.10 40.20)) " ,"coordinate": "WGS84"}}
JSON（异常）	参照状态码列表

2.3.7.5 查询所有地理围栏

接口名称：gpsp.fence.queryFencesByApp

接口描述：查询所有的地理围栏，支持分页。默认按创建时间递减排序。

接口参数：

参数名称	参数类型	参数说明	示例
pageNum	int	当前页数，如果 pageNum<=0，则查询所有的地理围栏。	pageNum=2
pageSize	int	分页大小	pageSize=20

返回结果：

返回类型	示例
JSON（正常）	{"code":0, "data":[{ "fenceName":"地理围栏 A","fenceType":"POLYGON","fenceDetail":{"geometry": "POLYGON((120.10 40.20, 120.20 40.20, 120.20 40.10,120.10 40.10, 120.10 40.20))",coordinate": "WGS84"} },{ "fenceName":"地理围栏 B","fenceType":"POLYGON","fenceDetail":{"geometry": "POLYGON((121.10 30.20, 121.20 30.20, 121.20 30.10,121.10 30.10, 121.10 30.20))",coordinate": "WGS84"} }]}

JSON（异常）	参照状态码列表

2.3.8 规则引擎

2.3.8.1 创建规则

接口名称：gpsp.rule.createRule

接口描述 :创建一个规则，目前仅支持地理围栏规则。配额有关内容请参考配额 2.3.1 概览-配额详情部分。

接口参数：

参数名称	参数类型	参数说明	示例
ruleName	string	规则名称，作为此规则的唯一标识。同一个应用下全局唯一，必选	ruleName=规则 A
ruleType	string	规则类型，目前仅支持地理围栏规则 FENCE，必选。	ruleType=FENCE
ruleDetail	JSON	规则详情，详情如下： 1. 地理围栏规则（fence）: （1） moveIn：进围栏，默认 false，可选 （2） moveOut：出围栏，默认为 false，可选 （3） allTime：bool 类型，默认为 true，代表全部时间生效，其他时间相关的设置将不生效。如果为 false，则根据用户的时间配置，可选 （4） startTime：开始时间，不设置代表从此时开始生效，可选 （5） endTime：结束时间，不设置代表开始后一直生效，可选	ruleDetail={"moveIn":true, "moveOut":true,"allTime":false,"startTime":1464147173225}

channels	JSON	消息通道名称列表，触发规则产生的信息将写入消息通道。 （1） 了解消息通道，请查看消息通道章节 （2） 目前消息通道仅支持阿里云产品 MNS （3） 触发的消息格式为： {"messageType":"FENCE","entityName":" 实体 A","fenceName":" 地理围栏 A" ,"ruleName":" 规则 A","eventTime":"1464147174227", "status":"MOVEIN","pointDetail":{"lat":120.02, "lon":"40.01","coordinate":"WGS84","gnssTime":1464147174225}}	channels=["消息通道 A"]

返回结果：

返回类型	**示例**
JSON（正常）	正确结果 {"code":0,data: "规则 A "}
JSON（异常）	参照状态码列表

2.3.8.2 删除规则

接口名称：gpsp.rule.deleteRule

接口描述：删除一条规则，删除规则前需要先解除规则与应用等的绑定关系，否则删除不成功。

接口参数：

参数名称	**参数类型**	**参数说明**	**示例**

ruleName	String	规则名称，必选	ruleName=规则 A

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0}
JSON（异常）	参照状态码列表

2.3.8.3 更新规则

接口名称：gpsp.rule.udpateRule

接口描述：更新一条规则

接口参数：

参数名称	参数类型	参数说明	示例
ruleName	string	规则名称，作为此规则的唯一标识，必选	ruleName=规则 A
ruleType	string	规则类型，目前仅支持地理围栏规则 FENCE，必选	ruleType=FENCE
ruleDetail	string	规则详情，详情如下： 1.地理围栏规则（fence）: （1） moveIn：进围栏，默认 false，可选 （2）moveOut：出围栏，默认为 false，可选 （3）allTime：bool 类型，默认为 true，代表全部时间生效，其他设置将不生效。如果为 false，则根据用户的时间配置，可选 （4）startTime：开始时间，不设置代表从此	ruleDetail={"moveIn":true, "moveOut":true,"allTime":false,"startTime":1464147173225}

		时开始生效，可选 （5）endTime：结束时间，不设置代表开始后一直生效，可选	
channels	JSON	消息通道名称列表，触发规则产生的信息将写入写入消息通道	channels=["消息通道 A"]

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0 }
JSON（异常）	参照状态码列表

2.3.8.4 查询单个规则

接口名称：gpsp.rule.queryRule

接口描述：查询单个规则

接口参数：

参数名称	参数类型	参数说明	示例
ruleName	string	规则名称，必选	ruleName=规则 A

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0, "data":{"ruleName":"规则 A","ruleType":"POLYGON","ruleDetail":{moveIn":true, "moveOut":true,"allTime":false,"startTime":1464147173225}, "channels":["消息通道 A","消息通道 B"]}}}

JSON（异常）	参照状态码列表

2.3.8.5 查询所有规则

接口名称：gpsp.rule.queryRulesByApp

接口描述：查询所有的规则，支持分页，默认按创建时间递减排序。

接口参数：

参数名称	参数类型	参数说明	示例
pageNum	int	当前页数，如果 pageNum<=0，则查询所有的规则，必选	pageNum=2
pageSize	int	分页大小	pageSize=20

返回结果：

返回类型	示例
JSON(正常)	正确结果 {"code":0, "data":[{"ruleName":"规则 A","ruleType":"FENCE","ruleDetail":{moveIn":true, "moveOut":true,"allTime":false,"startTime":1464147173225, "channels":["消息通道 A"]},"ruleName":"规则 B","ruleType":"FENCE","ruleDetail":{moveIn":true, "moveOut":true,"allTime":false,"startTime":1464147173225}, "channels":["消息通道 A"]}]}
JSON(异常)	参照状态码列表

2.3.9 消息通道

2.3.9.1 创建消息通道

接口名称：gpsp.channel.createChannel

接口描述：创建一个消息通道，目前仅支持阿里云消息通道服务 MNS，未来支持钉钉、微信、短信、邮件。

如需支持其他消息通道，请直接联系我们。配额有关内容请参考配额 2.3.1 概览-配额详情部分。

接口参数：

参数名称	参数类型	参数说明	示例
channelName	string	消息通道名称，同一个应用下全局唯一，必选	channelName=消息通道 A
channelType	string	消息通道类型，目前仅支持阿里云的消息服务（MNS），必选	channelType=MNS
channelDetail	JSON	消息通道详情，针对不同的类型参数详情如下， 1.阿里云的消息服务（MNS） （1）endPoint：服务节点，必选 （2）accessId：标识，必选 （3）accessKey：密匙，必选 （4）queueName：队列名称，必选	channelDetail={ "endPoint": "http://xx.mns.cn-shanghai.aliyuncs.com/", "accessId":"xx", "accessKey":"xx", "queueName":"queue-xx"}

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0, "data":"消息通道 A"}
JSON（异常）	参照状态码列表

2.3.9.2 删除消息通道

接口名称：gpsp.channel.deleteChannel

接口描述：删除消息通道，删除消息通道会强制删除消息通道与规则的绑定关系。

接口参数：

参数名称	参数类型	参数说明	示例
channelName	String	消息通道名称，同一个应用下全局唯一，必选	channelName=消息通道 A

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0}
JSON（异常）	参照状态码列表

2.3.9.3 更新消息通道

接口名称：gpsp.channel.updateChannel

接口描述：更新消息通道

接口参数：

参数名称	参数类型	参数说明	示例
channelName	String	消息通道名称，同一个应用下全局唯一，必选	channelName=消息通道 A
channelType	String	消息通道类型，目前仅支持阿里云的消息服务（MNS），必选	channelType=MNS
channelDetail	JSON	消息通道详情，针对不同的类型参数详情如下， 1.阿里云的消息服务（MNS）	channelDetail={"endPoint": "http://yy.mns.cn-shanghai.aliyuncs.com/",

		（1）endPoint：服务节点，可选 （2）accessId：标识，可选 （3）accessKey：密匙，可选 （4）queueName :队列名称，可选	"accessId":"yy", "accessKey ":"yy", "queueName":"queue-yy"}

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0}
JSON（异常）	参照状态码列表

2.3.9.4 查询单个消息通道

接口名称：gpsp.channel.queryChannel

接口描述：删除消息通道

接口参数：

参数名称	参数类型	参数说明	示例
channelName	string	消息通道名称，同一个应用下全局唯一，必选	channelName=消息通道 A

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0, "data":{"channelName":"消息通道 A"， "channelType":"MNS","channelDetail":{" "endPoint":

	"http://xx.mns.cn-shanghai.aliyuncs.com/", "accessId":"xx", "accessKey ":"xx", "queueName":"queue-xx"}}}
JSON（异常）	参照状态码列表

2.3.9.5 查询所有消息通道

接口名称：gpsp.channel.queryChannelsByApp

接口描述：查询所有消息通道，支持分页，默认按创建时间递减排序。

接口参数：

参数名称	参数类型	参数说明	示例
pageNum	int	当前页数，如果 pageNum<=0，则查询所有， 必选	pageNum=2
pageSize	int	分页大小	pageSize=20

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0, "data":[{"channelName":"消息通道 A"， "channelType" :"MNS","channelDetail":{"endPoint":"http://xx.mns.cn-shanghai.al iyuncs.com/","accessId":"xx"," accessKey":"xx","queueName":"queue-xx"},{"channelName":"消息通道 B"， "channelType" :"MNS","channelDetail":{"endPoint":"http://yy.mns.cn-shanghai.al iyuncs.com/","accessId":"yy","accessKey":"yy","queueName":"queue-yy"}}}}
JSON（异常）	参照状态码列表

2.3.10 关系管理

2.3.10.1 创建全局绑定关系

接口名称：gpsp.relation.bindApp

接口描述：将地理围栏、规则引擎对应用绑定，全局生效。如果将地理围栏与应用绑定，相当于地理围栏与所有实体绑定；如果将规则引擎与应用绑定，相当于规则引擎对所有实体生效。

接口参数：

参数名称	参数类型	参数说明	示例
type	string	绑定对象类型，目前支持 FENCE、RULE，分别对应地理围栏、规则引擎，必选	type=RULE
name	string	绑定对象名称，必选。	name=规则 A

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0}
JSON（异常）	参照状态码列表

2.3.10.2 解除全局绑定关系

接口名称：gpsp.relation.unbindApp

接口描述：解除全局绑定关系

接口参数：

参数名称	参数类型	参数说明	示例
type	string	绑定对象类型，目前支持 FENCE、RULE，分别对应地理围栏、规则引擎，必选	type=RULE

name	string	绑定对象名称，必选	name=规则 A

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0}
JSON（异常）	参照状态码列表

2.3.10.3 查询单个全局绑定关系

接口名称：gpsp.relation.queryAppBind

接口描述：查询单个全局绑定关系

接口参数：

参数名称	参数类型	参数说明	示例
type	string	绑定对象类型，目前支持 FENCE、RULE，分别对应地理围栏、规则引擎，必选	type=RULE
name	string	绑定对象名称，必选	name=规则 A

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0, "data":0}；0 代表未绑定，1 代表绑定
JSON（异常）	参照状态码列表

2.3.10.4 查询所有的全局绑定关系

接口名称：gpsp.relation.queryAppBinds

接口描述：查询所有的全局绑定关系，支持分页

接口参数：

参数名称	参数类型	参数说明	示例
type	string	绑定对象类型，目前支持 FENCE、RULE，分别对应地理围栏、规则引擎，必选	type=RULE
pageNum	int	当前页数 ,如果 pageNum<=0 ,则查询所有的绑定关系。	pageNum=2
pageSize	int	分页大小	pageSize=20

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0, "data":["规则 A","规则 B"]}
JSON（异常）	参照状态码列表

2.3.10.5 创建单个实体与单个地理围栏的绑定关系

接口名称：gpsp.relation.bindEntityAndFence

接口描述：创建实体与地理围栏的绑定关系，支持进行一对一绑定。

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
entityName	string	实体名称，必选	entityName=实体 A
fenceName	string	地理围栏名称，必选	fenceName =地理围栏 A

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0}

JSON（异常）	参照状态码列表

2.3.10.6 解除单个实体与单个地理围栏的绑定关系

接口名称：gpsp.relation.unbindEntityAndFence

接口描述：解除单个实体与单个地理围栏的绑定关系

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
entityName	string	实体名称，必选	entityName=实体 A
fenceName	string	地理围栏名称，必选	fenceNames=地理围栏 A

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0}
JSON（异常）	参照状态码列表

2.3.10.7 查询与某个实体绑定的所有地理围栏

接口名称：gpsp.relation.querybindFencesByEntity

接口描述：查询与某个实体绑定的所有地理围栏

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
entityName	string	实体名称，必选	entityName=实体 A

pageNum	int	当前页数，如果 pageNum<=0，则查询所有	pageNum=2
pageSize	int	分页大小	pageSize=20

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0, "data":["地理围栏 A","地理围栏 B"]}
JSON（异常）	参照状态码列表

2.3.10.8 查询与某个地理围栏绑定的所有实体

接口名称：gpsp.relation.querybindEntitiesByFence

接口描述：查询与某个地理围栏绑定的某个业务类别下的所有实体

接口参数：

参数名称	参数类型	参数说明	示例
fenceName	string	地理围栏名称，必选	fenceName=地理围栏 A
bizType	string	业务类型，目前支持"general"、"jt808"，默认为 general，可选	bizType=general
pageNum	int	当前页数，如果 pageNum<=0，则查询所有，必选	pageNum=2
pageSize	int	分页大小	pageSize=20

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0, "data":["实体 A","实体 B"]}

JSON（异常）	参照状态码列表

2.3.11 统计报表

2.3.11.1 实体数量统计

2.3.11.1.1 查询某个业务类别下所有实体的增加、活跃和总量统计

接口名称：gpsp.report.entity.queryCountByApp

接口描述：支持按日、月统计周期，查询应用的某个业务类别下所有实体的数量统计信息，包括如下几类：

（1） ADD 增加的实体数 ,例如 8 月 18 增加的实体数为 8 月 18 日实体总数减去 8 月 17 日实体总数。

（2） ACTIVE：活跃实体数，统计周期内上传过一个位置点即为活跃实体。

（3） TOTAL：实体总数

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选。	bizType=general
reportType	string	报表类型，当前支持 ADD、ACTIVE 和 TOTAL	reportType=ADD
periodType	string	统计周期，当前支持日（DAY）、月（MONTH）两种统计周期。	periodType=DAY
startTime	string	开始时间，针对不同的统计周期，参数详情如下： （1） periodType 为 DAY 时，start 的格式为 YYYYMMDD，示例格式为	1.peroidType 为 DAY 时 startTime=20160818 2.peroidType 为 MONTH 时 startTime=201608

		20150818 （2） peroidType 为 MONTH 时，start 的格式为 YYYYMM，示例格式为 201508	
endTime	string	结束时间，参数详情同上	1.peroidType 为 DAY 时 endTime=20160820 2.peroidType 为 MONTH 时 endTime=201609

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0, "data":{"reportType":"ADD","peroidType": "DAY" ,"report":[{"time":"20160818", "count":10},{"time":"20150819", "count":15},{"time":"20150820", "count":20}]}}
JSON（异常）	参照状态码列表

2.3.11.1.2 查询某个业务类别下所有实体的时长统计

接口名称：gpsp.report.entity.queryRuntimeByApp

接口描述：支持按日、月统计周期，查询应用中某个业务类别下所有实体的运行时长统计，时长单位为分钟。

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general

periodType	string	统计周期，当前支持日（DAY）、月（MONTH）两种统计周期。	periodType=DAY
startTime	string	开始时间，针对不同的统计周期，参数详情如下： （1）periodType 为 DAY 时，start 的格式为 YYYYMMDD，示例格式为 20150818 （2）peroidType 为 MONTH 时，start 的格式为 YYYYMM，示例格式为 201508	1.peroidType 为 DAY 时 startTime=20160818 2.peroidType 为 MONTH 时 startTime=201608
endTime	string	结束时间，参数详情同上	1.peroidType 为 DAY 时 endTime=20160820 2.peroidType 为 MONTH 时 endTime=201609

返回结果：

返回类型	**示例**
JSON（正常）	正确结果 {"code":0, "data":{"reportType":"RUNTIME","peroidType":"DAY" ,"report":[{"time":"20160818", "runtime":100},{"time":"20150819", "runtime":150},{"time":"20150820", "runtime":200}]}}
JSON（异常）	参照状态码列表

2.3.11.1.3 查询某个业务类别下所有实体的里程统计

接口名称：gpsp.report.entity.queryMileageByApp

接口描述：支持按日、月统计周期，查询应用中某个业务类别下所有实体的里程统计，里程单位为千米。

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
periodType	string	统计周期，当前支持日（DAY）、月（MONTH）两种统计周期。	periodType=DAY
startTime	string	开始时间，针对不同的统计周期，参数详情如下： （1）periodType 为 DAY 时，start 的格式为 YYYYMMDD，示例格式为 20150818 （2）peroidType 为 MONTH 时，start 的格式为 YYYYMM，示例格式为 201508	1.peroidType 为 DAY 时 startTime=20160818 2.peroidType 为 MONTH 时 startTime=201608
endTime	string	结束时间，参数详情同上	1.peroidType 为 DAY 时 endTime=20160820 2.peroidType 为 MONTH 时 endTime=201609

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0, "data":{"reportType":"MILEAGE","peroidType":"DAY","report":[{"time":"20160818", "mileage":100},{"time":"20150819", "count":120},{"time":"20150820", "mileage ":200}]}}
JSON（异常）	参照状态码列表

2.3.11.1.4 查询单个实体的时长统计

接口名称：gpsp.report.entity.queryRuntimeByEntity

接口描述：支持按日、月统计周期，查询单个实体的时长统计，时长单位为分钟。

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
entityName	string	实体名称，同一应用下全局唯一。	entityName=实体 A
periodType	string	统计周期，当前支持日（DAY）、月（MONTH）两种统计周期。	periodType=DAY
startTime	string	开始时间，针对不同的统计周期，参数详情如下： （1） periodType 为 DAY 时，start 的格式为 YYYYMMDD，示例格式为 20150818 （2） peroidType 为 MONTH 时，start 的格式为 YYYYMM，示例格式为 201508	1.peroidType 为 DAY 时 startTime=20160818 2.peroidType 为 MONTH 时 startTime=201608
endTime	string	结束时间，参数详情同上	1.peroidType 为 DAY 时 endTime=20160820 2.peroidType 为 MONTH 时 endTime=201609

返回结果：

返回类型	示例

JSON（正常）	正确结果 {"code":0, "data":{"reportType":"MILEAGE",peroidType": "DAY" ,"report":[{"time":"20160818", "runtime":1000},{"time":"20150819", "runtime":1200},{"time":"20150820", "runtime":1100}]}}
JSON（异常）	参照状态码列表

2.3.11.1.5 查询单个实体的里程统计

接口名称：gpsp.report.entity.queryMileageByEntity

接口描述：支持按日、月统计周期，查询单个实体的里程统计，里程单位为千米。

参数名称	参数类型	参数说明	示例
entityName	string	实体名称，同一应用下全局唯一。	entityName=实体 A
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
periodType	string	统计周期，当前支持日（DAY）、月（MONTH）两类统计维度。	periodType=DAY
startTime	string	开始时间，针对不同的统计周期，参数详情如下： （1）periodType 为 DAY 时，start 的格式为 YYYYMMDD，示例格式为 20150818 （2）peroidType 为 MONTH 时，start 的格式为 YYYYMM，示例格式为 201508	1.peroidType 为 DAY 时 startTime=20160818 2.peroidType 为 MONTH 时 startTime=201608
endTime	string	结束时间，参数详情同上	1.peroidType 为 DAY 时 endTime=20160820 2.peroidType 为 MONTH 时

			endTime=201609

返回结果：

返回类型	示例
JSON（正常）	正确结果 {"code":0, "data":{"reportType":"MILEAGE",peroidType":"DAY","startTime":"20150818",endTime:"20160818","report":[{"time":"20160818","mileage":100},{"time":"20150819", "mileage":120},{"time":"20150820","mileage":200}]}}
JSON（异常）	参照状态码列表

2.3.11.2 实体事件统计

2.3.11.2.1 查询单个实体的事件统计

接口名称：gpsp.report.entity.queryEventByEntity

接口描述： 查询实体的事件统计，目前支持如下事件列表：

（1） FENCE_MOVEIN：进地理围栏，需要事先设置进地理围栏规则，并使规则生效；同时将实体与地理围栏绑定。

（2） FENCE_MOVEOUT：出地理围栏，需要事先设置出地理围栏规则，并使规则生效；同时将实体与地理围栏绑定。

（3） FENCE_ALL：进出地理围栏，需要事先设置进/出地理围栏规则，并使规则生效；同时将实体与地理围栏绑定。

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认	bizType=general

		为 general，可选	
entityName	string	实体名称，必选	entityName=实体 A
reportType	string	事件类型，目前支持地理围栏类事件，必选	reportType =FENCE_MOVEIN
startTime	long	开始时间，Unix 时间戳，单位为毫秒，必选	startTime=1464147173225
endTime	long	结束时间，Unix 时间戳，单位为毫秒，必选	endTime=1464147183225
pageNum	int	当前页数，如果 pageNum<=0，则查询所有的规则	pageNum=2
pageSize	int	分页大小	pageSize=20

返回结果：

返回类型	**示例**
JSON（正常）	reportType 为 FENCE_MOVEIN 时，返回结果如下： 正确结果 {"code":0, "data":{"entityName": "实体 A"," reportType ":"FENCE_MOVEIN" ,"report":[{"lon":120.10, "lat":31.20, "coordinate":"WGS84","gnssTime":1464147183225},{"lon":121.10, "lat":32.42, "coordinate":"WGS84","gnssTime":1464147203225}]}}
JSON（异常）	参照状态码列表

2.3.11.3 位置统计报表

2.3.11.3.1 查询某个业务类别下所有实体实时位置区域分布

接口名称：gpsp.report.entity.regionDistribution

接口描述：查询某个类别下所有实体的实时位置区域分布。实时实体目前默认为 180s 内上传过位置点的实体，统计周期为 30s 一次。

接口参数：

参数名称	参数类型	参数说明	示例
bizType	string	业务类型，目前支持 general、jt808，默认为 general，可选	bizType=general
reportType	string	报表类型，目前支持 GLOBAL、STATE、PROVINCE、CITY，分别全球、国家、省、城市。当前仅支持 STATE、PROVICE。	reportType=STATE
regionName	string	区域名称 ,针对不同的区域类型 ,详情如下： （1） STATE：当前仅支持"CHN"，查询在中国的各省份的分布统计。 （2） PROVINCE：省份，查询在该省的各地区的分布统计，暂未开放。	regionName=CHN

返回结果：

返回类型	示例
JSON（正常）	reportType 为 STATE 时，返回结果如下： 正确结果 {"code":0, "data":{"reportType":"STATE" ,"report":[{"code":"370000" ," province":"山东省",:"count":100},{"code":"310000","province":"上海市","count":40}]}}
JSON（异常）	参照状态码列表

第三章 JT808 协议

3.1 概述

3.1.1 协议简介

JT808 协议是交通运输部的部标协议 ,该协议规定了卫星定位终端与平台之间的通讯协议和数据格式，包括协议基础、通讯连接、消息处理、协议分类与说明及数据格式。

千寻云踪支持 JT808 协议，可将满足 JT808 的卫星定位终端快速接入千寻云踪，大大降低开发者使用千寻云踪门槛。

您可以调用 2.3 章节中的相关接口进行管理工作，业务类型 bizType 请使用“jt808”。

3.1.2 协议文档

千寻云踪支持的 JT808 协议为交通运输部发布的 2013 年 1 月的版本（最新版），全名为《道路运输车辆卫星定位系统北斗兼容车载终端通讯协议技术规范》。

3.1.3 地址和端口号

JT808 协议地址：jt808.lbs.qxwz.com，对应 ip 地位为 60.205.129.90

JT808 协议端口：8808

3.1.4 已支持的协议功能

名称	说明
终端注册	终端向平台发送消息告知其安装在某一辆车上
终端注销	终端向平台发送消息告知从所安装的车辆拆下
终端鉴权	终端连接上平台时向平台发送消息以便使平台验证自己的身份
位置信息汇报	终端根据参数设定周期性或事件性地上传位置信息到平台

3.2 开发指南

3.2.1 新手上路

新手上路为您提供最基本的使用流程，您只需要完成下面三步，便可快速搭建一个最简单的业务系统。请先参照 2.1 章节完成账号注册和应用创建工作。

3.2.1.1 创建实体

首先您需要创建一个实体。实体创建时必须传入 SIM 卡号（qx_simCard）、终端 ID（qx_terminalId））和车牌号（qx_plateNumber），并且确保 SIM 卡号与 JT808 的终端的实际配置保持一致，否则将注册不成功。实体名称可以为任意值，只需保证实体名称不重复即可。

相关接口：

（1） 创建实体：gpsp.jt808.entity.createEntity

3.2.1.2 配置终端

创建好实体后，需要在终端上配置对应的参数，快速将终端接入千寻云踪的 JT808 网关。您既可以通过厂家提供软件通过串口进行配置，也可以通过短信进行配置。此处以对博实结的终端（BSJ_MINI）进行短信配置加以说明。配置短信示例如下：

域名：<SPBSJ*P:BSJGPS*G:jt808.lbs.qxwz.com,8808*A:CMNET*N:177****6822*C:0030*O:0120>

IP：<SPBSJ*P:BSJGPS*T:60.205.129.90,8808*A:CMNET*N:177****6822*C:0030*O:0120>

其中：

SPBSJ 短信头

*P:BSJGPS 密码

*G:jt808.lbs.qxwz.com,8808 jt808 的域名和端口号

*T:60.205.129.90,8808 jt808 的域名和端口号

*A:CMNET APN，即接入点名称

N:177****6822 终端上的 SIM 卡号

*C:0030 ACC 开回传，单位秒，代表 30s 回传一个点

*O:0120 ACC 关回传，单位秒，代表 120s 回传一个点

3.2.1.3 查询实时位置和历史轨迹

JT808 终端注册成功后，便会自动上报位置到云端进行存储和处理。您调用千寻云踪的接口便可查看终端的实时位置和历史轨迹。

相关接口：

（1）上传单个轨迹点：gpsp.point.locate

（2）批量上传轨迹点：gpsp.point.queryPoints

请注意，上述两个接口为位置管理中的的通用接口，其中 bizType 参数请填写为 jt808。

3.3 接口列表

在千寻云踪通用接口的基础上，针对 JT808 提供如下接口，以满足 JT808 的个性化业务。您可以基于这些接口，丰富扩展您的应用。

3.3.1 创建实体

接口名称：gpsp.jt808.entity.createEntity

接口描述：首先您需要创建一个实体。实体创建时必须传入 SIM 卡号（qx_simCard）、终端 ID（qx_terminalId））和车牌号（qx_plateNumber），并且确保 SIM 卡号与 JT808 的终端的实际配置保持一致，否则将注册不成功。实体名称可以为任意值，只需保证实体名称不重复即可。重点说明如下：

（1） SIM 卡号（qx_simCard）对应 jt808 协议消息头内容中的终端手机号。值得注意的是物联网卡出现后，号码位数已超出 12 位，协议已无法适应。可采用如下变通方法，自定义一个小于 12 位的数作为 SIM 卡号进行设置，自行维护该数值与 SIM 卡号的对应关系，终端上的 SIM 卡号同步地设置为该数字。

4.4.3 消息头

消息头内容详见 表 2：

表 2 消息头内容

起始字节	字段	数据类型	描述及要求
0	消息 ID	WORD	
2	消息体属性	WORD	消息体属性格式结构图见图 2
4	终端手机号	BCD[6]	根据安装后终端自身的手机号转换。手机号不足 12 位，则在前补充数字，大陆手机号补充数字 0，港澳台则根据其区号进行位数补充。
10	消息流水号	WORD	按发送顺序从 0 开始循环累加
12	消息包封装项		如果消息体属性中相关标识位确定消息分包处理，则该项有内容，否则无该项

消息体属性格式结构图如图 2 所示：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		分包	数据加密方式			消息体长度									

（2） 终端 ID（qx_terminalId））和车牌号（qx_plateNumber）对应 jt808 协议中的终端 ID 和车辆标识

8.5 终端注册

消息ID：0x0100。

终端注册消息体数据格式见表7。

表 7 终端注册消息体数据格式

起始字节	字段	数据类型	描述及要求
0	省域 ID	WORD	标示终端安装车辆所在的省域，0 保留，由平台取默认值。省域 ID 采用 GB/T 2260 中规定的行政区划代码六位中前两位。
2	市县域 ID	WORD	标示终端安装车辆所在的市域和县域，0 保留，由平台取默认值。市县域 ID 采用 GB/T 2260 中规定的行政区划代码六位中后四位。
4	制造商 ID	BYTE[5]	5 个字节，终端制造商编码
9	终端型号	BYTE[20]	20 个字节，此终端型号由制造商自行定义，位数不足时，后补“0X00”。
29	终端 ID	BYTE[7]	7 个字节，由大写字母和数字组成，此终端 ID 由制造商自行定义，位数不足时，后补“0X00”。
36	车牌颜色	BYTE	车牌颜色，按照 JT/T415-2006 的 5.4.12。 未上牌时，取值为 0。
37	车辆标识	STRING	车牌颜色为 0 时，表示车辆 VIN； 否则，表示公安交通管理部门颁发的机动车号牌。

（3） 终端设置千寻云踪的 808 协议的地址和端口后，首先向平台发送注册请求，平台仅对终端手机

号进行检验，默认将不对终端 ID 和车牌号检验。如果终端上报的终端手机号信息与平台登记的 qx_simCard 信息不相同，平台将直接拒绝。如果您需要开启对终端 ID 和车牌号的强制校验，请联系我们（service@wz-inc.com）。

接口参数：

参数名称	参数类型	参数说明	示例
entity	JSON	实体对象，其中 1.entityName 为实体名称 ,同一个应用下全局唯一，必选。 2.entityType 为实体类型，例如“电动车”、“可穿戴设备”、“货车”等，一旦设定不支持更改，必选。 3.attributes 为自定义扩展属性，其中自定义属性不能为 nm、tp、appKey、bizType,不允许以 qx_开头。其中 qx_simCard 、 qx_terminalId 、 qx_plateNumber 为预设字段 ,分别代表 SIM 卡号、终端编号、车牌号，需要与终端配置保持一致。	entity={"entityName":"沪 A808", "entityType":"小汽车","attributes":{"qx_simCard":"17717336822", "qx_terminalId":"11474734","qx_plateNumber":"沪 A808"}}

返回结果：

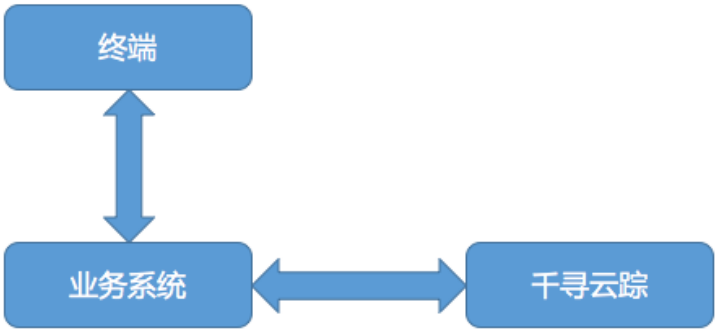
返回类型	示例
JSON（正常）	{"code":0,"data":"沪 A1234"}
JSON（异常）	参照状态码列表

第四章 常见问题

4.1 建设模式

开发者基于千寻云踪搭建业务系统，首先需要理清建设模式的问题，即终端与业务系统、终端与千寻云踪、业务系统与千寻云踪之间的关系。这里我们重点说明下两种主要的建设模式，分别适用于已有业务系统和需要新建业务系统的开发者。

4.1.1 终端-业务系统-千寻云踪



4.1.1.1 适用的场景

（1）开发者已有业务系统，不希望对业务系统进行大的改造

（2）终端协议千寻云踪未支持

4.1.1.2 特点

（1）终端与平台的交互由业务系统完成，即由开发者全权负责

（2）千寻云踪只与业务系统交互，完全不与终端直接交互

（3）千寻云踪聚焦于为开发者提供位置服务能力，业务功能的搭建由开发者独立完成。

4.1.1.3 优点

（1）原有业务系统无需改变

（2）只需同步少量信息到千寻云踪，便可使用千寻云踪的海量服务

（3）既可选择基于千寻云踪的接口搭建新的功能模块，也可用千寻的高性能接口替换业务系统的旧接口

（4）既可选择利用千寻云踪的海量存储和计算能力，也可选择仅仅利用千寻云踪的计算能力

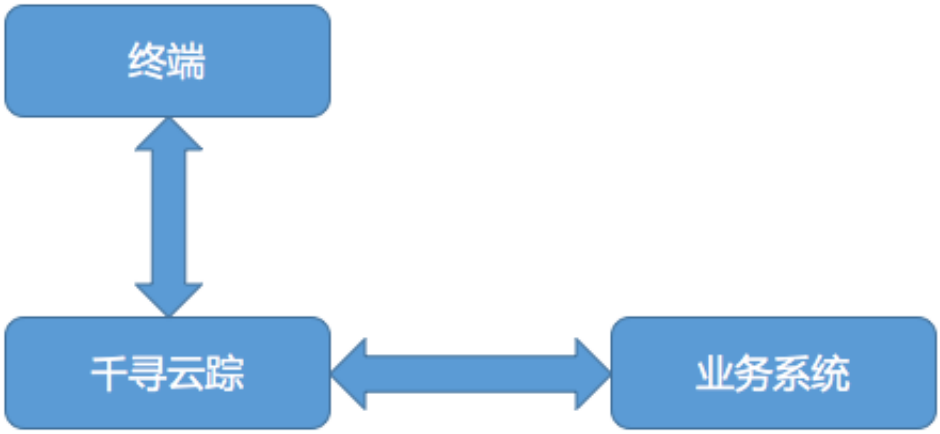
（5）千寻云踪自身提供配套工具，开发者在未来上线的商城选择丰富的 SAAS 类产品。

（6）释放平台数据增值服务能力，通过平台化运营帮助合作伙伴提升最终的用户体验。

4.1.1.4 缺点

（1）终端与平台交互需要开发者独立实现

4.1.2 终端-千寻云踪-业务系统



4.1.2.1 适用的场景

（1）终端协议千寻云踪已支持，或者是智能终端集成千寻云踪的 API。

（2）计划新建业务系统

（3）对原有业务系统进行大的改造

4.1.2.2 特点

（1）终端与平台的交互由千寻云踪完成

（2）业务系统与千寻云踪直接交互，并可通过千寻云踪的接口与终端进行交互。

（3）千寻云踪聚焦于为开发者提供位置服务能力，业务功能的搭建由开发者独立完成。

4.1.2.3 优点

（1）终端与平台交互由千寻云踪实现，开发者只需要调动千寻的接口来控制终端。

（2）终端进行简单的配置或开发便可将数据上传至千寻云踪，进而使用千寻云踪的海量服务

（3）千寻云踪聚焦于位置服务能力，开发者只需重点关注业务系统的建设和逻辑组织。

（4）千寻云踪自身提供配套工具，开发者可在未来上线的商城中选择丰富的 SAAS 类产品。

（5）释放平台数据增值服务能力，通过平台化运营帮助合作伙伴提升最终用户体验。

4.1.2.4 缺点

（1）原有业务系统需要进行较大的改造

4.2 数据同步方案

对于采用终端-业务系统-千寻云踪建设模式的客户，将面临着两个平台数据同步的问题。本文提出两种数据同步方案，供客户根据实际情况自主选择，这里数据同步是指实体的同步，核心是实体名称的同步。

4.2.1 强同步方案

所谓强同步方案，是指将两个平台的实体保持严格同步，增、删、改均需要保持信息同步。具体为

（1）开发者首先需要将自身平台中已经存在的实体在千寻云踪中进行创建。最简单的同步仅保持实体名称相同即可。相关接口：gpsp.entity.createEntity。

（2）自身平台新增实体时，需要同步地在千寻云踪中新增一个实体。相关接口：gpsp.entity.createEntity。

（3）自身平台删除实体时，需要同步地在千寻云踪中删除一个实体。相关接口：gpsp.entity.deleteEntity。

（4）自身平台更新实体时，需要同步地在千寻云踪中更新一个实体；如果您仅仅在千寻云踪中存储实体名称，那么可不维护更新的同步。 相关接口：gpsp.entity.updateEntity。

4.2.2 弱同步方案

所谓弱同步方案，是指两个平台的实体不保持严格同步，仅维护活跃的实体保持同步，千寻云踪中不存储不活跃的实体，千寻云踪不同步删除操作。

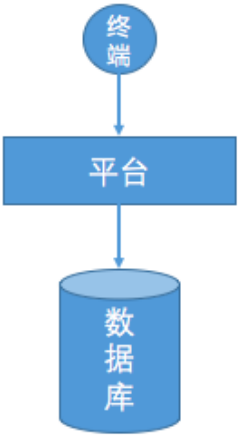
（1）开发者调动千寻云踪时，无需关注实体是否存在，直接调用上传单个轨迹点/批量上传轨迹点接口，如果成功，则处理完毕。相关接口：gpsp.point.addTrackPoint、gpsp.point.addTrackPoints。

（2）如果返回没有该实体的错误，则创建该实体后。相关接口：gpsp.entity.createEntity。

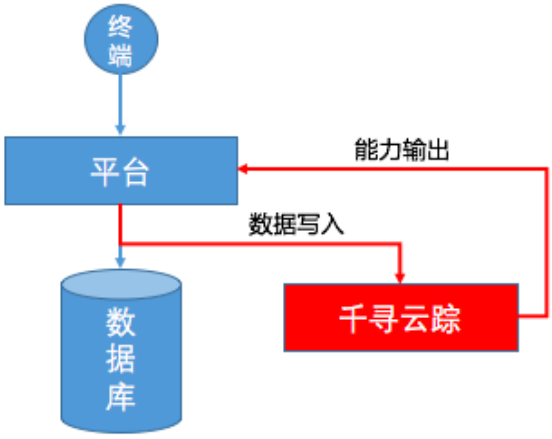
强、弱同步方案各有优缺点，强同步方案优点在于可以和开发者自身平台保持严格同步，缺点在于同步工作量大。弱同步方案的优点在工作量很小，但与自身平台无法严格同步。

对于位置点上传部分，在原有平台的中只需要在位置数据写入数据库的接口处，另开一份分支写入千寻云踪即可，基本上对现有系统没有影响。客户可根据自身的需要，选择千寻云踪的功能拓展自身平台的能力或替换已有的位置相关的接口。业务系统与千寻系统集成方案示例如下：

原有业务系统



业务系统与千寻云踪的集成



第五章 更新日志

5.1 千寻云踪 Web API V1.0 发布上线

5.1.1 发布时间

2016 年 6 月 21 日

5.1.2 版本描述

实现位置数据采集、存储和开放的主流程，支持实体管理、位置管理和配置管理

5.1.3 详细描述

（1） 实体管理：支持实体的创建、删除、更新和查询；实体支持自定义字段。

接口名称	接口说明
gpsp.entity.createEntity	创建实体
gpsp.entity.deleteEntity	删除实体
gpsp.entity.updateEntity	更新实体
gpsp.entity.queryEntity	查询单个实体
gpsp.entity.queryEntities	查询多个实体
gpsp.entity.queryEntitiesByApp	查询所有实体

（2） 位置管理：支持上传、批量上传轨迹点、查询历史轨迹；轨迹点支持自定义字段。

接口名称	接口说明
gpsp.point.addPoint	上传单个轨迹点
gpsp.point.addPoints	批量
gpsp.point.locate	查询实时位置
gpsp.point.queryPoints	查询历史轨迹

（3） 配置管理：支持存储配置、抽稀配置，支持查询配置详情。

接口名称	接口说明
gpsp.config.setSimplifyConfig	抽稀配置
gpsp.config.setStoreConfig	存储配置
gpsp.config.getConfig	获取配置表

5.2 千寻云踪 Web API V1.1 发布上线

5.2.1 发布时间

2016 年 8 月 5 日

5.2.2 版本描述

以地理围栏为核心，配套提供规则引擎、消息通道、关系管理、统计报表功能。

5.2.3 详情描述

（1） 地理围栏管理：支持地理围栏的创建、删除、更新和查询，目前仅支持多边形地理围栏。

接口名称	接口说明
gpsp.fence.createFence	创建地理围栏
gpsp.fence.deleteFence	删除地理围栏
gpsp.fence.updateFence	更新地理围栏
gpsp.fence.queryFence	查询单个地理围栏
gpsp.fence.queryFencesByApp	查询所有地理围栏

（2） 规则引擎管理：支持规则引擎的创建、删除、更新和查询，目前仅支持地理围栏的规则。

接口名称	接口说明
gpsp.rule.createRule	创建规则

gpsp.rule.deleteRule	删除规则
gpsp.rule.updateRule	更新规则
gpsp.rule.queryRule	查询单个规则
gpsp.rule.queryRulesByApp	查询所有规则

（3） 消息通道管理：支持消息通道的创建、删除、更新和查询，目前仅支持阿里云 MNS。

接口名称	**接口说明**
gpsp.channel.createChannel	创建消息通道
gpsp. channel.deleteChannel	删除消息通道
gpsp. channel.updateChannel	更新消息通道
gpsp. channel.queryChannel	查询单个消息通道
gpsp. channel.queryChannelsByApp	查询所有消息通道

（4） 关系管理：支持创建和解除全局绑定关系，目前仅支持规则、地理围栏的全局绑定；支持实体与地理围栏一对一的绑定和解绑；支持绑定关系的查询。

接口名称	**接口说明**
gpsp.relation.bindApp	创建全局绑定关系
gpsp.relation.unbindApp	解除全局绑定关系
gpsp.relation.queryAppBind	查询单个全局绑定关系
gpsp.relation.queryAppBinds	查询所有的全局绑定关系
gpsp.relation.bindEntityAndFence	创建单个实体与单个地理围栏的绑定关系
gpsp.relation.unbindEntityAndFence	解除单个实体与单个地理围栏的绑定关系
gpsp.relation.querybindFencesByEntity	查询与某个实体绑定的所有地理围栏

gpsp.relation.querybindEntitiesByFence	查询与某个地理围栏绑定的所有实体

（5） 统计报表：仅提供地理围栏类别的事件统计报表

接口名称	接口说明
gpsp.report.entity.queryEventByEntity	查询某个实体的事件统计

（6） 配置管理：支持全局设置是否开启地理围栏功能

接口名称	接口说明
gpsp.config.setFenceConfig	地理围栏配置

5.3 千寻云踪 Web API V1.2 发布上线

5.3.1 发布时间

2016 年 8 月 24 日

5.3.2 版本描述

以统计报表为核心，提供实体的数量统计报表、实体的事件统计报表；配置管理中可查看当前应用下的配额详情。

5.3.3 详情描述

（1） 实体数量统计：支持查询实体的新增数量统计、活跃数量统计、实体总量统计；支持查询所有实体和单个实体的里程和时长统计

接口名称	接口说明
gpsp.report.entity.queryCountByApp	查询某个业务类别下所有实体的新增、活跃和总量统计
gpsp.report.entity.queryRuntimeByApp	查询某个业务类别下所有实体的时长统计
gpsp.report.entity.queryMileageByApp	查询某个业务类别下所有实体的里程统计
gpsp.report.entity.queryRuntimeByEntity	查询单个实体的时长统计

gpsp.report.entity.queryMileageByEntity	查询单个实体的里程统计
gpsp.report.entity.queryEventByEntity	查询单个实体的事件统计
gpsp.report.entity.regionDistribution	查询某个业务类别下所有实体实时位置区域分布

（2）实体事件统计：支持查询单个实体的事件统计报表

接口名称	**接口说明**
gpsp.report.entity.queryEventByEntity	查询单个实体的事件统计

（3）配置管理：获取配置表中支持查询当前应用下的配额详情

接口名称	**接口说明**
gpsp.config.getConfig	获取配置表

6.4 千寻云踪 Web API V1.3 发布上线

5.4.1 发布时间

2016 年 9 月 7 日

5.4.2 版本描述

以 JT808 协议为核心，支持将终端的位置信息上传到千寻云踪，打通终端与平台的直接交互

5.4.3 详情描述

（1）支持的 JT808 协议功能如下：

名称	**说明**
终端注册	终端向平台发送消息告知其安装在某一辆车上
终端注销	终端向平台发送消息告知从所安装的车辆拆下
终端鉴权	终端连接上平台时向平台发送消息以便使平台验证自己的身份
位置上报	终端根据参数设定周期性或事件性地上传位置信息到平台

（2） 新增创建 jt808 业务类别实体接口

接口名称	接口说明
gpsp.jt808.entity.createEntity	创建 jt808 业务类别的实体接口

（3）对外提供 jt808 网关的域名和端口：jt808.lbs.qxwz.com:8808

（4）开放 bizType 参数，标志位业务类型，支持 general、jt808 两个类型，默认为 general。bizType 开放涉及的接口有：

接口名称	接口说明
gpsp.entity.deleteEntity	删除实体
gpsp.entity.updateEntity	更新实体
gpsp.entity.queryEntity	查询单个实体
gpsp.entity.queryEntities	查询多个实体
gpsp.entity.queryEntitiesByApp	查询所有实体
gpsp.point.locate	查询实时位置
gpsp.point.queryPoints	查询历史轨迹
gpsp.config.setSimplifyConfig	抽稀配置
gpsp.config.setStoreConfig	存储配置
gpsp.config.setFenceConfig	地理围栏配置
gpsp.config.getConfig	获取配置表
gpsp.relation.bindEntityAndFence	创建单个实体与单个地理围栏的绑定关系
gpsp.relation.unbindEntityAndFence	解除单个实体与单个地理围栏的绑定关系
gpsp.relation.querybindFencesByEntity	查询与某个实体绑定的所有地理围栏
gpsp.relation.querybindEntitiesByFence	查询与某个地理围栏绑定的某个业务类别下的所有实体

gpsp.report.entity.queryCountByApp	查询某个业务类别下所有实体的新增、活跃和总量统计
gpsp.report.entity.queryRuntimeByApp	查询某个业务类别下所有实体的时长统计
gpsp.report.entity.queryMileageByApp	查询某个业务类别下所有实体的里程统计
gpsp.report.entity.queryRuntimeByEntity	查询单个实体的时长统计
gpsp.report.entity.queryMileageByEntity	查询单个实体的里程统计
gpsp.report.entity.queryEventByEntity	查询单个实体的事件统计
gpsp.report.entity.regionDistribution	查询某个业务类别下所有实体实时位置区域分布

5.5 千寻云踪 Web API V1.4 发布上线

5.5.1 发布时间

2016 年 10 月 18 日

5.5.2 版本描述

以事件管理和行程管理为核心，扩展位置服务的外延，支持用户自主管理自定义事件和行程。

5.5.3 详情描述

（1）新增事件管理

接口名称	接口说明
gpsp.event.createEvent	创建事件
gpsp.event.deleteEvent	删除事件
gpsp.event.updateEvent	更新事件
gpsp.event.queryEvent	查询事件
gpsp.event.queryEvents	查询事件列表

（2）新增行程管理

接口名称	接口说明

gpsp.trip.createTrip	创建行程
gpsp.trip.deleteTrip	删除行程
gpsp.trip.updateTrip	更新行程
gpsp.trip.queryTrip	查询行程
gpsp.trip.queryTrips	查询行程列表

5.6 千寻云踪 Web API V1.4.1 发布上线

5.6.1 发布时间

2016 年 11 月 10 日

5.6.2 版本描述

优化位置上传、轨迹查询、统计报表功能，新增按行程名称进行行程管理的功能。

5.6.3 详情描述

（1）新增 gpsp.point.addTrackPoint，推荐使用该接口替换 gpsp.point.addPoint

（2）新增 gpsp.point.addTrackPoints，推荐使用该接口替换 gpsp.point.addPoints

（3）接口 gpsp.point.queryPoints，返回值中去掉 entityName 信息，用来降低流量。

（4）新增按行程名称进行行程管理的功能

新增接口列表如下：

接口名称	**接口说明**
gpsp.point.addTrackPoint	上传单个轨迹点，推荐使用该接口替换 gpsp.point.addPoint
gpsp.point.addTrackPoints	批量上传轨迹点，推荐使用该接口替换 gpsp.point.addPoints
gpsp.trip.createUniqueTrip	根据名称创建行程
gpsp.trip.deleteTripByName	根据名称删除行程
gpsp.trip.updateTripByName	根据名称更新行程

gpsp.trip.queryTripByName	根据名称查询单个行程

5.7 千寻云踪 Web API V1.4.2 发布上线

5.7.1 发布时间

2016 年 12 月 14 日

5.7.2 版本描述

优化行程管理和事件管理，将 eventId 和 tripId 由 string 调整为 long 类型。

5.7.3 详情描述

将 eventId 和 tripId 由 string 调整为 long 类型，涉及如下接口

接口名称	接口说明
gpsp.event.createEvent	创建事件
gpsp.event.deleteEvent	删除事件，
gpsp.event.updateEvent	更新事件
gpsp.event.queryEvent	查询事件
gpsp.trip.createTrip	创建行程
gpsp.trip.deleteTrip	删除行程
gpsp.trip.updateTrip	更新行程
gpsp.trip.queryTrip	查询行程