# “找导游”服务端接口文档

## 1 概述

访问协议：Http

访问方式：RestApi

接口返回格式：Json

接口地址：<http://118.186.218.43/api>

APP\_ID=1001;

APP\_SECRET\_KEY= 9f738a3934abf88b1dca8e8092043fbd;

## 2 接口说明

2.1 /user/getVerifyCode

描述：获取短信验证码

基本参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 说明 |
| mobile | string | mobile=139\*\*\*\*\*\*\*\* |

返回值：

{"result":0} 0表示成功，非0表示失败。

2.2/user/register

描述：注册

基本参数：

|  |  |  |
| --- | --- | --- |
| mobile | string | 手机号 |
| password | string | 登陆密码 |
| name | string | 用户名 |
| verifyCode | string | 发送到手机的验证码 |
| gender | int | 性别：0未设置 1男 2女 |

返回值：

成功后将返回userId,ticket和userSecretKey等字段。

2.3 /user/login

描述：登陆

基本参数：String mobile;String password

返回值：同register。成功后返回userId,ticket和userSecretKey。

2.4 /user/ applyForGuide

描述：用户申请成为导游

基本参数：

|  |  |  |
| --- | --- | --- |
| goodAtScenic | string | 景点。多个用空格分开。 |
| birthday | long | 出生日期。14位数字。 |
| beGuideYear | int | 获得导游证年份。4位数。 |
| guideCardUrl | string | 导游证图片地址。调用上传资源接口后可得到。 |
| guideCardId | string | 导游证号 |
| location | string | 地理位置信息 |
| city | int | 所在城市代号。由客户端定义。 |

返回值：

{"result":0} 0表示成功，非0表示失败。

2.5 /user/getUserInfo

接口描述：获取用户信息。如果是游客，返回游客信息，如果是导游，返回导游信息。

基本参数：

|  |  |  |
| --- | --- | --- |
| userId | int | 系统内用户id |

返回字段：

|  |  |
| --- | --- |
| 游客信息： | 导游信息（包括游客信息）： |
| int userId | string goodAtScenic |
| string userName | long birthday |
| string mobile | int beGuideYear |
| int gender | string guideCardUrl |
| int userType 0表示游客，1表示导游 | string guideCardId |
| string headUrl | string location |
|  | int city |

2.6 /user/searchGuide

接口描述：根据条件搜索导游

基本参数：

|  |  |  |
| --- | --- | --- |
| city | int | 城市代号 |
| gender | int | 性别 |
| scenic | string | 擅长景点 |
| start | int | start和rows组合用来表示分页的情况。客户端可通过这俩参数实现翻页功能。 |
| rows | int |

返回字段：

int数组，每个元素为userId;

2.7 /user/getNearByGuide

接口描述：获取附近的导游

基本参数：

|  |  |  |
| --- | --- | --- |
| location | string | 经纬度“38.6518141832995,104.07643139362494” |
| dist | double | 半径距离，单位km。 |
| start | int | 翻页参数 |
| rows | int |

返回字段：

int数组，每个元素为userId;

2.8 /resource/uploadResource

接口描述：上传资源

基本参数：

|  |  |  |
| --- | --- | --- |
| suffix | string | 上传资源后缀。比如jpg |

文件流通过muliti-part方式上传。

返回值：

{"result":url} url为上传成功的资源url。

2.9 /inviteEvent/invite

接口描述：单点预约导游

基本参数：

|  |  |  |
| --- | --- | --- |
| guideId | int | guide的userId |
| scenic | string | 景点 |
| startTime | long | 开始时间 |
| endTime | long | 结束时间 |

返回值：

{"result":0} 0表示成功，非0表示失败。

2.10 /inviteEvent/inviteAll

接口描述：广播预约导游。服务端根据游客所在location，按照地理位置远近顺序分发给一定数量的导游。

基本参数：

|  |  |  |
| --- | --- | --- |
| scenic | string | 景点 |
| startTime | long | 开始时间 |
| endTime | long | 结束时间 |
| location | string | 游客所在地理位置 |
| gender | int | 按性别过滤。  0未设置 1男 2女 |

返回值：

{"result":0} 0表示成功，非0表示失败。

2.11 /inviteEvent/cancle

接口描述：游客取消已发出的预约

基本参数：

|  |  |  |
| --- | --- | --- |
| eventId | long | 系统内事件Id |
| guideId | int | 游客Id。广播时传0 |

返回值：

{"result":0} 0表示成功，非0表示失败。

2.12 /inviteEvent/getHistoricalInviteEvents

接口描述：游客获取预约信息。客户端通过此接口获取历史信息。客户端可通过轮训此接口，实现消息提醒功能。

基本参数：

|  |  |  |
| --- | --- | --- |
| start | int | 翻页参数。也可通过调整参数轮训接口实现消息提醒。 |
| rows | int |

返回字段：

InviteEvent的对象数组。InviteEvent定义见2.13.

2.13 /inviteEvent/getOneInviteEvent

接口描述：获取单条预约信息的详细信息。

基本参数：

|  |  |  |
| --- | --- | --- |
| eventId | long | 事件Id |

返回字段：

|  |  |  |
| --- | --- | --- |
| eventId | long | 事件Id |
| userId | int | 游客Id |
| guideId | int | 导游Id,广播事件为0 |
| eventType | int | 0表示单点预约，1表示广播 |
| eventStatus | int | 事件状态：  0 预约中  1 已取消  2 被接受  3 被拒绝  4 已评价 |
| startTime | long | 开始时间 |
| endTime | long | 结束时间 |
| scenic | string | 景点 |
| createTime | long | 事件创建时间 |
| satisfaction | int | 满意度：  0 未评价 1满意 2 不满意 |

2.14 /inviteEvent/setSatisfaction

接口描述：评价

基本参数：

|  |  |  |
| --- | --- | --- |
| eventId | long | 事件Id |
| guideId | int | 导游Id |
| satisfaction | int | 1满意 2 不满意 |

返回值：

{"result":0} 0表示成功，非0表示失败。

2.15 /guideEvent/accept

接口描述：导游接受某次预约

基本参数：

|  |  |  |
| --- | --- | --- |
| eventId | long | 事件Id |
| userId | int | 游客Id |

返回值：

{"result":0} 0表示成功，非0表示失败。

2.16 /guideEvent/refuse

接口描述：导游拒绝某次预约

基本参数：

|  |  |  |
| --- | --- | --- |
| eventId | long | 事件Id |
| userId | int | 游客Id |

返回值：

{"result":0} 0表示成功，非0表示失败。

2.17 /guideEvent/getHistoricalGuideEvents

接口描述：导游获取历史记录。客户端可通过轮训此接口，实现导游的消息提醒。

基本参数：

|  |  |  |
| --- | --- | --- |
| start | int | 翻页参数。也可通过调整参数轮训接口实现消息提醒。 |
| rows | int |

返回字段：

GuideEvent数组。GuideEvent定义见2.18

2.18 /guideEvent/getOneGuideEvent

接口描述：获取某条导游事件详细信息

基本参数：

|  |  |  |
| --- | --- | --- |
| eventId | long | 事件Id |

返回字段：

|  |  |  |
| --- | --- | --- |
| guideId | int | 导游Id |
| eventId | long | 事件Id |
| userId | int | 游客Id |
| status | int | 事件状态：  0 事件产生等待处理  1 已接受  2 已拒绝  3 被评价  4 被取消  5 事件被其他导游接受（广播） |
| satisfaction | int | 被评价为满意或不满意 |
| createTime | long | 事件创建时间 |
| eventType | int | 事件类型，0为单点预约，1为广播预约 |

## 3 接口其他说明

3.1平台级参数

以下平台级参数要求每次请求都必须传入。  
app\_id：申请应用时分配的app\_id，调用接口时候代表应用的唯一身份。  
v：API的版本号，请设置成1.0  
call\_id：当前调用请求队列号，建议使用当前系统时间的毫秒值。  
sig：签名认证，见3.3描述。  
t：用户登录后分配的ticket。登陆后操作的接口要求每次将ticket传上来。Ticket在注册或登陆成功时返回。客户端需要保存。

3.2 post请求；  
参数格式为key=value，必须对value进行URL编码（UTF-8）；

3.2签名认证计算规则

将请求中所有参数进行排序，排序为字典序。将排序好的参数进行转换　  
去掉"&",例如：k1=v1&k2=v2&k3=v3变为 k1=v1k2=v2k3=v3；  
在上述转换后的串末尾追加上APP\_SECRET\_KEY。要求登陆的接口需要在上述后面再追加userSecretKey。用MD5算出上述串的MD5值，然后作为sig的值传入请求中。

3.3 api防重发机制  
每次客户端必须传递call\_id参数给服务器，对于同一个api接口和同样的参数，call\_id的值不能一 样。call\_id建议取客户端系统当前时间值

## 4 客户端访问api测试代码，仅供参考

**package** com.tg.util;

**import** java.io.BufferedReader;

**import** java.io.DataOutputStream;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.net.HttpURLConnection;

**import** java.net.URL;

**import** java.util.HashMap;

**import** java.util.Map;

**import** java.util.Vector;

**import** java.util.zip.GZIPInputStream;

**import** org.apache.commons.httpclient.HttpException;

**public** **class** MCPUtil {

**public** **static** String *BASE\_API\_URL* = "http://118.186.218.43/api/";

**public** **static** **final** String *APP\_SECRET* = "9f738a3934abf88b1dca8e8092043fbd";

**public** **static** **final** String *APP\_ID* = "1001";

**private** **static** Map<String, String> *params* = **new** HashMap<String, String>();

**private** **static** String *ticket* = "BUjVQQVVgV20DM109DTBQOw0\_UmFTZVIzDmcPM1VjAWA.";

**private** **static** String *userSecretKey* = "0cc23bcfe33f62326aef583e3691c262";

**public** **static** **void** main(String[] args) **throws** HttpException, IOException {

*params*.put("t", *ticket*);

*params*.put("userSecretKey", *userSecretKey*);

*params*.put("guideId", "10000007");

*params*.put("scenic", "test");

*params*.put("startTime", System.*currentTimeMillis*() + "");

*params*.put("endTime", System.*currentTimeMillis*() + "");

*putOtherParam*(**true**);

String result = *POST*("inviteEvent/invite");

System.*out*.println(result);

}

**public** **static** String POST(String url) **throws** HttpException, IOException {

HttpURLConnection connection = **null**;

BufferedReader reader = **null**;

**try** {

URL hurl = **new** URL(*BASE\_API\_URL* + url);

connection = (HttpURLConnection) hurl.openConnection();

connection.setRequestMethod("POST");

connection.setDoOutput(**true**);

connection.connect();

DataOutputStream out = **new** DataOutputStream(

connection.getOutputStream());

StringBuffer sbb = **new** StringBuffer("");

**for** (String key : *params*.keySet()) {

sbb.append(key);

sbb.append("=");

sbb.append(*params*.get(key));

sbb.append("&");

}

out.writeBytes(sbb.toString());

out.flush();

out.close();

// 读取响应

GZIPInputStream gin = **new** GZIPInputStream(

connection.getInputStream());

reader = **new** BufferedReader(**new** InputStreamReader(gin, "utf-8"));

String lines;

StringBuffer sbs = **new** StringBuffer("");

**while** ((lines = reader.readLine()) != **null**) {

lines = **new** String(lines.getBytes(), "utf-8");

sbs.append(lines);

}

reader.close();

**return** sbs.toString();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

// logger.error("appstore receipt error:",e);

}

**return** "ERROR";

}

**private** **static** **void** putOtherParam(**boolean** NeedTicket) {

**if** (NeedTicket && !*ticket*.isEmpty()) {

*params*.put("t", *ticket*);

}

*params*.put("app\_id", *APP\_ID*);

*params*.put("v", "1.0"); // **TODO**

*params*.put("call\_id", String.*valueOf*(System.*currentTimeMillis*()));

*params*.put("gz", "compression");

*params*.put("sig",

*getSig*(*params*, *APP\_SECRET*, NeedTicket ? *userSecretKey* : **null**));

}

**private** **static** String getSig(Map<String, String> params,

String appSecretKey, String userSecretKey) {

Vector<String> vecSig = **new** Vector<String>();

**for** (String key : params.keySet()) {

String value = params.get(key);

vecSig.add(key + "=" + value);

}

String[] nameValuePairs = **new** String[vecSig.size()];

vecSig.toArray(nameValuePairs);

**for** (**int** i = 0; i < nameValuePairs.length; i++) {

**for** (**int** j = nameValuePairs.length - 1; j > i; j--) {

**if** (nameValuePairs[j].compareTo(nameValuePairs[j - 1]) < 0) {

String temp = nameValuePairs[j];

nameValuePairs[j] = nameValuePairs[j - 1];

nameValuePairs[j - 1] = temp;

}

}

}

StringBuffer nameValueStringBuffer = **new** StringBuffer();

**for** (**int** i = 0; i < nameValuePairs.length; i++) {

nameValueStringBuffer.append(nameValuePairs[i]);

}

nameValueStringBuffer.append(appSecretKey);

**if** (userSecretKey != **null** && !userSecretKey.isEmpty()) {

nameValueStringBuffer.append(userSecretKey);

}

String sig = MD5Util.*md5*(nameValueStringBuffer.toString());

**return** sig;

}

}