

脸脸网服务器与手机端的开发接口API

YXY

2012 年 11 月 7 日

目录

1	初始化	5
2	登录和退出	6
2.1	老的登录方式: oauth2	6
2.2	新的登录方式: xauth	8
2.3	退出	8
3	根据经纬度获得可能的现场商家	9
4	根据经纬度获得附近的商家	11
5	获得附近的用户	12
6	根据商家ID获得商家的基本信息	12
7	根据商家ID获得当前在该商家的所有用户	14
8	进入现场签到	15
9	获得用户基本信息	16

目录	2
10 获得用户的头像	18
11 获得用户的图片列表	18
12 获得当前登录用户基本信息	19
13 设置当前登录用户基本信息	21
14 当前登录用户上传图片	22
15 当前登录用户批量调整所有图片的位置	23
16 当前登录用户删除图片	24
17 添加我关注的人	25
18 删除我的关注的人	26
19 粉丝列表	27
20 关注(好友)列表	29
21 黑名单列表	29
22 添加黑名单	30
23 删除黑名单	30
24 XMPP协议接口	31
24.1 与Openfire聊天服务器接口	31
24.2 脸脸中的聊天用户的三种类型	32
24.3 摇一摇及其消息格式	32
24.4 个人之间聊天的消息发送状态确认	32

目录	3
25 在聊天室发图	33
25.1 上传图片	34
25.2 发送消息	35
25.3 在聊天室收到消息后获取图片	35
26 个人聊天时发图	36
26.1 个人聊天时上传图片	36
26.2 发送消息	36
26.3 在个人聊天时收到消息后获取图片	37
26.4 消息状态	37
27 优惠券	37
27.1 商家推送优惠券	37
27.2 优惠券展示	38
27.3 优惠券使用	38
28 关于程序各页面之间的切换规则	38

注意事项:

1. 所有的链接，如果要获得json格式的数据，最好带上".json"的后缀。
因为同一个链接以后可能返回多种格式的数据，比如xml/html/json等。
2. 当调用出错时，会在返回的json中包含"error"信息。
3. 只有开发调试时调用http的接口，发布时全部使用https接口。

1 初始化

手机应用启动后，初次访问时调用此API。

https://60.191.119.190/init/init			
参数	类型	必填	说明
model	字符串	必须	硬件型号
os	字符串	必须	手机操作系统版本
mac	字符串	必须	无线网卡MAC地址的MD5
hash	字符串	必须	hash验证码
ver	字符串	可选	软件版本

注意：本API直接通过IP地址“60.191.119.190”访问，不通过DNS。而后续所有API调用的IP地址根据本调用的返回的IP地址确定。在目前只有单台服务器的情况下，返回的ip地址也是“60.191.119.190”，但是以后会根据离用户的距离返回就近的IP地址。

hash的计算方式为：model+os+mac+”init”进行SHA1算法后取前32位，和登录时的hash算法 [2.2] 类似。

```
返回值：
{
  "ip": "60.191.119.190",
  "xmpp": "60.191.119.190"
}
```

后续的http访问使用返回的ip地址，xmpp协议的访问使用xmpp地址。

本API的作用一个是给客户端推荐IP地址，一个是统计应用的开机情况和操作系统分布。

2 登录和退出

2.1 老的登录方式：oauth2

/oauth2/sina_callback			
参数	类型	必填	说明
code	字符串	必须	新浪返回的code

例子：

访问: /oauth2/sina_callback?code=...

该访问由新浪微博通过302转向发起。

返回值:

```
{
  "id":1,
  "password":"15c663b8ff620502",
  "logo": "",
  "name" : "name"
  "gender" : 1
  "wb_uid":"1884834632",
  "expires_in":75693,
  "token":"2.00aaZYDCMcnDPCb4dc439e06i2m_GC",
  "expires_at":1338431259
}
```

其中, id和password是该用户在脸脸网的id和密码, 在首次新浪授权时创建。该id加上"@dface.cn"就是openfire的jid。(目前还没有和jid关联上。)

后面几个字段都是新浪提供的, 其中wb_uid是该用户的新浪微博的uid, token是授权的access_token, 其它几个是授权有效期信息。

在首次登录获得用户的id和password以后, 可以缓存到本地。以后用户每次登录, 返回的id和password是不会改变的, 除非服务器端重置密码(比如出现密码泄漏等安全情况)。

logo可以得知用户是否成功上传头像。对于第一次登录的用户, 其logo为空, 此时需要引导用户上传头像。

2.2 新的登录方式：xauth

/oauth2/login			
参数	类型	必填	说明
name	字符串	必须	用户名
pass	字符串	必须	密码
mac	字符串	必须	网卡Mac地址
hash	字符串	必须	hash验证码

本接口的输出和oauth2登录接口的输出相同。

hash的计算方式为：`name+pass+mac+"dface"`进行SHA1算法后取前32位。

比如用户名为`name`，密码为`pa`，mac地址为`ss`

那么待hash的字符串为`"namepassdface"`，

其hash码为`"11f4004a73a65117071bc4a7d3dfdf07"`。

新的登录方式不需要调用新浪的Xauth API，直接调用本API即可以登录。客户端应该不需要保存AppSecret。通过预先保存的AppKey和本调用输出中包含的token应该就可以访问新浪微博的API了。

2.3 退出

/oauth2/logout			
参数	类型	必填	说明

退出调用不需要参数。当用户主动退出时，调用本接口并断开XMPP的连接。

3 根据经纬度获得可能的现场商家

/aroundme/shops			
参数	类型	必填	说明
lat	浮点数	必须	经度
lng	浮点数	必须	纬度
accuracy	整数	必须	经纬度的精确度

注意：一定要在accuracy小于200m或者调用获取GPS的API超过三次后才能调用本接口。

Gps获取位置是一个逐步精确的过程。很多时候首次获取的误差超过1000米。此时显然无法准确获得商家列表。此时建议等待0.5秒再次获取gps。要保证gps的精确度小于200m，或则定位三次以上确实不能更准确了。宁可等待的时间久一些，也要保证定位的准确性。

最多返回50条数据，也有可能没有数据（比如西部沙漠地区）。

例子：

访问: /aroundme/shops.json

返回值:

```
[
{
  "name": "新紫轩花店",
  "address": "文一路266号",
  "lng": 120.12763,
  "id": 40056,
  "phone": "",
  "lat": 30.28691,
  "lo": [30.297, 120.1288],
  "t": 1,
  "user": 0,
  "male": 0,
  "female": 0
}
```

除了返回商家的id、名称、电话、经纬度以后，增加了在该商家的用户总数“user”、男“male”、女“female”数量。

[2012-8-7] 新增lo字段。这是商家的实际经纬度，以前的lat/lng是地图上显示的经纬度，两者有几百米的误差。计算商家和自己的距离时，要采用lo来计算。

[2012-8-8] 新增t字段，表示商家的类型。

4 根据经纬度获得附近的商家

/shop/nearby			
参数	类型	必填	说明
lat	浮点数	必须	经度
lng	浮点数	必须	纬度
accuracy	整数	必须	经纬度的精确度
page	整数	可选	分页，缺省值为1
pcount	整数	可选	每页的数量，缺省为20
name	字符串	可选	商家的名称
type	整数	可选	商家类型

商家类型：

- 1. 酒吧●活动
- 2. 咖啡●茶馆
- 3. 餐饮●酒店
- 4. 休闲●娱乐
- 5. 购物●广场
- 6. 楼宇●社区

用户当前所在的现场只有一个，但是由于定位有误差，所以服务器返回最有可能的几个商家让用户选择；而附近的商家有很多，按距离由近到远排序，且支持查询、分类筛选和分页。

5 获得附近的用户

/aroundme/users			
参数	类型	必填	说明
gender	数字	必填	用户的性别，未设置0男1女2

输出格式和“根据商家ID获得商家用户”的接口一样。

6 根据商家ID获得商家的基本信息

/shop/info			
参数	类型	必填	说明
id	整数	必须	商家的ID

例子：

```
{
  "name": "浙江科技产业大厦",
  "t": 6,
  "lat": 30.280254,
  "lng": 120.121824,
  "address": "古翠路80",
  "id": 4928288,
  "staffs": [
    "502e6303421aa918ba000001"
  ],
  "notice":""
}
```

staffs是该商家的员工数组。在商家聊天室，商家和商家的员工发言时都加V并加黄色背景。只有该商家在商家聊天室才加V，其它商家不做特殊处理。

notice是该商家发布的公告。

7 根据商家ID获得当前在该商家的所有用户

/shop/users			
参数	类型	必填	说明
id	整数	必须	商家的ID
page	整数	可选	分页，缺省值为1
pcount	整数	可选	每页的数量，缺省为20

例子：

```
[
{"id":1,
"logo":"/phone2/images/namei2.gif",
"name":"name23",
"wb_uid":1884834632,
"gender":0,
"friend":true,
"follower":false,
"birthday":null}
]
```

"friend"表示该用户是否是当前用户的朋友；"follower"代表该用户是否关注了当前用户。

[2012—8—8] 新增time字段，表示用户在该商家的最后活跃时间。

[2012—10—30] 新增分页参数。

在商家的用户的统计数据根据签到来统计，而签到则是当用户进入商家聊天室时自动后台发送。

8 进入现场签到

/checkins			
参数	类型	必填	说明
lat	浮点数	必须	经度
lng	浮点数	必须	纬度
accuracy	整数	必须	经纬度的精确度
shop_id	整数	必须	现场商家id
user_id	整数	必须	签到用户id
od	整数	必须	实际签到的商家的排序
altitude	浮点数	可选	海拔高度
altacc	整数	可选	海拔高度的精确度

注意：该请求必须是POST请求。如果是GET请求，获得的是签到列表。

[2012-8-10] 新增od字段，表示用户实际签到的商家在现场商家列表中的顺序位置，从1开始，也就是从/aroundme/shops中获得的商家列表中，那个被用户选中了。

9 获得用户基本信息

/user_info/get			
参数	类型	必填	说明
id	整数	必须	用户id

例子：

访问: /user_info/get?id=1

返回值:

```
{
  "name":null,
  wb_uid:1884834632
  "gender":null,
  "logo":"/system/imgs/1/original/clojure.png?1339398140",
  "logo_thumb":"/system/imgs/1/thumb/clojure.png?1339398140"
  "birthday":null,
  "friend":true,
  "follower":false,
  "signature":"","job":null,"jobtype":null,"hobby":"Weiqi, go",
  "last": "1天以前 顺旺基(益乐路店)"
  "id":1
}
```

获得用户基本信息里有两个关系字段: **friend**和**follower**。这里的关系都是针对我的, 这里的我就是当前登录用户。如果我关注了该用户, 那么该用户就是我的**friend**; 如果该用户关注了我, 那么他就是我的**follower**。

[2012-8-8] 新增签名、职业、爱好等字段。

[2012-8-8] 新增**last**字段, 表示用户最后在脸上的时间和地点, 一个以空格分割的字符串。

10 获得用户的头像

/user_info/logo			
参数	类型	必填	说明
id	整数	必须	用户id
size	整数	可选	目前0代表原图；1代表thumb缩略图，大小为75*75；2代表thumb2缩略图，大小为150*150。默认为1。

输出的Response Header中包含“Img_url”头，这个头就是查看用户信息时显示的用户头像的路径。

例如：“Img_url:/system/imgs/1/thumb2/1.png?1339649979”

注意：采用阿里云存储后，会输出302重定向，而不是直接给二进制流。

11 获得用户的图片列表

/user_info/photos			
参数	类型	必填	说明
id	整数	必须	用户id

例子：

访问： /user_info/photos?id=1
返回值：

```
[
{"logo_thumb2":"/system/imgs/2/thumb2/2.png.png?1340616951","updated_at":"2012-06-12 14:06:16",
"id":2,"logo":"/system/imgs/2/original/2.png.png?1340616951","img_file_size":8188},
{"logo_thumb2":"/system/imgs/4/thumb2/2.png.png?1340617039","updated_at":"2012-06-12 14:06:16",
"id":4,"logo":"/system/imgs/4/original/2.png.png?1340617039","img_file_size":8188}
]
```

图片列表中的第一张图片就是用户的头像。

12 获得当前登录用户基本信息

/user_info/get_self			
参数	类型	必填	说明

例子：

访问: /user_info/get_self

返回值:

```
{  
  "name":null,  
  wb_uid:1884834632  
  "gender":null,  
  "birthday":null,  
  "invisible":0,  
  "logo":"/system/imgs/1/original/clojure.png?1339398140",  
  "logo_thumb":"/system/imgs/1/thumb/clojure.png?1339398140"  
  "password":"c84dad462d5b7282",  
  "id":1  
}
```

13 设置当前登录用户基本信息

/user_info/set			
参数	类型	必填	说明
name	字符串	选填	用户的名字. 长度小于64
gender	数字	选填	用户的性别, 未设置0男1女2
birthday	字符串	选填	用户的生日,格式如2012-06-01
signature	字符串	选填	签名档. 长度小于255
job	字符串	选填	职业
jobtype	整数	选填	职业类别
hobby	字符串	选填	爱好. 长度小于255
invisible	数字	选填	0: 不隐身,1: 对陌生人隐身,2: 全部隐身

注意：该请求必须是POST请求。

用户初次登录时获取新浪微博的名称 / 性别 / 出生日期等信息并提交到服务端。以后更新时可以只更新其中的某个字段。

隐身的效果是：1、在商家的用户列表中不在出现该用户2、不更新自己的位置信息3、用户进入商家时，不发打招呼的信息。

例子：

访问: curl -b "_session_id=ead9ac4f6291c55bb467ad4138eca2ed"
-F "name=newname" /user_info/set

返回值:

```
{
  "name":newname,
  "gender":null,
  "birthday":null,
  "logo":"/system/imgs/1/original/clojure.png?1339398140",
  "logo_thumb":"/system/imgs/1/thumb/clojure.png?1339398140"
  "id":1
}
```

14 当前登录用户上传图片

/user_logos/create			
参数	类型	必填	说明
user_logo[img]	file	必须	头像文件

注意:

1. 该请求必须是POST请求，enctype="multipart/form-data"。
2. 所上传文件的type必须是image/jpeg', 'image/gif' 或者'image/png'。
3. 图片文件必须小于5M。

- 4. 图片名为logo；缩略图分为两种大小，logo_thumb为75*75的png，logo_thumb2为150*150的png。

例子：

访问： curl -F "user_logo[img]=@firefox.png;type=image/png" /user_logos

```
{
  "logo_thumb2": "/system/imgs/6/thumb2/csky2.png?1340779488",
  "updated_at": "2012-06-27T06:44:48Z",
  "id": 6,
  "logo": "/system/imgs/6/original/csky2.png?1340779488",
  "img_file_size": 156604,
  "user_id": 3,
  "logo_thumb": "/system/imgs/6/thumb/csky2.png?1340779488"
}
```

15 当前登录用户批量调整所有图片的位置

/user_logos/change_all_position			
参数	类型	必填	说明
ids	逗号分割的整数	必须	所有图片的id按新的循序排列

说明：

比如原有图片的循序是2468，新的循序是8642，此时需要调用本接口，传递ids=8,6,4,2。此时服务器会更新所有图片的排序，而客户端只需要发送一个请求。

例子:

```
访问: curl -F "ids=3,1,4"  
/user_logos/position
```

16 当前登录用户删除图片

/user_logos/delete			
参数	类型	必填	说明
id	整数	必须	图片的id

例子:

```
访问: curl -F "id=2"  
/user_logos/delete
```

输出:

```
{"deleted": "2"}
```


17 添加我关注的人

/follows/create			
参数	类型	必填	说明
user_id	整数	必须	当前登录用户的id
follow_id	整数	必须	关注用户的id

注意：该请求必须是POST请求。

例子：

```
访问： curl -b "_session_id=f03bef9371c119b6fcecbeefdaaac1b2"
        -d "user_id=2&follow_id=1" /follows
```

返回值：

```
{
  "id":1,
  "user_id":2,
  "follow_id":1
}
```

18 删除我的关注的人

/follows/delete			
参数	类型	必填	说明
user_id	整数	必须	当前登录用户的id
follow_id	整数	必须	关注用户的id

注意：该请求必须是POST请求。

例子：

```
访问： curl -b "_session_id=f03bef9371c119b6fcecbeefdaaac1b2"
        -d "user_id=2&follow_id=3" /follows/delete
```

返回值：

```
{
  "deleted":3
}
```

19 粉丝列表

/follow_info/followers			
参数	类型	必填	说明
id	整数	必须	用户的id
page	整数	可选	分页，缺省值为1
pcount	整数	可选	每页的数量，缺省为20
name	字符串	可选	用户的名字
hash	整数	可选	是否返回hash，缺省值为0

例子：

访问: `/follow_info/followers?id=2`

返回值:

```
[
{"count":1}
{"data":
{"id":1,
"name":"name23",
"wb_uid":1884834632,
"gender":0,
"friend":true,
"follower":true,
"birthday":null}
}
]
```

其中 `count` 是符合条件的粉丝的总数。

获得用户基本信息里有两个关系字段: `friend`和`follower`。这里的关系都是针对我的, 这里的我就是给定ID的用户。如果我关注了该用户, 那么该用户就是我的`friend`; 如果该用户关注了我, 那么他就是我的`follower`。

这里返回的所有用户, 其`"follower"`值一定为`true`。

20 关注(好友)列表

/follow_info/friends			
参数	类型	必填	说明
id	整数	必须	用户的id
page	整数	可选	分页，缺省值为1
pcount	整数	可选	每页的数量，缺省为20
name	字符串	可选	用户的名字
hash	整数	可选	是否返回hash，缺省值为0

返回的内容格式同粉丝列表一样
这里返回的所有用户，其”friend”值一定为true。

21 黑名单列表

/blacklists			
参数	类型	必填	说明
id	整数	必须	用户的id
page	整数	可选	分页，缺省值为1
pcount	整数	可选	每页的数量，缺省为20
name	字符串	可选	用户的名字
hash	整数	可选	是否返回hash，缺省值为0

返回的内容格式和好友列表一样。

22 添加黑名单

/blacklists/create			
参数	类型	必填	说明
user_id	整数	必须	当前登录用户的id
block_id	整数	必须	加黑阻止的用户的id
report	整数	可选	是否同时举报被加黑的用户，0不举报1举报

注意：该请求必须是POST请求。

返回值：

```
{
  "id":1,
  "user_id":2,
  "report":false,
  "block_id":1
}
```

23 删除黑名单

/blacklists/delete			
参数	类型	必填	说明
user_id	整数	必须	当前登录用户的id
block_id	整数	必须	加黑用户的id

注意：该请求必须是POST请求。

例子：

```
访问： curl -b "_session_id=f03bef9371c119b6fcecbeefdaaac1b2"
        -d "user_id=2&block_id=3" /blacklists/delete
```

返回值：

```
{
"deleted":3
}
```

24 XMPP协议接口

24.1 与Openfire聊天服务器接口

脸脸网的XMPP服务器采用的是Openfire。

1. 脸脸网的用户id加上"@dface.cn"就是openfire的jid，两个系统的密码一致。
2. 商家的id加上"@c.dface.cn"就是现场聊天室的jid。
3. 脸脸的用户名就是openfire的用户名，且也就是聊天时的用户名。
Xmpp协议允许用户在加入聊天室时取一个名字，脸脸的聊天室应该忽略该名字。
4. 脸脸中的关注和粉丝关系是单向的，和xmpp中的好友roster关系无关。
5. 只要用户登录，其xmpp协议中的presence就是在线，没有其它状态。

6. 脸脸中的隐身也和xmpp中的隐身状态无关。
7. xmpp中的任意两个用户可以发送消息，不管对方是否隐身，只要对方未设置黑名单即可。如果对方未登录，那么就是离线消息。

24.2 脸脸中的聊天用户的三种类型

1. 管理员，其jid固定为"502e6303421aa918ba000001@dface.cn";
2. 商家，其jid为's+商家id@dface.cn';
3. 个人，其jid为'个人id@dface.cn'。

如果一个jid以字母s开头，那么它一定是商家。

24.3 摇一摇及其消息格式

目前，用户在聊天室摇一摇时，客户端发送“用户名摇了摇手机和大家say hello”给服务器。这导致要分析摇一摇数据很困难，所以要为摇一摇定义特殊的消息格式。

摇一摇的消息格式为：

[摇一摇:\$name]

其它客户端收到摇一摇的消息后，再将其转换为文字描述。

24.4 个人之间聊天的消息发送状态确认

XMPP协议的消息发送状态确认参考规范“[XEP-0022: Message Events](#)”。其定义了四种消息事件：Offline、Delivered、Displayed、Composing。Dface只需要其中的两种。

当接收端收到消息时，发送`delivered`确认消息，例如：

```
<message id="Kk98S-16" to="s6@dface.cn/ylt">  
<x xmlns="jabber:x:event"><delivered/><id>purplea5e6669c</id></x>  
</message>
```

当接收端展示消息时，发送`displayed`确认消息，例如：

```
<message id="Kk98S-17" to="s6@dface.cn/ylt">  
<x xmlns="jabber:x:event"><displayed/><id>purplea5e6669c</id></x>  
</message>
```

发送端根据从接收端获得的状态通知更改单条消息的发送状态。

只有类型是`message`，且`body`不为空的消息需要确认状态。

当本地无法发送消息时（比如无网络、或者连接不上xmpp服务器），消息的状态为“发送失败”。发送失败的消息可以再次发送。

25 在聊天室发图

聊天发图主要流程是：客户端选择（拍摄）一张照片，通过http上传到服务器。上传完成后在xmpp里发送一个特定格式的消息。其它客户端收到消息后获取图片显示并发送回执消息。

25.1 上传图片

/photos/create			
参数	类型	必填	说明
photo[img]	file	必须	头像文件
photo[room]	字符串	必须	聊天室的id
photo[weibo]	0/1	必须	是否同步发送到微博
photo[desc]	字符串	可选	图像的说明文字

注意：

- 1. 该请求必须是POST请求，enctype=”multipart/form-data”。
- 2. 所上传文件的type必须是image/jpeg’, ’image/gif’ 或者’image/png’。
- 3. 图片文件必须小于5M。
- 4. 图片名为logo；缩略图分logo_thumb2为150*150的png。

例子：

访问: `curl -F "photo[img]=@firefox.png;type=image/png" /photos`

```
{ "_id": "509a1ffcbe4b1982a4000002",  
  "weibo": false, "room": "1",  
  "user_id": "502e61bfbe4b1921da000005",  
  "img_tmp": "20121107-1646-42114-4251/111.jpg",  
  "logo": "/uploads/tmp/20121107-1646-42114-4251/111.jpg",  
  "logo_thumb2": null, "id": "509a1ffcbe4b1982a4000002" }
```

当返回的响应中包含字段, 且logo_thumb2为null时, 说明此时图片上传到了服务器, 但是还未启动后台处理。后台异步处理图片, 包括生成缩略图以及上传到阿里云。

25.2 发送消息

当图片发送完成后, 客户端发送一个xmpp消息, 其中的body内容格式为:

```
[img:$id]$desc
```

比如上面的图片发送成功后, 发送消息"`[img:502fd10abe4b19ed48000002]`"。其中\$desc是可选的图片说明文字。

25.3 在聊天室收到消息后获取图片

接收端收到消息后, 判断body的内容是否符合图片消息的格式。如果符合就调用下面的接口获取图片。

/photos/show			
参数	类型	必填	说明
id	字符串	必须	图片id
size	整数	可选	目前0代表原图；2代表thumb2缩略图，大小为150*150。默认为2。

注意：聊天时发图，取消的75*75大小的缩略图。[2012-11-7]

聊天室里发图不需要发送已读回执消息。

26 个人聊天时发图

26.1 个人聊天时上传图片

/photo2s/create			
参数	类型	必填	说明
photo[img]	file	必须	头像文件
photo[to_uid]	字符串	必须	接收人的id

26.2 发送消息

当图片发送完成后，客户端发送一个xmpp消息，其中的body内容格式为：

[img:\$id]

26.3 在个人聊天时收到消息后获取图片

/photo2s/show			
参数	类型	必填	说明
id	字符串	必须	图片id
size	整数	可选	目前0代表原图；2代表thumb2缩略图，大小为150*150。默认为2。

注意：聊天时发图，取消的75*75大小的缩略图。[2012-11-7]

26.4 消息状态

和普通的文字消息一样，接收者收到图片后发送状态更新消息（收到 / 已读）给发送者。发送者将状态显示在图片旁边。

27 优惠券

27.1 商家推送优惠券

当用户使用脸脸时，比如签到 / 摇一摇等，可以收到商家推送过来的优惠券。优惠券也是一个xmpp消息，其中的body内容格式为：

[优惠券:\$name:\$shop:\$id:\$time]

其中，name是优惠券的名称、shop是商家的名称，id是该优惠券的id，time是该张优惠券的下发时间。

同一张id的优惠券，如果time不同的话，分开显示和使用。

所有优惠券都统一显示在会话中的“优惠券”文件夹中。优惠券的消息状态和其他消息同样处理。

27.2 优惠券展示

优惠券以图片的方式展示，由服务器端生成该图片。当用户进入优惠券文件夹查看优惠券时，调用下面的接口获得优惠券的图片。

/coupons/img			
参数	类型	必填	说明
id	字符串	必填	该优惠券的id
size	整数	可选	目前0代表原图(580, 224)；1代表缩略图，大小为(290,112)。默认为1。

27.3 优惠券使用

目前，所有优惠券都只支持单次使用，使用后即过期。当用户确认使用优惠券时，调用如下的接口：

/coupons/use			
参数	类型	必填	说明
id	字符串	必填	该优惠券的id

28 关于程序各页面之间的切换规则

程序的首页是选择现场页面。现场则是一个聊天室页面。

1、程序干净启动时，要判断用户是否处于登录状态。

1.1如果已经登录，进入选择现场页面，并从服务器端获取新的商家列表。

1.2如果没有登录，进入登录页面。

2、程序从睡眠状态被激活时，要判断距离上次的运行时间。

2.1 如果不足12个小时，上次被任务切换时停留在那个页面就仍然停留在该页面。

2.2 如果超过12个小时，进入选择现场页面，并从服务器端获取新的商家列表。

3、从现场返回到选择现场时，判断当前的经纬度和上次进入现场时的经纬度。

3.1如果超过1000米，提示用户是否刷新现场位置。

3.2 如果不足1000米，不处理。用户要选择其它现场的话，从滚轮中选择。

4、从其它tab点击现场时，判断用户是否摇进过现场

4.1 如果摇进过某个现场，直接进入该现场聊天室。进入时要特殊处理下部的tab，提供动画。只有这种情况需要这么特殊处理。

4.2 如果没有摇，那么就是选择现场页面。

5、从附近里点击商家的时候。不进入现场。显示商家的聊天室。但是该聊天室不是现场，没有聊天工具栏。在附近里进入商家聊天室时不能聊天。