V5KF 小程序用户信息推送接口

（V1.0）

深圳市智客网络科技有限公司

版权所有 不得复制

目录·

1 接口介绍

用户基本信息推送接口用于微信小程序应用中向 V5KF 服务器推送其访客的基本用户信息。

用户基本信息包括：

- 用户昵称：用于座席端显示该客户昵称信息
- 用户性别：用于座席端显示该客户性别信息
- 用户头像：用户座席端显示该客户头像信息
- VIP 等级：用于给客户分配座席时设定优先级别
- 专属客服代表：用于指定服务该客户的客服人员
- 用户区域信息：用户注册位置区域信息

如果 V5 座席平台中需要显示来自微信小程序客户的基本信息时，必须遵循《V5KF 小程序用户信息推送接口》规范，向 V5 服务器推送客户基本信息。

2 接口规范

2.1接口规范

URL：推送用户基本信息接口 URL，由 V5KF 提供。第三方在其微信小程序应用中获取进入客服会话客户的基本信息，然后 POST 到 V5 服务器。请求 URL：

http://chat.v5kf.com/public/wxapp/cinfo/$APPID/callback?nonce=$NONCE&timestamp=$TIMESTAMP&signature=$SIGNATRUE

请求 URL

参数	说明
$APPID	微信小程序对应的 AppID。

请求参数：

参数	说明
token	固定字符串，由 V5KF 提供，请向客服咨询获取（不包含在请求 URL 中）。
nonce	随机码。
timestamp	时间戳。
signature	生成的签名码(content, nonce, timestamp, token 连接成字符串进行 sha1 运算)。

请求 Json 内容：

参数	说明
openId	用户对应微信小程序 OpenID
nickName	用户昵称。字符串（可选）。
gender	用户性别。整数（可选。0-未知，1-男，2-女）
avatarUrl	用户头像 URL 地址。字符串（可选）
city	用户所在城市。字符串（可选）
province	用户所在省份。字符串（可选）
country	用户所在国家。字符串（可选）
vip	用户 VIP 等级（可选）。整数（可选。0-5，5 表示最高 VIP 登记，拥有最高调度优先权；0 表示普通客户）
csr	专属客服人员 ID。整数（可选）。

2.2接口签名算法

接口签名算法参考如下（PHP）：

```
$token      = 'abcdef';
$nonce      = RANDOM;
$timestamp  = TIME;
$json       = POST_CONTENT
$tmp = $json.$nonce.$timestamp.$token;
$signature = sha1($tmp);
```

3 小程序调用示例

3.1获取用户信息

通过 wx.getUserInfo 接口获取用户信息，参考：

https://mp.weixin.qq.com/debug/wxadoc/dev/api/open.html#wxgetuserinfoobject

3.2获取 openId

通过 wx.login 接口获取到 code，用 code 通过接口

https://api.weixin.qq.com/sns/jscode2session?appid=APPID&secret=SECRET&js_code=JSCODE&grant_type=authorization_code

换取到 openId（此步骤小程序官方建议在服务端获取，小程序内无法添加此安全域名）。

参考：

https://mp.weixin.qq.com/debug/wxadoc/dev/api/api-login.html#wxloginobject

3.3提交用户信息

将小程序获取到的用户信息和 openId 组合到 globalData.userInfo，在客户点击“客服会话”按钮时，POST 提交到 V5 服务器。

本示例用户信息保存在 app.js 的 globalData 中，结构如下：

```
App({
  globalData: {
    userInfo: {
      avatarUrl: '头像url',
      city: '城市',
      country: '国家',
      gender: '性别',
      nickname: '昵称',
      province: '省',
      openId: 'openId'
    },
    appid: '您的小程序App_ID',
    // ……
  },
  //……
}
```

客服会话按钮示例（注意绑定客服消息回调 bindcontact）：

参考：https://mp.weixin.qq.com/debug/wxadoc/dev/component/button.html

```
<button
    open-type="contact"
    send-message-title="{{title}}"
    show-message-card="{{true}}"
    bindtap="bindtap"
    class="contact-btn"
    type="default-light"
    size="23"
    session-from="weapp"
    >
</button>
```

bindcontact 客服消息回调示例：

```
bindtap: function (e) {
  // 提交用户信息
  getApp().postUserInfo();
}
```

可在 app.js 中统一处理 postUserInfo：

```
Var util = require('./util.js'); // 导入sha1工具函数

postUserInfo: function() {
  var userInfo = this.globalData.userInfo;
  var data = JSON.stringify(userInfo);
  var timestamp = Date.now(),
    nonce = Math.random().toString(36).substr(2), //随机字符串
    token = 'abcdef', //小程序配置的token（后台小程序接入处配置可见）
    sign = util.sha1(data + nonce + timestamp +
token).toLowerCase(); //组合串进行签名计算
  //提交到v5
  wx.request({
    url: 'https://chat.v5kf.com/public/wxapp/cinfo/' +
this.globalData.appid + '/callback?signature=' + sign + '&nonce='
+ nonce + '&timestamp=' + timestamp,
    method: 'POST',
    data: data,
    header: {
      content-type': 'application/json'
    },
    success: function (res) {
      console.log('postUserInfo success');
    }
  });
}
```

其中 sha1 算法实现如下 util.js：

```
function encodeUTF8(s) {
    var i, r = [], c, x;
    for (i = 0; i < s.length; i++)
        if ((c = s.charCodeAt(i)) < 0x80) r.push(c);
        else if (c < 0x800) r.push(0xC0 + (c >> 6 & 0x1F), 0x80 + (c & 0x3F));
        else {
            if ((x = c ^ 0xD800) >> 10 == 0) //对四字节UTF-16转换为Unicode
                c = (x << 10) + (s.charCodeAt(++i) ^ 0xDC00) + 0x10000,
                    r.push(0xF0 + (c >> 18 & 0x7), 0x80 + (c >> 12 & 0x3F));
            else r.push(0xE0 + (c >> 12 & 0xF));
            r.push(0x80 + (c >> 6 & 0x3F), 0x80 + (c & 0x3F));
        };
    return r;
}

// 字符串加密成 hex 字符串
function sha1(s) {
    var data = new Uint8Array(encodeUTF8(s))
    var i, j, t;
    var l = ((data.length + 8) >>> 6 << 4) + 16, s = new Uint8Array(l << 2);
    s.set(new Uint8Array(data.buffer)), s = new Uint32Array(s.buffer);
    for (t = new DataView(s.buffer), i = 0; i < l; i++)s[i] = t.getUint32(i << 2);
    s[data.length >> 2] |= 0x80 << (24 - (data.length & 3) * 8);
    s[l - 1] = data.length << 3;
    var w = [], f = [
        function () { return m[1] & m[2] | ~m[1] & m[3]; },
        function () { return m[1] ^ m[2] ^ m[3]; },
        function () { return m[1] & m[2] | m[1] & m[3] | m[2] & m[3]; },
        function () { return m[1] ^ m[2] ^ m[3]; }
    ], rol = function (n, c) { return n << c | n >>> (32 - c); },
        k = [1518500249, 1859775393, -1894007588, -899497514],
        m = [1732584193, -271733879, null, null, -1009589776];
    m[2] = ~m[0], m[3] = ~m[1];
    for (i = 0; i < s.length; i += 16) {
        var o = m.slice(0);
        for (j = 0; j < 80; j++)
            w[j] = j < 16 ? s[i + j] : rol(w[j - 3] ^ w[j - 8] ^ w[j - 14] ^ w[j - 16], 1),
                t = rol(m[0], 5) + f[j / 20 | 0]() + m[4] + w[j] + k[j / 20 | 0] | 0,
                m[1] = rol(m[1], 30), m.pop(), m.unshift(t);
        for (j = 0; j < 5; j++)m[j] = m[j] + o[j] | 0;
    };
    t = new DataView(new Uint32Array(m).buffer);
    for (var i = 0; i < 5; i++)m[i] = t.getUint32(i << 2);
```

```
    var hex = Array.prototype.map.call(new Uint8Array(new Uint32Array(m).buffer),
function (e) {
        return (e < 16 ? "0" : "") + e.toString(16);
    }).join("");
    return hex;
}

module.exports = {
    sha1: sha1
}
```