

# 插件使用说明

## 1. 安装

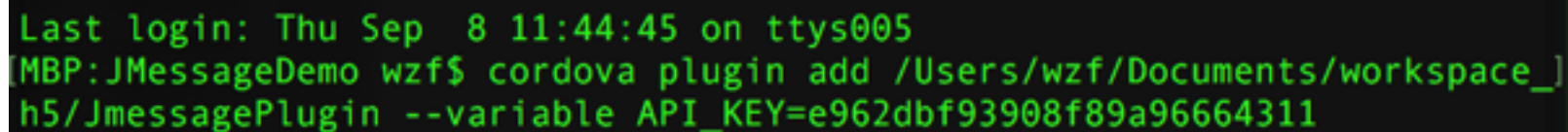
打开CMD或终端，进入到项目根目录，执行如下命令。

### 1.1 安装命令

用于安装JMessage插件：

`cordova plugin add [插件的根目录] --variable API_KEY=[您申请的JMessage AppKey]`

e.g: `cordova plugin add /Users/wzf/Documents/workspace_h5/JmessagePlugin --variable API_KEY=e962dbf93908f89a96664311`

A terminal window with a black background and green text. The text shows the last login time and the command used to install the JMessage plugin.

```
Last login: Thu Sep  8 11:44:45 on ttys005
[MBP:JMessageDemo wzf$ cordova plugin add /Users/wzf/Documents/workspace_
h5/JmessagePlugin --variable API_KEY=e962dbf93908f89a96664311
```

### 1.2 删除命令

用于删除JMessage插件：

`cordova plugin remove cordova-plugin-jmessage`

### 1.3 编译插件

`ionic build` 或者 `sudo ionic build`

## 2. 初始化

### 2.1 AppKey 配置

安装插件时，会自动配置好AppKey值，如果安装完插件后，发现AppKey并未配置成功，请进入移动终端平台项目中，进行手动配置：

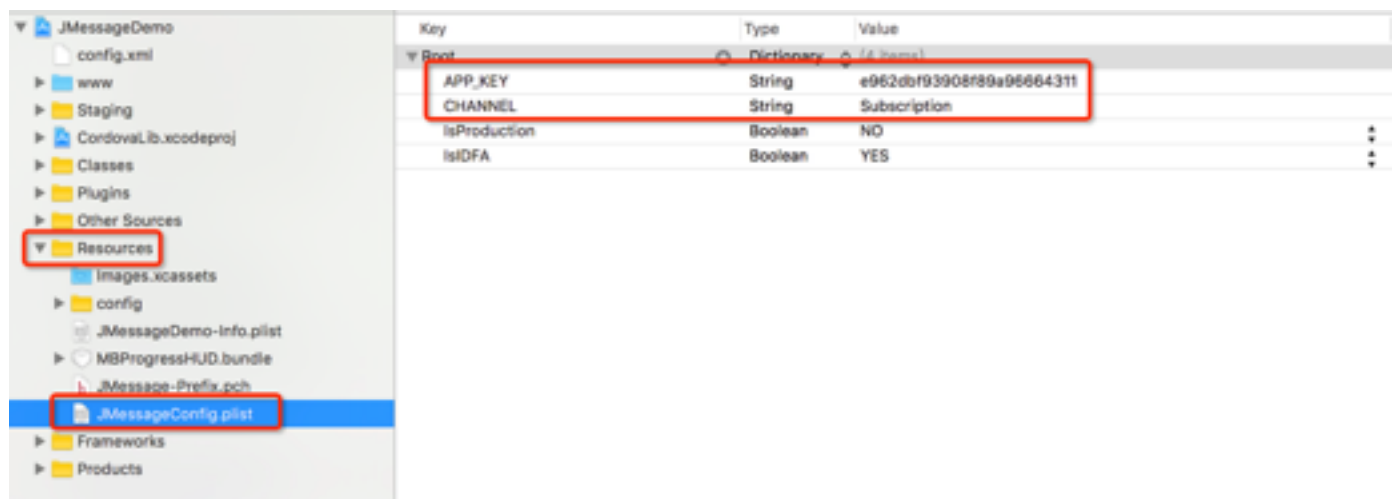
#### 2.1.1 android 配置

打开AndroidManifest.xml，修改APP\_KEY和CHANNEL的值：

```
<category android:name="com.atme.shuixing" />
</intent-filter>
</activity>
<service android:enabled="true" android:exported="false" android:name=
    "cn.jpush.android.service.DownloadService" />
<receiver android:exported="false" android:name="cn.jpush.android.service.AlarmReceiver" />
<meta-data android:name="JPUSH_CHANNEL" android:value="developer-default" />
<meta-data android:name="JPUSH_APPKEY" android:value="e962dbf93908f89a96664311" />
<receiver android:enabled="true" android:exported="false" android:name=
    "cn.jpush.im.android.helpers.IMReceiver">
    <intent-filter android:priority="1000">
        <action android:name="cn.jpush.im.android.action.IM_RESPONSE" />
    </intent-filter>
</receiver>
```

#### 2.1.2 iOS 配置

找到JMessageConfig.plist文件，输入 / 修改APP\_KEY和CHANNEL值。



## 2.2 插件初始化

### 2.2.1 初始化API

```
window.plugins.jmessagePlugin.init();
```

### 2.2.2 调用实例

```
.run(function($ionicPlatform) {  
  $ionicPlatform.ready(function() {  
    // Hide the accessory bar by default (remove this to show it  
    // for form inputs)  
    if (window.cordova && window.cordova.plugins && window.cordova.plugins.Keyboard) {  
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);  
      cordova.plugins.Keyboard.disableScroll(true);  
    }  
  
    if (window.StatusBar) {  
      // org.apache.cordova.statusbar required  
      StatusBar.styleDefault();  
    }  
  
    // 进入页面时初始化JMessagePlugin  
    window.plugins.jmessagePlugin.init();  
  });  
});  
  
.config(function($stateProvider, $urlRouterProvider) {
```

在项目运行时，调用`window.plugins.jmessagePlugin.init()`，初始化插件。

### 3. 登录注册

#### 3.1 图示

iPhone 6 - iPhone 6 / iOS 9.3 (13E230)

运营商 下午2:28

Register / Login Close

username

password

Register

username

password

Log in

退出登录

## 3.2 登录

### 3.2.1 登录API

```
// @param username      用户名
// @param password      密码
// @param suc_fuc(data)  API成功回调函数,
// @param error_fuc(error) API失败回调函数,
window.plugins.jmessagePlugin.doLogin(username, password, suc_fuc(data), error_fuc(error));
```

### 3.2.2 登录API实例

```
// 登录
$scope.doLogin = function() {

    var username = $scope.loginData.username;
    var password = $scope.loginData.password;

    window.plugins.jmessagePlugin.login(username, password,
    function (status) {
        alert('登录成功!');
        $scope.closeLogin();
    },
    function (msg) {
        alert("登录失败: " + msg);
    });
};
```

## 3.3 注册

### 3.3.1 注册API

```
// @param username      用户名
// @param password      密码
// @param suc_fuc(data)  API成功回调函数,
// @param error_fuc(error) API失败回调函数,
window.plugins.jmessagePlugin.doRegister(username, password, suc_fuc(data), error_fuc(error));
```

### 3.3.2 注册API实例

```
// 注册
$scope.doRegister = function () {

    var username = $scope.loginData.username;
    var password = $scope.loginData.password;

    window.plugins.jmessagePlugin.doRegister(username,
    password,
    function (status) {
        alert('注册成功!');
        $scope.closeLogin();
    },
    function (msg) {
        alert("注册失败: " + msg);
    });
};
```

## 3.4 退出登录

### 3.4.1 退出登录API

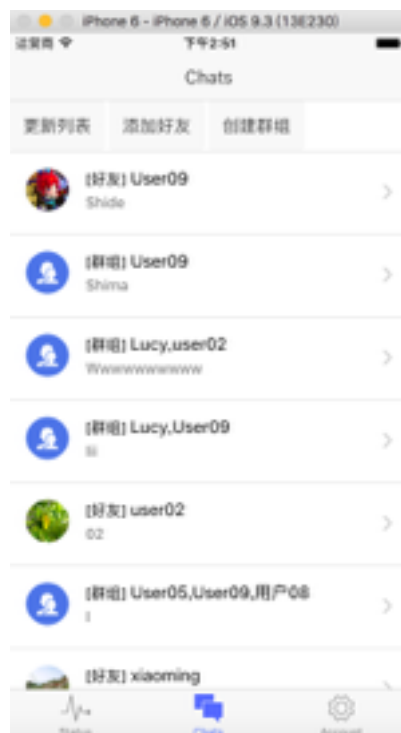
```
window.plugins.jmessagePlugin.exitLogin(suc_fuc(data), error_fuc(error));
```

### 3.4.2 退出登录实例

```
// 退出登录
$scope.exitLogin = function() {
    window.plugins.jmessagePlugin.exitLogin(
        function (data) {
            alert(data); // 成功退出登录
        },
        function (data) {
            alert(data); // 退出成功失败!
        })
};
```

## 4. 会话列表

4.1 图示



## 4.2 初始化

### 4.2.1 会话列表初始化API

```
// 初始化会话列表时，主要以callback      函数回调为主处理回调数据，该回调函数监视信息变动等通知。  
// @params suc_fuc(data)                  API调用成功回调  
// @params error_fuc(error)              API调用失败回调  
// @params callback(type, data)          自定义通知回调：type—通知类型；data—返回JSON数据；  
window.plugins.jmessagePlugin.initConverListCtrl(suc_fuc(data), error_fuc(error), callback(type, data));
```

### 4.2.2 会话列表初始化实例

```
// 初始化会话列表  
var initConverListCtrl = function () {  
    // 初始化会话列表项  
    window.plugins.jmessagePlugin.initConverListCtrl(  
        function (data) {  
            // alert("完成会话列表初始化");  
        },  
        function (error) {  
            // alert("会话列表初始化失败");  
        },  
        // 会话变动通知回调接口  
        function (type, resultData) {  
            /**  
             * 会话信息变动,刷新列表  
             * 会话内容有变动时,会触发此回调接口,返回会话信息数据  
             * if(type == ),  
             */  
            loadConverListData();  
        }  
    );  
}
```



## 4.3 获取会话列表

### 4.3.1 获取会话列表API

```
// 获取会话列表时，主要以callback      函数回调为主处理回调数据，该回调函数返回会话消息JSON串。  
// @params suc_fuc(data)                API调用成功回调  
// @params error_fuc(error)             API调用失败回调  
// @params callback(count, jsonStr) 自定义回调：count—未读消息数；jsonStr—返回会话列表JSON数据；  
window.plugins.jmessagePlugin.getConversationList(suc_fuc(data), error_fuc(error), callback(count, jsonStr));
```

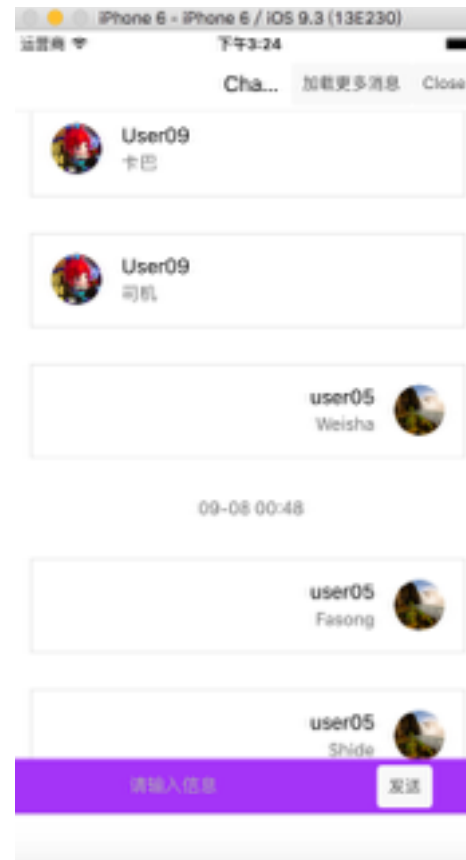
### 4.3.2 获取会话列表实例

```
// 加载会话列表数据  
var loadConverListData = function(){  
    window.plugins.jmessagePlugin.getConversationList(  
        function(data){  
        },  
        function(error){  
        },  
        function(count, converListJson){  
            /**  
             * 刷新会话列表页面数据  
             * @params count 未读消息数  
             * @params converListJson 会话列表JSON串  
             */  
            refreshConverListData(angular.fromJson(converListJson));  
        }  
    );  
};
```

### 4.3.3 改变信息读取状态 [待完善]

## 5. 会话详情

### 5.1 图例



### 5.2 初始化

#### 5.2.1 初始化会话详情API

```

// 初始化会话详情时，主要以callback      函数回调为主处理回调数据。
// @params targetId                      单聊用户名 (username) / 群聊ID (gid)
// @params conversationType              会话类型
// @params suc_fuc(data)                  API调用成功回调
// @params error_fuc(error)               API调用失败回调
// @params callback(type, data)           自定义回调：type—通知类型；data—根据type解析数据；

window.plugins.jmessagePlugin.initConverViewCtrl(targetId, suc_fuc(data), error_fuc(error), callback(type,
data));

```

### 5.2.2 初始化会话详情实例

```

// 初始化会话详情
var initConverViewCtrl = function (targetId, converType) {

    // 初始化会话列表项
    window.plugins.jmessagePlugin.initConverViewCtrl(
        targetId, // 单聊用户名 (username) / 群聊ID (gid)
        converType, // 会话类型
        function (data) { // 成功
        },
        function (error) { // 失败
        },
        function (type, resultData) {

            // 异常通知
            if(type == '10101100'){
                alert(resultData);
            }

            // 刷新消息列表
            if(type == '10101102'){
                if(resultData){
                    var resultDataJson = angular.fromJson(resultData);
                    refreshMsgListData(resultDataJson);
                }
            }
        }
    );
}

```

## 5.3 消息列表 [\[见.加载更多消息列表\]](#)

### 5.3.1 获取消息列表API

### 5.3.2 获取消息列表实例

## 5.4 加载更多消息列表

### 5.4.1 加载更多消息列表API

// 用户滑动消息列表时加载更多消息

// @params start—页码; offset—增量; [此两个参数为保留参数, 本版本插件默认增量:50]

// @params suc\_fuc(data) API调用成功回调

// @params error\_fuc(error) API调用失败回调

window.plugins.jmessagePlugin.loadMoreConverDetailMsgList(start, offset, suc\_fuc(data), error\_fuc(error));

### 5.4.2 加载更多消息列表实例

```
// 加载更多消息列表(分页显示消息列表, 目前页码和每页条数参数保留)
$scope.loadMoreConverDetailMsgList = function () {
    var aStart = '1'; var aOffset = '30';
    window.plugins.jmessagePlugin.loadMoreConverDetailMsgList(
        aStart, aOffset,
        function (data) {
            var moreDataJson = angular.fromJson(data);
            refreshAfterLoadMoremsg(moreDataJson);
        }, function (error) {
            // 加载更多消息列表失败
            console.log('jmessage: ----- 加载更多消息列表失败: ' + error);
        }
    );
}
```

## 5.5 发送文本消息

### 5.5.1 发送文本消息API

```
// 用户发送文本消息
// @params content      发送的文本信息
// @params suc_fuc(data)  API调用成功回调
// @params error_fuc(error) API调用失败回调
window.plugins.jmessagePlugin.sendMessage(content, suc_fuc(data), error_fuc(error));
```

### 5.5.2 发送文本消息实例

```
$scope.sendMessage = function () {
    var msgContent = $scope.input.message;

    // 调用发送信息API
    window.plugins.jmessagePlugin.sendMessage(
        msgContent,
        function (data) { // 发送成功

            // 发送消息成功,刷新页面通知
            $scope.$broadcast('refresh.page.data', data);
        },
        function (error) { // 发送失败
            alert('消息发送失败!');
        }
    );

    // 清空输入框内容
    $scope.input = {'message': ''};
}
```

5.6 发送图片消息 [待完善]

5.7 发送语音消息 [待完善]

## 6. 添加好友

### 6.1 图例



### 6.2 添加好友API

```
// 用户添加好友
// @params username      用户名
// @params suc_fuc(data)  API调用成功回调
// @params error_fuc(error) API调用失败回调
window.plugins.jmessagePlugin.sendMessage(username, suc_fuc(data), error_fuc(error));
```

## 6.3 添加好友实例

```
// 添加好友函数
// @params username 用户名
var addFriend = function (username) {
    window.plugins.jmessagePlugin.addFriend(
        username,
        function (data) {
            alert("添加好友成功");
            toChatDetail(data);
        },
        function (error) {
            alert(error);
        }
    );
}
```

## 7. 创建群组 [待完善]

### 7.1 图例

### 7.2 创建群组会话

#### 7.2.1 创建群组会话API

```
// 创建群组接口
// @param groupName    群组名称
// @param desc          群组描述
// @param suc_fuc       成功回调函数
// @param error_fuc     失败回调函数
window.plugins.jmessagePlugin.createGroup(groupName, desc, suc_fuc(data), error_fuc(error));
```

### 7.2.2 创建群组会话实例

```
// 创建群组
$scope.createGroup = function () {
    var defaultGroupName = ''; // 群组名称,默认为空,可通过修改群组名称接口来修改
    var defaultGroupDesc = ''; // 群组描述

    window.plugins.jmessagePlugin.createGroup(
        defaultGroupName,
        defaultGroupDesc,

        // 创建群组成功后回调该函数,返回值为会话信息JSON串
        function(data){

            console.log('+++++++ 创建群组成功: ' + data);

            if(data){
                var conversation = angular.fromJson(data);

                console.log('===== 创建群组会话: ' + conversation);
                // 跳转到会话详情页面
                toChatDetail(conversation);
            }
        },

        // 创建群组失败回调
        function(error){
            //var errorJson = angular.fromJson(error);
            alert(error);
        });
}
```



## 8. 我的信息 [待完善]

### 8.1 我的信息详情

```
[ {
  "avatarPath": "...../thumb/5155A691A17F3F1E.jpg", // 会话头像 [单聊 | 群组]
  "unreadCount": "99", // 未读信息数
  "targetAppKey": "e962dbf93908f89a96664311", // JMessage AppKey
  "latestMessage": { // 单聊 / 群聊最后发送的消息信息
    "msgId": "msgId_1473267264853978", // 消息ID
    "targetAppKey": "e962dbf93908f89a96664311", // 会话AppKey
    "fromAppKey": "e962dbf93908f89a96664311", // 消息来源的AppKey
    "isReceived": "0", // 是否为接收的消息: 0-发送的消息; 1-接收的消息;
    "contentType": "1", // 内容类型: 1-文本; 2-图片; 3-语音; 4-事件;
    "otherSide": "user05", // 发送消息的用户名
    "content": { // 消息内容
      "text": "Shide" // 内容文本 / 图片 / 语音 / 事件
    },
    "messageJson": { // 会话消息对象JSON串
      "target_type": "single", // 会话类型: single-单聊; group-群聊;
      "target_id": null, // ...
      "target_name": null, // ...
      "from_type": "user", // ...
      "from_id": "user05", // ...
      "from_name": "user05", // ...
      "from_platform": "i", // ...
      "create_time": 1473267264, // ...
      "msg_type": "text", // ...
      "msg_body": { // ...
        "text": "Shide" // ...
      },
      "from_appkey": "e962dbf93908f89a96664311" // ...
    },
    "serverMessageId": "114410054", // ...
    "status": "5" // 状态
  },
  "title": "User09", // 会话标题
  "targetId": "User09", // 会话拥有者
  "conversationType": "1" // 会话类型: 1-单聊; 2-群聊;
```

## 8.2 上传头像

## 8.3 修改基础信息（昵称、性别、地址、个性签名）

### 8.3.1 设置昵称

```
// 设置昵称
// @params nickname          用户昵称
// @params suc_fuc(data)     API调用成功回调
// @params error_fuc(error)  API调用失败回调

window.plugins.jmessagePlugin.setNickname(nickname, suc_fuc(data), error_fuc(error));
```

## 8.4 修改密码

## 9. 好友信息 **[待完善]**

### 9.1 好友信息详情

## 10. 群组信息 **[待完善]**

### 10.1 获取群组信息

#### 10.1.1 创建群组会话API

```
// 获取群组信息（包含群组用户列表）
// @param groupId      群组ID
// @param suc_fuc       成功回调函数
// @param error_fuc     异常回调函数
window.plugins.jmessagePlugin.getGroupInfo(groupId, suc_fuc(data), error_fuc(error));
```

### 10.2 群组添加好友

#### 10.2.1 群组添加好友API（一）

```
// 添加好友到群组
// @param groupId      群组ID
// @param username     用户名称
// @param suc_fuc       成功回调函数
```

```
// @param error_fuc          异常回调函数
window.plugins.jmessagePlugin.addMemberToGroup(groupId, username, suc_fuc(data), error_fuc(error));
```

### 10.2.2 群组添加好友API (二)

```
// 批量添加好友到群组
// @param groupId            群组ID
// @param usernameArrayJson  用户名称集合JSON串
// @param suc_fuc            成功回调函数
// @param error_fuc          异常回调函数
window.plugins.jmessagePlugin.addMembersToGroup(groupId, usernameArrayJson, suc_fuc(data), error_fuc(error));
```

## 10.3 群组移除好友

### 10.3.1 移除添加好友API (一)

```
// 移除群组成员
// @param groupId    群组ID
// @param username    用户名称
// @param suc_fuc     成功回调函数
// @param error_fuc   异常回调函数
window.plugins.jmessagePlugin.removeMemberFromGroup(groupId, username, suc_fuc(data), error_fuc(error));
```

### 10.3.2 移除添加好友API (二)

```
// 批量移除群组成员
// @param groupId    群组ID
// @param usernameArrayJson  用户名称集合JSON串
// @param suc_fuc     成功回调函数
// @param error_fuc   异常回调函数
window.plugins.jmessagePlugin.removeMembersFromGroup(groupId, usernameArrayJson, suc_fuc(data), error_fuc(error));
```

## 10.4 群组好友列表 (群组信息中已返回)

(略)

## 10.5 修改群名称

### 10.5.1 修改群组名称

```
// 修改群组名称
// @param groupId      群组ID
// @param groupname     新群组名称
// @param suc_fuc       成功回调函数
// @param error_fuc     异常回调函数
window.plugins.jmessagePlugin.updateGroupName(groupId, groupname, suc_fuc(data), error_fuc(error));
```

## 10.6 设置消息免打扰

## 10.7 删除并退出群组

# 11. 设置 [待完善]

## 11.1 加入黑名单

## 11.2 设置免打扰

## 11.3 清空消息记录

## 12. JSON数据说明

### 12.1 会话列表JSON说明

### 12.2 消息列表JSON说明

#### 12.2.1 获取消息列表回调JSON数据

```
{
  "msgModelList":{...},    // 消息模型列表,根据msgId可获取消息内容模型 [文本|图片|语音|事件|时间]
  "msgIdList":{...},      // 消息ID列表, 消息显示、数量及排序,均依据此列表;
  "msgImgArray":{...},    // 点击图片,用于预览图片显示(待完善)
  "members":{...}         // 用户信息集合,可根据username获取用户对象
}
```

#### 12.2.2 获取消息ID列表JSON数据 [msgIdList]

```
"msgIdList": [
  "<NSObject: 0x7fce7a952920>", // 事件消息模型 [eg. 好友被删除事件]
  "186701",                    // 时间消息模型 [超过一定时间时, 消息列表会显示时间]
  "msgId_1473248225194640",    // 普通消息模型 [文本 | 图片 | 语音]
  "msgId_1473248233389515",
  "msgId_1473248235736573",
  "msgId_1473248240331457",
  "msgId_1473248287330437",
  "msgId_1473248290123352",
  "108371",
  "msgId_1473266895517014",
  "msgId_1473266911237453",
  "661278",
  "msgId_1473267211709482",
  "msgId_1473267264853978"
]
```

### 12.2.3 获取消息模型列表JSON数据 [msgModelList]

```
"msgModelList": {                                     // 消息模型列表
    "108371": {                                         // 消息对应的KEY, 值为msgIdList中的消息ID [msgId]
        "photoIndex": "0",                             // 预览图索引
        "contentSize": {                               // 文本视图展示的宽高 (用于气泡展示, iOS返回, Android未知)
            "width": "228.000000",                     // 文本宽度
            "height": "62.960938"                      // 文本高度
        },
        "contentHeight": "62.96094",
        "imageSize": {                                 // 发送的图片大小
            "width": "0.000000",
            "height": "0.000000"
        },
        "message": "",                                 // 消息内容 (若为图片 / 语音时, 则message为空)
        "isErrorMessage": "0",                         // 是否为事件消息模型, 是 则只显示异常信息messageError
        "messageError": "",                            // 事件消息内容
        "isTime": "1",                                 // 是否为时间消息模型, 是 则只显示messageTime
        "messageTime": "1473266895",                  // 消息发送时间
        "timeId": "108371"                             // 消息ID
    }
}
```

#### 12.2.4 获取用户信息集合JSON数据 [members]

```
"members": {                                     // 会话用户集合
  "User09": {                                     // 根据用户名获取用户信息对象
    "username": "User09",                       // 用户名
    "address": "",                              // 地址
    "password": "",                             // 密码
    "createTime": "",                          // 创建时间
    "noteText": "",                            // 签名
    "noteName": "",
    "avatar": "qiniu/image/5155A691A17F3F1E",   // 用户头像[判断是否上传头像]
    "blackList": "0",                          // 黑名单列表
    "star": "0",                               //
    "isInBlacklist": "0",                     // 用户是否在黑名单中
    "uid": "17227349",                        // 用户ID
    "thumbAvatarPath": "...../thumb/5155A691A17F3F1E.jpg", // 用户头像缩略图
    "token": "",                              // 口令
    "gender": "0",                            // 性别：0-男性；1-女性
    "resourceID": "5155A691A17F3F1E",
    "appKey": "e962dbf93908f89a96664311",
    "signature": "",
    "originAvatarPath": "...../large/5155A691A17F3F1E.jpg", // 用户头像原图
    "noDisturb": "0",                         //
    "thumbAvatarData": "",                    // 用户头像数据
    "isNoDisturb": "0",                      // 是否免打扰：0-否；1-是；
    "region": "",                            // 地区
    "nickname": "",                          // 昵称
    "birthday": ""                           // 生日
  }
}
```

