[朋友圈 1](#_Toc485712769)

[Create 1](#_Toc485712770)

[search 2](#_Toc485712771)

[Update 2](#_Toc485712772)

[Delete 3](#_Toc485712773)

[群 4](#_Toc485712774)

[Create 4](#_Toc485712775)

[Search 4](#_Toc485712776)

[Update 5](#_Toc485712777)

[Delete 8](#_Toc485712778)

[群事件（publicGroupEvent） 9](#_Toc485712779)

[Create 9](#_Toc485712780)

[Search 9](#_Toc485712781)

[群成员处理（memberPenalize） 10](#_Toc485712782)

[Create 10](#_Toc485712783)

[Search 11](#_Toc485712784)

[群交互（publicGroupInteraction） 12](#_Toc485712785)

[Create 12](#_Toc485712786)

[Search 12](#_Toc485712787)

[Delete 13](#_Toc485712788)

[用户加入的群(usersPublicGroup) 13](#_Toc485712789)

[Create 14](#_Toc485712790)

[Search 14](#_Toc485712791)

[Update 14](#_Toc485712792)

[Delete 14](#_Toc485712793)

# 朋友圈

## Create

**url:** [http://domain/user\_friend\_group/](http://domain/wdtp/create) POST

**输入参数：**

Values:{

recordInfo:{Name:{value:’aaaa’}},

method:0

}

**备注：**

1. 新建用户，创建默认的朋友圈，名称是“我的朋友”
2. 用户Id从session中获得

**步骤：**

1. 检查输入参数是否正确

2. 检查是有有session（用户是否登录）

3. 用户是否有权限建立朋友圈

4. 用户的朋友圈是否已经达到上限

5. 创建新的记录

## search

**url:** [http://domain/pyq/](http://domain/wdtp/create) POST

**输入参数：**

Values:{

recordInfo:{Name:{value:’aaaa’}},

method:1

}

**备注：**

1. 用户Id从session中获得

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. 从session中读取用户id

3. 配合传入的名称进行检索

## Update

**url:** [http://domain/pyq/](http://domain/wdtp/create) POST

**输入参数：**

Values:{

recordInfo:{Name:{value:’aaaa’}},

method:2,

updateSubField:{

friendsInGroup :{

From: object\_id,

To: object\_id,

eleArray: []

}

}

}

**备注：**

1. 用户Id从session中获得
2. 更新圈中朋友有3中情况：1：直接移除（删除朋友），2：从一个圈移动到另一个圈, 3：新朋友加入一个圈

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. 从session中读取当前用户id

3. 读取updateSubField中的from和to对应的记录，检查其owner是否为user\_id

4. 如果to有设置（说明操作是add或者move），检查eleArray中的元素是否位于from指定的记录的friendsInGroup字段中。

5. 如果to存在，而from不存在，说明是add操作，对totalFriendsNum执行+操作

6. 反之，from存在，而to不存在，说明是delete，对totalFriendsNum执行-操作

5. 根据from/to的设置情况修改from/to对应的记录（step3中读取），然后保存

## Delete

**url:** [http://domain/pyq/](http://domain/wdtp/create) POST

**输入参数：**

Values:{

recordInfo:{Name:{value:’aaaa’}},

method:3

recIdArr:[]

}

**备注：**

1. 用户Id从session中获得

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. 从session中读取当前用户id

3. 读取recIdArr对应的记录，检查其owner是否为user\_id

4. 将step3中所有记录的friendsInGroup字段中的object\_id移动到default group对应的字段中（如果要删除的圈中有朋友，则在前端直接给出提示，删除后圈中朋友会移动到default中）

# 群

## Create

**url:** <http://domain/q/> POST

**输入参数：**

Values:{

recordInfo:{Name:{value:’aaaa’}},

method:0

}

**备注：**

**步骤：**

1. 检查是否有session（用户是否登录）

2. 用户是否有权限建立新群

1. 检查输入参数是否正确
2. 添加内部参数：CreatorId=userId, userId插入MembersId, userId插入adminId，如果输入没有addInRule，自动设置成allowanyone；自动往插入Histroy插入event= **createPublicGroup**的记录
3. 创建新的记录，并获得群的objectId

6. 在**usersPublicGroup**查找userId，是否存在。不存在，新建记录：userId=当前用户id，将step5中的群objectId插入字段currentJoinGroup；否则查找到存在的记录，直接将群objectId插入字段currentJoinGroup

7. 保存

## Search

分成2种：

1. 读取群的事件信息（创建时间，成员加入退出等） enum.ev =0(ev=event)

2. 读取群的动态 enum.ia =0(ia=interaction)

读取群基本资料：名称，人数，成员

**url:** <http://domain/q/>?method=1&type=**0**&gid=objectid(base64) GET

**输入参数：**

NA

**备注：**

1. 用户Id从session中获得

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确: type是否为0或者1，gid是否能转换成objectId

2. 从session中读取用户id

3. userId是否在gid这个群中

4. 读取**publicGroup->** Histroy字段，转换成可读格式返回

**url:** <http://domain/q/>?type=**1**&gid=objectid(base64) GET

**输入参数：**

NA

**备注：**

1. 用户Id从session中获得

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确: type是否为0或者1，gid是否能转换成objectId

2. 从session中读取用户id

3. userId是否在gid这个群中

4. 读取**publicGroupInteraction**的记录，转换成可读格式返回

## Update

**由于下列几种更新不太可能同时发生，所以使用单独的API执行**

种类：

1. 更新history（批准新成员加入，踢掉已有成员）

2. 更新history（成员自己加入，自主退出）

3. 更新history（管理员添加和删除，只能由创建者操作）

3. 更新history（管理员禁言成员，还要在memberStatus添加新记录）

4. 更新member加入的rule

**1. 更新history（**批准新成员加入（addInRule是permitAllow），踢掉已有成员（admin操作）**）**

**url:** [http://domain/q/](http://domain/wdtp/create)/admin/member POST

**输入参数：**

Values:{

recordInfo:{Name:{value:’aaaa’}},

method:2,

recordId: 群的objectId

updateSubField:{

MembersId {

From:

To:

eleArray

}

}

}

**备注：**

1. 用户Id从session中获得
2. 更新群中成员只有2种情况：1：直接移除（删除成员），2：群中加入一个成员

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. from/to，2者只能有一个存在

3. 从session中读取当前用户id

4.读取**publicGroup**记录，判断session中读取的userId是否有权进行操作（creatorId或者位于adminId中）

5. 如果to存在**且**from不存在，说明是add操作，对eleArray中的用户加入membersId

6. 反之，from存在且to不存在，说明是delete，从membersId，将eleArray中的数据移除

7. 处理**usersPublicGroup，**新建或者更新userid对应的记录

**2. 更新history（**批准新成员加入（addInRule是permitAllow），踢掉已有成员（admin操作）**）**

**url:** [http://domain/q/](http://domain/wdtp/create)member POST

**输入参数：**

Values:{

recordInfo:{Name:{value:’aaaa’}},

method:2,

recordId: 群的objectId

updateSubField:{

MembersId {

From:

To:

eleArray

}

}

}

**备注：**

1. 用户Id从session中获得
2. 更新群中成员只有2种情况：1：直接移除（删除成员），2：群中加入一个成员

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. from/to，2者只能有一个存在

3. 从session中读取当前用户id

4.读取**publicGroup**记录，判断session中读取的userId是否有权进行操作（creatorId或者位于adminId中）

5. 如果to存在**且**from不存在，说明是add操作，对eleArray中的用户加入membersId

6. 反之，from存在且to不存在，说明是delete，从membersId，将eleArray中的数据移除

7. 处理**usersPublicGroup，**新建或者更新userid对应的记录

**3. 更新history**（管理员添加和删除，只能由创建者操作）

**url:** [http://domain/q/](http://domain/wdtp/create)admin POST

**输入参数：**

Values:{

recordInfo:{Name:{value:’aaaa’}},

method:2,

recordId: 群的objectId

updateSubField:{

adminId {

From:

To:

eleArray:[objectId1….objectId N]

}

}

}

**备注：**

1. 用户Id从session中获得
2. 只有创建者才能更新管理员

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. from/to，2者只能有一个存在

3. 从session中读取当前用户id

4. 读取**publicGroup**记录，判断当前用户是否为creator

5. 如果to存在**且**from不存在，说明是add操作，将setp3中读取的userId加入adminId

6. 反之，from存在且to不存在，说明是delete，将setp3中读取的userId从adminId中移除

**4. 更新history（**addInRule**）**

**url:** [http://domain/q/](http://domain/wdtp/create)addInRule POST

**输入参数：**

Values:{

recordInfo:{Name:{value:’aaaa’}},

method:2,

recordId: 群的objectId

recordInfo:{

addInRule:{value: anyoneAllow|permitAllow }

}

}

**备注：**

1. 用户Id从session中获得

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. 从session中读取当前用户id

3. 读取**publicGroup**记录，判断session中读取的userId是否有权进行操作（creatorId或者位于adminId中）

4. 操作

## Delete

**url:** <http://domain/q/> POST

**输入参数：**

Values:{

recordInfo:{Name:{value:’aaaa’}},

method:3

recordId: object\_Id

}

**备注：**

1. 用户Id从session中获得

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. 从session中读取当前用户id

3. 读取**publicGroup**记录，检查其cratorId是否为user\_id

4. 将dDate设置

# 群事件（publicGroupEvent）

## Create

**url:** <http://domain/qsj/> POST

**输入参数：**

Values:{

recordInfo:{

publicGroupId:{value:’aaaa’},

eventType:{‘value’:’bbbb’},

targetId:{value:’asdf’} optional

status:{value:0}

},

method:0

}

**备注：**

1. sourceId从session中获得

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. 从session中读取当前用户id

3. 读取**js中**eventType的定义，根据事件的类型，决定当前用户是否可以操作（例如，某些事件只能有admin发起，则需检查当前用户是否有admin权限）

## Search

**url:** <http://domain/qsj/> POST

**输入参数：**

Values:{

recordInfo:{

publicGroupId:{value:’aaaa’},

eventType:{‘value’:’bbbb’},

targetId:{value:’asdf’} optional

status:{value:0}

},

method:1

}

**备注：**

1. sourceId从session中获得

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. 从session中读取当前用户id

3. 读取**js中**eventType的定义，根据事件的类型，决定当前用户是否可以读取事件（例如，某些事件只能有admin发起，则需检查当前用户是否有admin权限；某些事件普通用户也可以读取（例如，申请加入群，查看申请进度））

# 群成员处理（memberPenalize）

## Create

**url:** <http://domain/qcycl/> POST 群成员处理

**输入参数：**

Values:{

recordInfo:{

creatorId:{value:object\_id}

publicGroupId:{value:’aaaa’},

memberId: objectId

penalizeType:{‘value’:0},

duration: int

},

method:0

}

**备注：**

1. userId从session中获得

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. 从session中读取当前用户id

3. 读取**publicGroup**记录，检查user\_id是否有权限操作（creator或者adminId）

4. 插入新记录

## Search

**url:** [http://domain/ qcycl /](http://domain/qsj/) POST

**输入参数：**

Values:{

searchParams:{

creatorId:[{value:object\_id},{value:object\_id}]

publicGroupId:[{value:’aaaa’}], groupid只能是一个值，因为client一次只能处理一个group

penalizeType:[{‘value’:’bbbb’}], 过滤查看的处罚类型

memberId:[{value:’asdf’},{value:’asdf’}] 过滤用户

},

method:1

}

**备注：**

1. userId从session中获得

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. 从session中读取当前用户id

3. 根据用户的类型，读取对应的处理记录（adminId：读取所有处理记录； 普通用户：只读取自己相关的处理）

# 群交互（publicGroupInteraction）

## Create

**url:** <http://domain/qjh/> POST 群交互

**输入参数：**

Values:{

recordInfo:{

publicGroupId:{value:’aaaa’},

content: string 非html

},

method:0

}

**备注：**

1. userId从session中获得

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. 从session中读取当前用户id

3. 读取**publicGroup**记录，检查user\_id是否为群成员

4. 读取member Penalize ，当前用户是否被处罚中

5. 为出入参数添加creatorId=userId

6. 插入新记录

## Search

**url:** [http://domain/ qcycl /](http://domain/qsj/) POST

**输入参数：**

Values:{

searchParams:{

publicGroupId: [{value:object\_id}]

},

method:2

}

**备注：**

1. userId从session中获得

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. 从session中读取当前用户id

3. 检查userId是否位于publicGroupId的成员

4. 读取**publicGroupInteraction中的信息**

## Delete

**url:** [http://domain/ qjh /](http://domain/%20qjh%20/) POST

**输入参数：**

Values:{

recIdArr: 要删除的群交互信息的Id

method:3

}

**备注：**

1. userId从session中获得

**步骤：**

1. 检查是否有session（用户是否登录）

1. 检查输入参数是否正确

2. 从session中读取当前用户id

3. 检查userId是否可以删除交互信息（1. 是否为此信息的creator。 2是否为admin）

4. 如果是消息作者，更新dDate；如果是admin删除，更新deleteById和dDate字段；

# 用户加入的群(usersPublicGroup)

**不提供独立的API，而是被其他的API调用**

## Create

当用户注册成功，创建一个新的记录，其中currentJoinGroup为空数组

**备注：**

1. 无需任何client输入，userId从create user（注册成功后）获得

# Search

**url:** [http://domain/ yhq /](http://domain/%20yhq%20/) **POST** 用户群

**参数：**

Values:{

method:1

}

**备注**：

1. userId从session中获得

**步骤；**

1. 从session中读取userId

2. 从userPublicGroup中读取对应的记录

## Update

**2个单独的方法，对应加入/退出群。**

当admin允许加入，或者admin剔除；或者用户自主加入，自主退出。更新userPublicGroup的currentJoinGroup字段

# Delete

当删除用户是，同步删除对应的记录