# 统一认证服务接口文档

版本：V0.2.9

**修订记录**

| 时间 | 修订人 | 版本 | 修订内容 |
|---|---|---|---|
| 2015-4-14 | | V0.1 | 起稿 |
| 2015-4-16 | 顺炽国<br>叶霄霄<br>朱斌 | V0.2 | 1. 公共参数放置到 HTTP 自定义头中<br>2. 增加客户端版本号上传<br>3. 去掉 Email 相关注册、找回密码操作<br>4. 增加帐号在不同平台的登录说明<br>5. 增加子帐号解绑功能<br>6. 修改验证码发送及模板配置接口 |
| 2015-5-19 | 顺炽国 | V0.2.1 | 1.完善 HTTP 响应码描述 |
| 2015-7-7 | 顺炽国 | V0.2.2 | 1.修复 API 地址，统一返回结果 |
| 2015-7-22 | 叶霄霄 | V0.2.3 | 1.完善统一返回错误码 |
| 2015-7-23 | 叶霄霄 | V0.2.4 | 1.修复部分 API 参数错误 |
| 2015-7-23 | 叶霄霄 | V0.2.5 | 1.增加用户基本信息具体字段的数据校验 |
| 2015-7-24 | 叶霄霄 | V0.2.6 | 1.修复部分 API 参数错误 |
| 2015-7-27 | 朱斌 | V0.2.7 | 1.修复授权和短信部分 API 参数错误 |
| 2015-7-28 | 叶霄霄 | V0.2.8 | 1.增加验证码使用说明 |
| 2015-7-29 | 叶霄霄 | V0.2.9 | 1. 增加通过账号获取基本信息接口<br>2. 修改注册时的返回码用户编码不合法为公共的错误码账号不合法 |
| 2015-8-14 | 顺炽国 | V0.3 | 1.登录接口增加 Token 过期时间参数 |

# 目录

# 1. 约定

## 1.1. 请求地址定义

接口建议采用 HTTP 协议，基于 RESTFul 标准，请求路径格式统一采用如下格式：

**http(s)://host:port/api/version/resource**?param1=xxxx&param2=xxxx……

说明：

1. http(s)根据服务器部署情况来确认是使用 HTTP 还是 HTTPS 协议
2. host:port 指认证服务器的地址和服务端口号
3. api 是固定名称
4. version 是 API 的特定版本号，如:v1,v2 等，目前版本号默认为 V1
5. resource 表示操作哪个资源如：Token（认证授权码）、SyncUser（用户同步）等
6. 参数可以使用 URL 的方式提交，也可以使用 POST 的方式

## 1.2. 提交方法

| | |
|---|---|
| **PUT** | 更新数据 |
| **DELETE** | 删除数据 |
| **GET** | 查询数据 |
| **POST** | 新增或大量数据内容的更新、文件上传等 |

## 1.3. HTTP 自定义头信息

对于一些公共参数，如：客户端使用的协议版本号、应用 ID、访问授权码（AccessToken）等，可以统一放置在 HTTP 自定义头中。

约定的 HTTP 自定义头如下：

| HTTP 头 | 说明 |
|---|---|
| XRK-CLIENT-VERSION | 客户端使用接口的版本号信息 |
| XRK-APPID | 客户端的 AppID（注：此 AppID 是指应用的统一 ID，与应用版本、平台无关） |
| XRK-ACCESS-TOKEN | 当前用户登录后得到的 AccessToken，如果没有则不需要 |

目前约定的 appId 如下：

| 序号 | AppName | AppID | 应用名称 |
|---|---|---|---|
| 1 | substation | 1001 | 向日葵主站 |
| 2 | vip_dashboard | 1002 | 我的向日葵（移动版） |

| | | | |
|---|---|---|---|
| 3 | bxr | 1003 | 微站 |
| 4 | iask | 1004 | 问吧 |
| 5 | community | 1005 | 社区 |
| 6 | sso_web | 2001 | 统一认证站点 |
| 7 | xrk_admin | 2002 | 向日葵后台管理系统 |
| 8 | koubei | 2003 | 口碑 |
| 9 | bxr_admin | 2004 | 微站后台 |
| 10 | im_api | 2005 | 消息 API |
| 11 | xrk_api | 2006 | 微站 API |
| 12 | datahub | 2007 | 数据交换中心（Oracle） |
| 13 | account_api | 2008 | 统一接口 API |
| 14 | mysso | 2901 | 旧统一认证服务 |
| 15 | Weixin | 3001 | 微信 |

## 1.4. HTTP 请求 HEADER

**User-Agent：**

统一认证服务只提供给内部应用使用，因此不考虑 User-Agent 头的格式。此信息交由对外服务接口控制。

**Content-Type:**

对于 GET、DELETE 请求，URL 使用 UrlEncode 编码。

对于 POST、PUT 请求， Content-Type 需要设置为：application/x-www-form-urlencoded

对于上传文件的请求则不在此接口文档讨论范围内。

注：请求的参数内容编码默认使用 utf-8 编码。

## 1.5. 请求示例

```
GET /ACCOUNT/76885 HTTP/1.1
Connection: Keep-Alive
Host: api.xrk.com
```

```
POST /ACCOUNT HTTP/1.1
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Host: api.xrk.com

name=test&url=http%3A%2F%2www.xrk.com%2F13561234&mobile=132112233
44&.....
```

# 1.6. 返回值

服务端返回默认返回为 JSON 格式，响应结果复用 HTTP 的状态码，因此客户端使用时应先判断 HTTP 的状态码再作二次判断，一般分为如下两种情况：

**1) 执行成功**

此时返回内容默认为各个接口定义的返回对象，不作特殊说明

**2) 执行失败**

默认的失败格式如下：

{

code: int, ---状态码，0 默认为成功，其它见约定

message: string, --描述信息，如：异常时的一些描述

}

需要注意的是，HTTP 响应码失败时一般会经过两次判断，第一次判断 HTTP 状态码属于哪种错误类型，再解析返回的详细错误响应内容。

附常见的操作返回的响应码

| 方法 | 典型用法 | 返回状态码 |
|---|---|---|
| GET | - 获取表示<br>- 变更时获取表示（缓存） | 200（OK） - 表示已在响应中发出 |
| | | 204（无内容） - 资源有空表示 |
| | | 301（Moved Permanently） - 资源的 URI 已被更新 |
| | | 303（See Other） - 其他（如，负载均衡） |
| | | 304（not modified）- 资源未更改（缓存） |
| | | 400 （bad request）-指错误格式的请求 |
| | | 401 （unauthorized）-未授权的请求访问 |
| | | 403 （forbidden）-禁止访问指定的内容 |
| | | 404 （not found）- 资源不存在 |
| | | 405 （method not allowed） - 没有权限访问指定的方法 |
| | | 406 （not acceptable）- 服务端不支持请求的响应格式，如：请求的编码、语言、媒体格式、字符集等 |
| | | 415 （unsupported media type）-不支持请求的返回格式 |
| | | 422 （unprocessable entity）-不能处理的请求实体，一般用于检验参数失败的情况 |
| | | 500 （internal server error）- 通用错误响应 |
| | | 503 （Service Unavailable）- 服务端当前无法处理请求 |
| DELETE | - 删除资源 | 200 （OK）- 资源已被删除 |
| | | 301 （Moved Permanently）- 资源的 URI 已更改 |
| | | 303 （See Other）- 其他，如负载均衡 |
| | | 400 （bad request）-指错误格式的请求 |
| | | 401 （unauthorized）-未授权的请求访问 |
| | | 403 （forbidden）-禁止访问指定的内容 |
| | | 404 （not found）- 资源不存在 |
| | | 405 （method not allowed） - 没有权限访问指定的方法 |
| | | 409 （conflict）- 通用冲突 |

| | | |
|---|---|---|
| | | 500 （internal server error）- 通用错误响应 |
| | | 503 （Service Unavailable）- 服务端当前无法处理请求 |
| PUT | - 用客户端管理的实例号创建一个资源<br>- 通过替换的方式更新资源<br>- 如果未被修改，则更新资源（乐观锁） | 200 （OK）- 如果已存在资源被更改 |
| | | 201 （created）- 如果新资源被创建 |
| | | 301（Moved Permanently）- 资源的 URI 已更改 |
| | | 303 （See Other）- 其他（如，负载均衡） |
| | | 400 （bad request）-指错误格式的请求 |
| | | 401 （unauthorized）-未授权的请求访问 |
| | | 403 （forbidden）-禁止访问指定的内容 |
| | | 404 （not found）- 资源不存在 |
| | | 406 （not acceptable）- 服务端不支持所需表示/p> |
| | | 405 （method not allowed） - 没有权限访问指定的方法 |
| | | 409 （conflict）- 通用冲突 |
| | | 412 （Precondition Failed）- 前置条件失败（主要用于并发控制，如多人同时修改时，根据 HTTP 头(If-Unmodified-Since 或 If-Match)判断资源是否允许修改） |
| | | 415 （unsupported media type）- 接受到的表示不受支持 |
| | | 422 （unprocessable entity）-不能处理的请求实体，一般用于检验参数失败的情况 |
| | | 500 （internal server error）- 通用错误响应 |
| | | 503 （Service Unavailable）- 服务当前无法处理请求 |
| POST | - 使用服务端管理的（自动产生）的实例号创建资源<br>- 创建子资源<br>- 部分更新资源<br>- 如果没有被修改，则不过更新资源（乐观锁） | 200（OK）- 如果现有资源已被更改 |
| | | 201（created）- 如果新资源被创建 |
| | | 202（accepted）- 已接受处理请求但尚未完成（异步处理） |
| | | 301（Moved Permanently）- 资源的 URI 被更新 |
| | | 303（See Other）- 其他（如，负载均衡） |
| | | 400（bad request）- 指错误格式的请求 |
| | | 401 （unauthorized）-未授权的请求访问 |
| | | 403 （forbidden）-禁止访问指定的内容 |
| | | 404 （not found）- 资源不存在 |
| | | 405 （method not allowed） - 没有权限访问指定的方法 |
| | | 406 （not acceptable）- 服务端不支持所需表示 |
| | | 409 （conflict）- 通用冲突 |
| | | 412 （Precondition Failed）- 前置条件失败（主要用于并发控制，如多人同时修改时，根据 HTTP 头(If-Unmodified-Since 或 If-Match)判断资源是否允许修改） |
| | | 415 （unsupported media type）-不支持请求的返回格式 |
| | | 422 （unprocessable entity）-不能处理的请求实体，一般用于检验参数失败的情况 |
| | | 500 （internal server error）- 通用错误响应 |
| | | 503 （Service Unavailable）- 服务当前无法处理请求 |

## 1.7. JSONP 支持

在需要跨域访问的情况下，认证服务接口提供对 JSONP 的支持功能，需要在请求参数中指定 jsonp 参数，例如：
http://127.0.0.1/api/user/get?id=12&jsonp=callback_function_name
那么返回的内容格式如下：

```
callback_function_name({
        //用户信息对象
})
```

异常时则为：

```
callback_function_name({
    code: 101,
    message: “用户名必须输入”
})
```

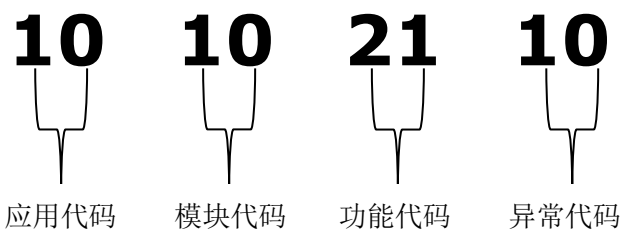
**注：jsonp 参数为保留参数，其它接口中不能使用此参数**

**说明：**

1. 如果无特殊说明，后须接口描述中默认只描述成功请求返回的内容，以及发生错误时的错误 code。
2. 每个接口都会标名请求的 resource，对应 URL 请求中的名称，调用者请自行组合。

## 1.8. 错误码约定

Code 一共 8 位，每两位为一组，其含义如下：




现有约定如下：
**应用：**
统一认证服务平台：10

**模块**
全局代码: 00
统一认证模块：01
用户管理：02

授权监听：03

**功能**
全局代码：00
其它功能同章节编号，如：2.1 对应 01，2.13 对应 13，以此类推。

公共异常代码列表（功能代码默认为 00）：

| | |
|---|---|
| 10000001 | accessToken 已过期 |
| 10000002 | accessToken 不存在 |
| 10000003 | AppID 不存在 |
| 10000004 | 时间戳无效 |
| 10000005 | 用户不存在 |
| 10000006 | 请求参数不正确 |
| 10000007 | 请求方法不正确 |
| 10000020 | 内部服务异常 |
| 10000021 | 验证码校验失败，不能继续当前操作 |
| 10000022 | 验证码校验结果已过期，不能继续当前操作 |
| 10000023 | uid 不合法 |
| 10000024 | 手机号码不合法 |
| 10000025 | 用户账号格式不合法 |

# 2. 用户接口

## 2.1. 验证用户参数

### 2.1.1. 功能说明

注册时，验证用户的关键参数是否已被使用。

### 2.1.2. 约定

| | |
|---|---|
| **Resource** | account/parameter |
| **METHOD** | GET |

### 2.1.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| mobile | String | Yes | 手机号码 |

### 2.1.4. 错误代码

### 2.1.5. 返回值

{
mobile：boolean，-- true-可用，false-不可用
}

## 2.2. 获取验证码

### 2.2.1. 功能说明

注册、找回密码、绑定手机时，调用此接口获取手机验证码。

### 2.2.2. 约定

| | |
|---|---|
| **Resource** | account/ captcha |
| **METHOD** | PUT |

### 2.2.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| mobile | String | Yes | 手机号 |
| checkType | Int | Yes | 检查方式，1-注册，2-找回密码，3-绑定， 4-登录 |

### 2.2.4. 错误代码

| | |
|---|---|
| 10020201 | 检查方式不合法 |

### 2.2.5. 返回值

{
timeout:int 验证码过期时间(秒)
}

## 2.3. 校验验证码

### 2.3.1. 功能说明

校验注册验证码的有效性。

进行注册、修改密码、重置密码、绑定手机、绑定邮箱这几个操作之前，需要传入不同的检查方式(checkType)校验验证码，校验成功后才可以正常调用这几个接口。否则会请求失败。

### 2.3.2. 约定

**Resource** account/ captcha
**METHOD** GET

### 2.3.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| mobile | String | Yes | 手机号码 |
| captcha | String | Yes | 用户产生的验证码 |
| checkType | int | Yes | 检查方式，1-注册，2-找回密码，3-绑定 |

### 2.3.4. 错误代码

10020301 验证码不合法
10020302 验证码已过期
10020303 检查方式不合法

### 2.3.5. 返回值

```
{
    result: Boolean,   --布尔值, true-成功，false-失败
}
```

## 2.4. 用户注册

### 2.4.1. 功能说明

新增用户信息到统一认证服务中心，并获得统一认证中心所分配的用户唯一 ID。

生成用户时，假如主账号时手机号，则自动绑定手机号；主账号为邮箱地址时同理。传入的用户基本信息中，自动过滤 emailIsVerify、mobileIsVerify 两个字段；假如主账号为手机号，自动过滤 mobile 字段；假如主账号为邮箱，自动过滤 email 字段。

### 2.4.2. 约定

**Resource** account/user
**METHOD** POST

### 2.4.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| mobile | String | Yes | 用户编码（一般用作对应 App 登录账号，目前默认为手机号） |
| password | String | Yes | 用户的密码，对原始密码经过 MD5 加密后的值 |
| userInfo | String | No | 用户的基础信息，使用 JSON 二次包装 |
| extendInfo | String | No | 用户的扩展信息，使用 JSON 二次包装 |
| unverified | String | NO | 注册时是否需要校验验证码。当且仅当该参数的值为 true 时，注册流程不需要校验验证码。 |

**注：**
userInfo 数据结构如下表：

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| sex | int | No | 性别 |
| userName | String | No | 用户名 |
| mobile | String | No | 手机号 |
| email | String | No | 电子邮箱 |
| qq | String | No | 用户 qq 号 |
| address | String | No | 地址 |
| postcode | String | No | 邮编 |

例：
{

```
    sex:1,
    userName:"张三",
    mobile:"13300000000",
    email:"xxx@xxx.com",
    qq:"1222222222",
    address:"广州市海珠区花城大道建滔广场",
    postcode:"510006"
}
```

再序列化成字符串

extendInfo 为用户的扩展信息，具体字段视用户场景而自定义，平台只负责存储，只需将相应键值对使用 JSON 包装传入即可。

如：

```
{
    IDCode: "1111111111",
    Site: "http://www.xxx.cc"
    ……
}
```

再序列化成字符串作为参数传入。

## 2.4.4. 错误代码

| | |
|---|---|
| 10020401 | 密码不合法 |
| 10020402 | 用户基础信息不合法 |
| 10020403 | 用户扩展信息不合法 |
| 10020404 | 性别参数不合法 |
| 10020405 | 手机号码不合法 |
| 10020406 | 电子邮箱不合法 |
| 10020407 | qq 号码不合法 |
| 10020408 | 邮政编码不合法 |
| 10020499 | 账号已存在 |

## 2.4.5. 返回值

```
{
    uid: long,          --生成的用户唯一 ID
    appId: int,         --所属应用
    userCode: String    --用户编码
}
```

## 2.5. 查询用户信息

### 2.5.1. 功能说明

查询用户的基本信息。

### 2.5.2. 约定

| | |
|---|---|
| **Resource** | account/user/{uid} |
| **METHOD** | GET |

### 2.5.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|

注：{uid}替换为实际的用户 ID。

### 2.5.4. 错误代码

### 2.5.5. 返回值

```
{
    userInfo:Object   用户基本信息
    extendInfo:Object   用户扩展信息
}
```

注：userInfo 和 extendInfo 的说明参照 2.4.3

返回示例：

```
{
    userInfo: {sex:1,userName:"张三",mobile:"13300000000"},
    extendInfo: {IDCode:11111,site:"www.aaa.cc"}
}
```

### 2.5.6. 示例

服务网站：http://www.test.com

用户 ID：10021
请求地址则为：
http://www.test.com/account/user/10021

## 2.6. 更新用户信息

### 2.6.1. 功能说明

更新用户的基本信息。用户的基本信息里，mobileIsVerify、emailIsVerify 两个字段不允许修改，即使传入了也会自动过滤。当用户的主账号为手机号时，mobile 字段不允许修改；当用户的主账号为邮箱账号时，email 字段不允许修改，即使传入了也会自动过滤。

### 2.6.2. 约定

| | |
|---|---|
| **Resource** | account/user/{uid} |
| **METHOD** | PUT |

### 2.6.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| userInfo | String | Yes | 用户的基础信息，使用 JSON 二次包装 |
| extendInfo | String | No | 用户的扩展信息，使用 JSON 二次包装 |

注：userInfo 和 extendInfo 的说明参照 2.4.3

### 2.6.4. 错误代码

| | |
|---|---|
| 10020601 | 用户基础信息不合法 |
| 10020602 | 用户扩展信息不合法 |
| 10020604 | 性别参数不合法 |
| 10020605 | 手机号码不合法 |
| 10020606 | 电子邮箱不合法 |
| 10020607 | qq 号码不合法 |
| 10020608 | 邮政编码不合法 |

### 2.6.5. 返回值

```
{
    result: Boolean,    --布尔值, true-成功，false-失败
```

}

## 2.7. 修改用户密码

### 2.7.1. 功能说明

更新用户密码。

### 2.7.2. 约定

**Resource** account/password/{uid}
**METHOD** PUT

### 2.7.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| oldpwd | String | Yes | 原密码，对原始密码经过 MD5 加密后的值 |
| password | String | Yes | 新密码，对原始密码经过 MD5 加密后的值 |
| unverified | String | NO | 修改用户密码时，是否需要校验验证码。当且仅当该参数的值为 true 时，修改用户密码的流程不需要校验验证码。 |

### 2.7.4. 错误代码

10020701 密码格式不合法
10020702 原始密码错误

### 2.7.5. 返回值

```
{
    result: Boolean,   --布尔值, true-成功，false-失败
}
```

## 2.8. 重新设置密码

### 2.8.1. 功能说明

使用找回密码功能的最后一步，重新设置用户密码。

### 2.8.2. 约定

**Resource** account/password/{uid}
**METHOD** POST

### 2.8.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
| --- | --- | --- | --- |
| password | String | Yes | 新密码，对原始密码经过 MD5 加密后的值 |
| unverified | String | NO | 重置密码时，是否需要校验验证码。当且仅当该参数的值为 true 时，重置密码的流程不需要校验验证码。 |

### 2.8.4. 错误代码

10020801 密码不合法

### 2.8.5. 返回值

```
{
    result: Boolean,  --布尔值, true-成功，false-失败
}
```

## 2.9. 删除用户（该版本未实现）

### 2.9.1. 功能说明

删除指定用户。

### 2.9.2. 约定

**Resource** account/user/{uid}
**METHOD** DELETE

### 2.9.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
| --- | --- | --- | --- |

### 2.9.4. 错误代码

10020901　　没有操作权限

### 2.9.5. 返回值

{
　　result: Boolean,　--布尔值, true-成功，false-失败
}

*注：作为安全考虑，删除用户需要考虑是否限制为只允许添加者删除。*

## 2.10. 查询手机绑定状态

### 2.10.1. 功能说明

查询账号是否已绑定手机。

### 2.10.2. 约定

**Resource**　　account/bind/{uid}
**METHOD**　　GET

### 2.10.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
| --- | --- | --- | --- |

### 2.10.4. 错误代码

### 2.10.5. 返回值

{
    mobile:  boolean，--true-已绑定，false-未绑定
}

## 2.11. 绑定手机

### 2.11.1. 功能说明

账号绑定手机。用户注册时，假如主账号为手机号，则自动绑定手机。假如用户已经绑定了手机 A，允许用户重新绑定手机 B 以替代手机 A；在这种场景下，假如用户的主账号也为手机号 A，则重新绑定手机 B 之后，用户的主账号也更换为手机 B。

### 2.11.2. 约定

**Resource** account/bind/{uid}
**METHOD** PUT

### 2.11.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| mobile | String | Yes | 假如是绑定手机，必须提供该字段 |
| unverified | String | NO | 绑定手机时，是否需要校验验证码。当且仅当该参数的值为 true 时，绑定手机的流程不需要校验验证码。 |

### 2.11.4. 错误代码

10021102 手机号已被他人绑定
10021103 该用户已绑定手机

### 2.11.5. 返回值

{

result: Boolean,　--布尔值, true-成功，false-失败
}

## 2.12. 解绑手机

### 2.12.1. 功能说明

将账号与手机号解绑。假如用户的主账号为手机号，则不允许解绑手机号。

### 2.12.2. 约定

**Resource**　account/bind/{uid}
**METHOD**　DELETE

### 2.12.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
| --- | --- | --- | --- |
| unverified | String | NO | 解绑手机时，是否需要校验验证码。当且仅当该参数的值为 true 时，解绑手机的流程不需要校验验证码。 |

### 2.12.4. 错误代码

10021200　该用户未绑定手机
10021201　不允许解绑手机（用户的主账号是手机）

### 2.12.5. 返回值

{
result: Boolean,　--布尔值, true-成功，false-失败
}

## 2.13. 绑定应用账号

### 2.13.1. 功能说明

将已注册的用户绑定到未登录过的 App 上。比如用户正在使用公众号，此时若绑定用

户的向日葵账号，则公众号可以获取向日葵账号的头像、昵称等信息用在公众号上。

### 2.13.2. 约定

**Resource** account/subAccount/{uid}
**METHOD** POST

### 2.13.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| tempId | String | Yes | 用户使用 APP 时生成的临时账号 |
| subAppId | int | Yes | 要绑定的子应用 ID |

### 2.13.4. 错误代码

10021301 tempID 不合法
10021302 该子账号已被绑定
10021303 子应用 ID 不合法

### 2.13.5. 返回值

```
{
    result: Boolean,  --布尔值, true-成功，false-失败
}
```

## 2.14. 查询绑定账号

### 2.14.1. 功能说明

查询当前账号已关联的账号（包含第三方系统账号），需要注意的是，对于内部应用，返回的是与用户相关的临时 ID。

### 2.14.2. 约定

**Resource** account/subAccount/{uid}
**METHOD** GET

### 2.14.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|

### 2.14.4. 错误代码

### 2.14.5. 返回值

[uidObj1, uidObj2, ……,uidObjn]
每个对象结构如下:
{
uid:String, --用户 ID
bindAppId:int, --绑定账号所在的应用 ID
bindAppName:String, --绑定账号所在的应用名称
account: String, --账号名称
appId:int, --账号对应的应用 ID
appName:String --账号对应的应用名称
thridParty:Boolean --是否第三方应用，如果是则为 true
}

## 2.15. 查询指定子账号

### 2.15.1. 功能说明

查询指定子账号在某个应用下的信息。

### 2.15.2. 约定

**Resource** account/subAccount
**METHOD** GET

### 2.15.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| **subaccount** | **String** | **Yes** | **子账号名称** |
| **subappid** | **Int** | **Yes** | **子账号所在的应用 ID** |

### 2.15.4. 错误代码

10000005 子账号不存在

### 2.15.5. 返回值

```
{
    uid:String,              --用户 ID
    bindAppId:int,           --绑定账号所在的应用 ID
    bindAppName:String,      --绑定账号所在的应用名称
    account: String,         --账号名称
    appId:int,               --账号对应的应用 ID
    appName:String           --账号对应的应用名称
    thridParty:Boolean       --是否第三方应用，如果是则为 true
}
```

## 2.16. 子账号解除绑定

### 2.16.1. 功能说明

将对应 app 与已注册的账号解除绑定。

### 2.16.2. 约定

**Resource** account/subAccount/{uid}
**METHOD** DELETE

### 2.16.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| subAppId | int | Yes | 要解除绑定的应用 ID |

### 2.16.4. 错误代码

10021501 子应用 ID 不合法

10021502　　子账号不存在

### 2.16.5. 返回值

```
{
    result: Boolean,  --布尔值, true-成功，false-失败
}
```

## 2.17. 禁用账号（管理员接口）

### 2.17.1. 功能说明

封禁指定账号。这里的 uid 是指管理员账号的 uid，而 targetId 指的是要操作的目标账号的 uid。

### 2.17.2. 约定

**Resource**　　account/status/{uid}
**METHOD**　　PUT

### 2.17.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| targetId | long | Yes | 目标用户的 UID |

### 2.17.4. 错误代码

10021601　　用户已被禁用
10021699　　没有操作权限

### 2.17.5. 返回值

```
{
    result: Boolean,  --布尔值, true-成功，false-失败
}
```

## 2.18. 账号解禁（管理员接口）

### 2.18.1. 功能说明

将指定账号解封。这里的 uid 是指管理员账号的 uid，而 targetId 指的是要操作的目标账号的 uid。

### 2.18.2. 约定

**Resource** account/status/{uid}
**METHOD** DELETE

### 2.18.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
| --- | --- | --- | --- |
| targetId | long | Yes | 目标用户的 UID |

### 2.18.4. 错误代码

10021701 该用户未被禁用
10021799 没有操作权限

### 2.18.5. 返回值

```
{
    result: Boolean,  --布尔值, true-成功，false-失败
}
```

## 2.19. 查看用户状态（管理员接口）

### 2.19.1. 功能说明

查看用户是否被禁用。这里的 uid 是指管理员账号的 uid，而 targetId 指的是要操作的目标账号的 uid。

### 2.19.2. 约定

**Resource** account/status/{uid}
**METHOD** GET

### 2.19.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
| --- | --- | --- | --- |
| targetId | long | Yes | 目标用户的 UID |

### 2.19.4. 错误代码

10021899 没有操作权限

### 2.19.5. 返回值

```
{
    result: Boolean,  --布尔值, true-账号可用（未被禁用），false-账号不可用（被禁用）
}
```

## 2.20. 通过账号查询用户信息

### 2.20.1. 功能说明

通过用户账号，查询用户的基本信息。

### 2.20.2. 约定

**Resource** account/user
**METHOD** GET

### 2.20.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
| --- | --- | --- | --- |
| account | String | Yes | 要查询的用户的账号 |

### 2.20.4. 错误代码

### 2.20.5. 返回值

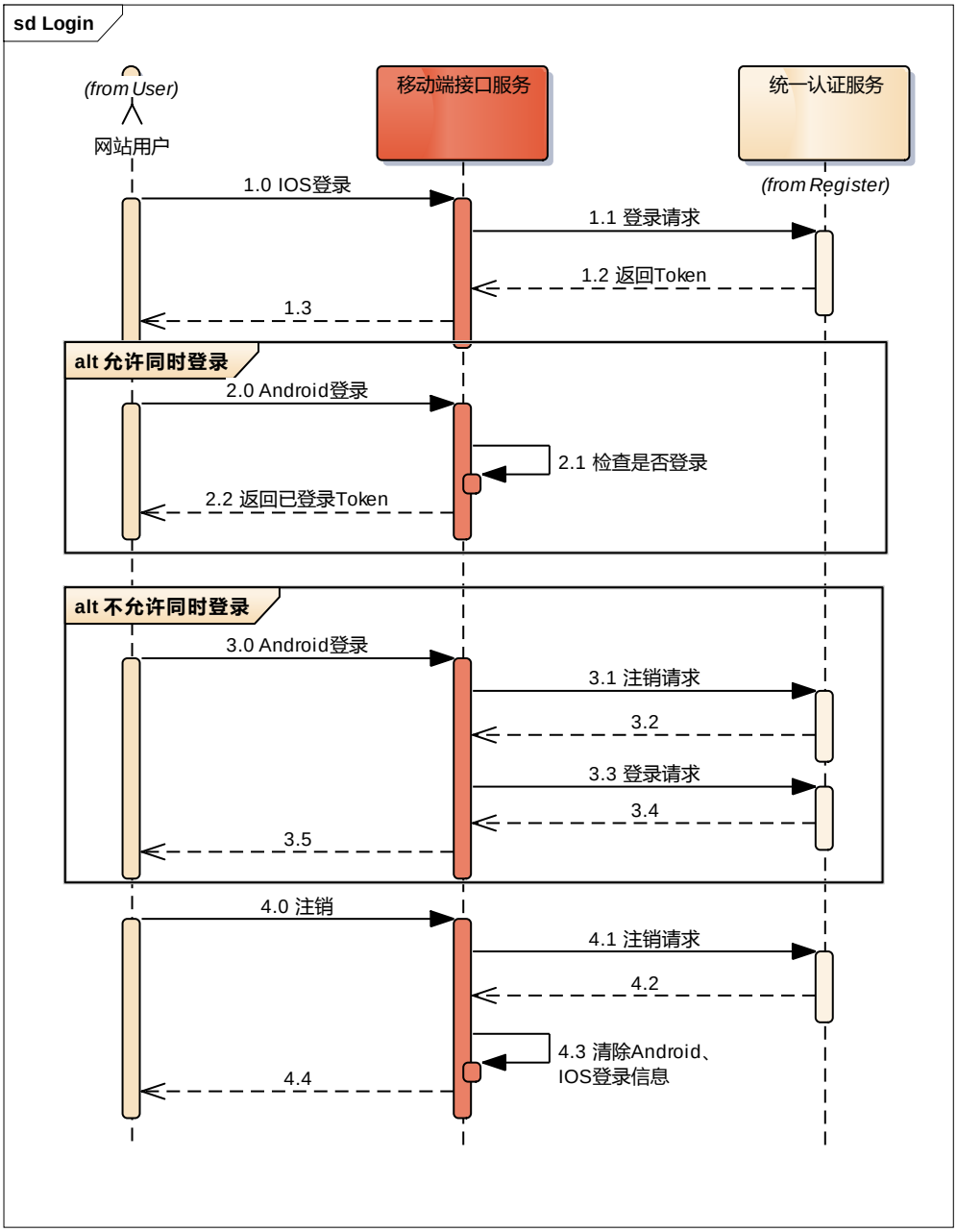
{
    userInfo:Object  用户基本信息
    extendInfo:Object  用户扩展信息
}

注：userInfo 和 extendInfo 的说明参照 2.4.3

返回示例：
{
    userInfo: {sex:1,userName:"张三",mobile:"13300000000"},
    extendInfo: {IDCode:11111,site:"www.aaa.cc"}
}

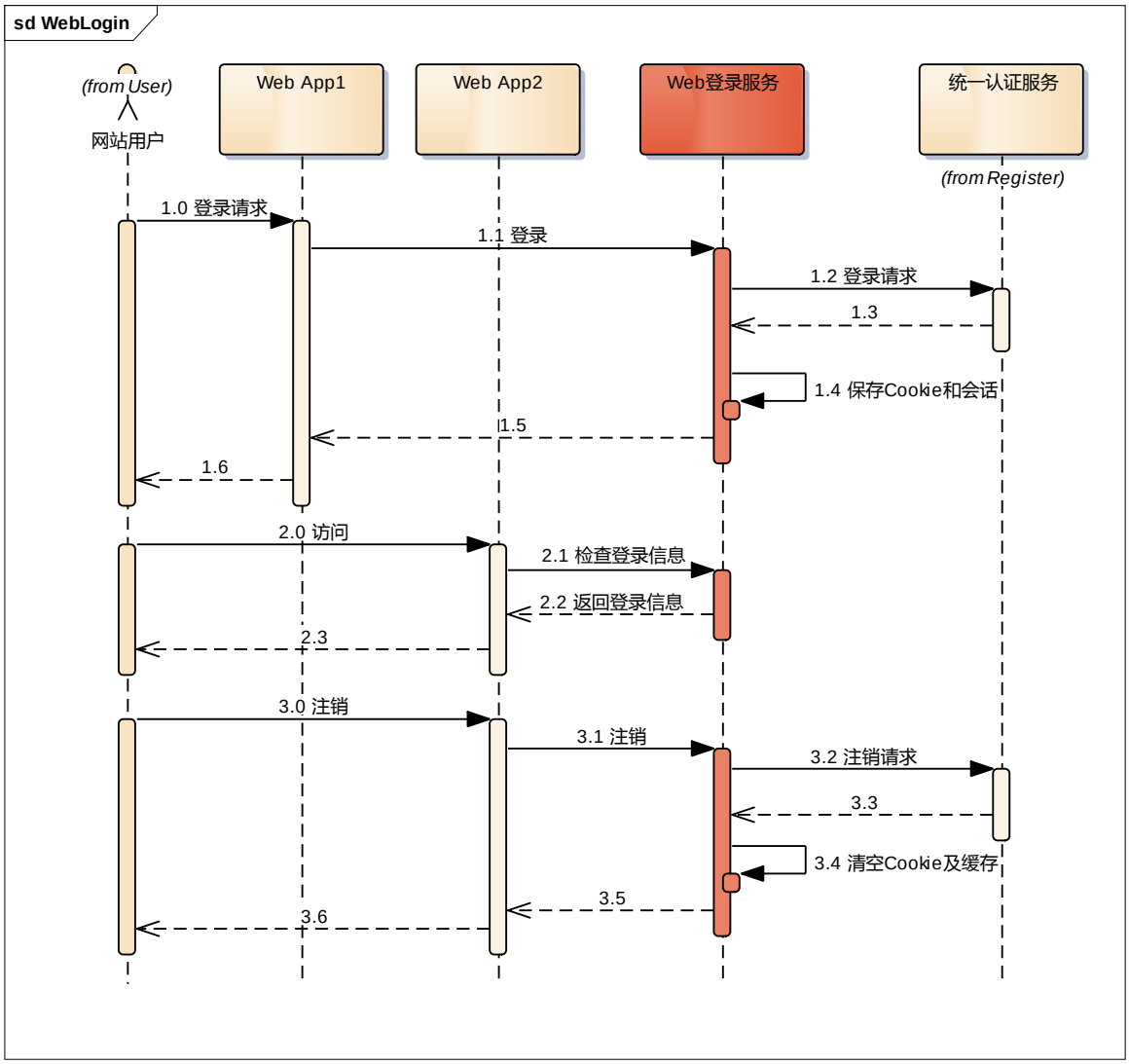
# 3. 认证接口

移动端认证流程：




对于 App 端来说，用户是否共享，由

WEB 端登录流程




对于 Web 来说，其实所有登录操作都是通过 WEB 登录服务进行的，如果某个 Web 应用不需要统一登录，则可以直接与统一认证服务对接。

# 3.1. 认证接口

## 3.1.1. 功能说明

提供登录认证的操作，同时取得接口访问授权 Key。同一个 AppID 只能允许存在一个登录会话。

## 3.1.2. 约定

| | |
|---|---|
| **Resource** | AccessToken |
| **METHOD** | POST |

## 3.1.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| Account | String | Yes | 用户账号，目前系统约定为手机号码 |
| password | String | Yes | 经过混淆（salt）加密后的密码，加密规则见下面的说明 |
| timestamp | Int | Yes | 当前的时间戳，基于 Unix 时间戳 |
| Scope | String | No | 请求授权的范围，保留参数 |
| expireTime | Int | No | 过期时间，如果不使用，则使用系统默认过期时间，时间单位为分钟 |

**参数说明**

**password:**

在数据库中存储的密码建议采用如下规则：

Md5(Md5(password)+account)

先对用密码进行 md5 运算得到的值与用户 ID 相加，再进行 MD5 计算，得到最终存储的密码。

对于上传的用户密码，是经过 ticket 混淆后的密码，其算法在原来数据库的基础上增加了一次 md5 运算，如下:

Md5(Md5(Md5(password)+account)+timestamp)

**timestamp:**

使用 Unix 的时间戳，从 1970-1-1 到现在的毫秒数，其作用主要是用于混淆及超时检测

## 3.1.4. 错误代码

10010101 密码错误

## 3.1.5. 返回值

```
{
    uid: string,            --对应用户 ID
    appId:string            --对应的 AppID
    accessToken: string,    --访问 Token
    refreshToken: String,   --获取新访问 Token 时使用，使用一次后失效，过期后不能使用
    expireTime:datetime     --过期时间
}
```

### 3.1.6. 示例

UID： d16896080d144661b9150f2f705d177f
Password: 123456
Timestamp: 1427422210 (2015-3-27 10:10:10)

加密步骤如下：

1. 对原始密码进行 md5 运算：

md5(password) = e10adc3949ba59abbe56e057f20f883e

2. 将运算后的 md5 值与用户 ID 相加，进行第二次 md5 运算（此值也是存在数据库中的密码值）

md5(md5(password)+uid)
=md5(e10adc3949ba59abbe56e057f20f883ed16896080d144661b9150f2f705d177f)
= 924e6760ff0931dcb25a3aa8f072e42e

3. 对第二步计算后的值与当前时间戳相加进行第三次 md5 运算，得到最终的加密密码

Md5(Md5(Md5(password)+uid)+timestamp)
=md5(924e6760ff0931dcb25a3aa8f072e42e1427422210)
=4bf046e8c02d5098a72400baa633b50b

## 3.2. 使用 UID 登录

### 3.2.1. 功能说明

使用分配的 UID 登录系统。

### 3.2.2. 约定

**Resource** AccessToken/{uid}
**METHOD** POST

### 3.2.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| password | String | Yes | 仅对密码作 md5 运算 |
| timestamp | Int | Yes | 当前的时间戳，基于 Unix 时间戳 |
| Scope | String | No | 请求授权的范围，保留参数 |
| expireTime | Int | No | 过期时间，如果不使用，则使用系统默认过期时 |

间，时间单位为分钟

### 3.2.4. 错误代码

### 3.2.5. 返回值

```
{
    uid: string,              --对应用户 ID
    appId:string              --对应的 AppID
    accessToken: string,      --访问 Token
    refreshToken: String,     --获取新访问 Token 时使用，使用一次后失效，过期后不能使用
    expireTime:datetime       --过期时间
}
```

## 3.3. 使用验证码登录

### 3.3.1. 功能说明

通过手机收到的验证码进行登录认证，同时取得接口访问授权 Key。同一个 AppID 只能允许存在一个登录会话，如果用户通过用户名和密码登录后，再通过验证码登录上次的授权码不会发生变化。

### 3.3.2. 约定

| | |
|---|---|
| **Resource** | AccessToken/captcha |
| **METHOD** | POST |

### 3.3.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| Mobile | String | Yes | 登录的手机号 |
| Captcha | String | Yes | 手机接收到的验证码 |
| Scope | String | No | 请求授权的范围，保留参数 |
| expireTime | Int | No | 过期时间，如果不使用，则使用系统默认过期时间，时间单位为分钟 |

**参数说明**

**Captcha:**

可以通过获取验证码接口中的 checkType=4 获取到。

## 3.3.4. 错误代码

| | |
|---|---|
| 10020301 | 无效的验证码 |

## 3.3.5. 返回值

```
{
    uid: string,              --对应用户 ID
    appId:string              --对应的 AppID
    accessToken: string,      --访问 Token
    refreshToken: String,     --获取新访问 Token 时使用，使用一次后失效，过期后不能使用
    expireTime:datetime       --过期时间
}
```

# 3.4. 使用子账号免密登录

## 3.4.1. 功能说明

如果用户绑定了某个应用下的子账号，那么可以使用子账号及关联的应用 ID 免密登录，获取到当前注册用户 ID 及登录的授权信息。

## 3.4.2. 约定

| | |
|---|---|
| **Resource** | AccessToken/subaccount |
| **METHOD** | POST |

## 3.4.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| subaccount | String | Yes | 子账号 |
| subAppId | String | Yes | 子账号所属的 AppID |

| | | | |
|---|---|---|---|
| Scope | String | No | 请求授权的范围，保留参数 |
| expireTime | Int | No | 过期时间，如果不使用，则使用系统默认过期时间，时间单位为分钟 |

### 3.4.4. 错误代码

| | |
|---|---|
| 10020301 | 无效的验证码 |

### 3.4.5. 返回值

```
{
    uid: string,            --对应用户 ID
    appId:string            --对应的 AppID
    accessToken: string,    --访问 Token
    refreshToken: String,   --获取新访问 Token 时使用，使用一次后失效，过期后不能使用
    expireTime:datetime     --过期时间
}
```

## 3.5. 更新授权码

### 3.5.1. 功能说明

在授权码即将超时或需要重新获取授权码时，可以调用此接口重新获取得到新的授权码。

### 3.5.2. 约定

| | |
|---|---|
| **Resource** | AccessToken |
| **METHOD** | PUT |

### 3.5.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| uid | String | Yes | 当前登录用户的 ID |
| refreshToken | String | Yes | 要获得的 Token |

### 3.5.4. 错误代码

| | |
|---|---|
| 10010201 | refreshToken 已过期 |
| 10010202 | refreshToken 不正确 |

### 3.5.5. 返回值

```
{
    uid: string,              --对应用户 ID
    AppID: string             --应用 ID
    accessToken: string,      --访问 Token
    refreshToken: String,     --获取新访问 Token 时使用，使用一次后失效
    expireTime:datetime       --过期时间，使用 unix timestamp 格式
}
```

## 3.6. 用户登录信息查询

### 3.6.1. 功能说明

查询指定用户已登录的信息。

### 3.6.2. 约定

| | |
|---|---|
| **Resource** | AccessToken/{uid} |
| **METHOD** | GET |

### 3.6.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|

注：{uid}替换为实际的用户 ID

### 3.6.4. 错误代码

| | |
|---|---|
| 10010301 | 用户未登录 |

### 3.6.5. 返回值

[obj1, obj2……]
每个 Obj 的对象如下：
{
uid: string, --对应用户 ID
AppID: string --应用 ID
accessToken: string, --访问 Token
refreshToken: String, --获取新访问 Token 时使用，使用一次后失效
expireTime:datetime --过期时间，使用 unix timestamp 格式
}

### 3.6.6. 示例

服务网站：http://www.test.com
用户 ID：10021
请求地址则为：
http://www.test.com/accesstoken/10021

## 3.7. 授权查询

### 3.7.1. 功能说明

验证授权码是否可用。

### 3.7.2. 约定

| | |
|---|---|
| **Resource** | AccessToken |
| **METHOD** | GET |

### 3.7.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|

### 3.7.4. 错误代码

### 3.7.5. 返回值

```
{
    uid: string,              --对应用户 ID
    AppID: string             --应用 ID
    accessToken: string,      --访问 Token
    refreshToken: String,     --获取新访问 Token 时使用，使用一次后失效
    expireTime:datetime       --过期时间，使用 unix timestamp 格式
}
```

## 3.8. 注销授权

### 3.8.1. 功能说明

注销已经授权的请求。

### 3.8.2. 约定

| | |
|---|---|
| **Resource** | AccessToken |
| **METHOD** | DELETE |

### 3.8.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|

### 3.8.4. 错误代码

### 3.8.5. 返回值

```
{
    result: Boolean,   --布尔值, true-成功，false-失败
}
```

# 4. 授权信息监听接口

## 4.1. 用户授权推送注册

### 4.1.1. 功能说明

在其它内部应用需要获取当前已授权用户的授权信息时，可以通过注册到推送接口，当有新用户授权成功时，将同步已授权的授权码信息到观察者提供的接口上。

注：

1. 推送的回调接口必须为有效的 Http 地址，且能正确接收 POST 请求
2. 推送的信息内容详见 3.1.9 章节定义的推送内容
3. 如果在不同网络环境，需要统一认证服务与观察者之间网段的访问问题

### 4.1.2. 约定

| | |
|---|---|
| **Resource** | Authorize/push |
| **METHOD** | POST |

### 4.1.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| callBackUrl | String | Yes | 回调 Url |

### 4.1.4. 返回值

正常时的返回 JSON 数据包示例：

```
{
    appID: string,          --对应申请应用 ID
    status: string,         -申请注册推送授权的状态,0 表示成功,1 表示失败。
    regTime:datetime        --注册时间
}
```

### 4.1.5. 错误码

| | |
|---|---|
| 10000003 | appID 不存在 |
| 10030101 | 授权注册推送 url 地址不合法 |
| 10030102 | 此应用观察者已存在 |

## 4.2. 用户授权推送解除

在其它内部应用不需要当前已授权用户的授权信息时，可以通过解除到推送接口，解除成功后已授权的授权码信息将不会推送到观察者提供的接口上。

### 4.2.1. 约定

| | |
|---|---|
| **Resource** | Authorize/push |
| **METHOD** | DELETE |

### 4.2.2. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|

### 4.2.3. 返回值

正常时的返回 JSON 数据包示例：

{
appID: string, --对应申请应用 ID
status: string, --解除注册推送授权的状态,0 表示成功,1 表示失败。
delTime:datetime --解除时间
}

### 4.2.4. 错误码

| | |
|---|---|
| 10000003 | appID 不存在 |
| 10030103 | 观察者不存在 |

## 4.3. 用户授权信息推送

因考虑到性能问题，统一认证服务推送不会采用实时推送的方式，而会根据不同的策略在缓冲区满或到指定推送时间后主动进行推送。

### 4.3.1. 推送内容

以数组的格式推送到应用申请注册的回调 url 地址上
[authObj1, authObj2,……]
其中每个对象格式如下：
{
pushType: String, --消息类型，目前支持如下三种消息：
Authroize_login，Authorize_Update，Authroize_Logout
message: --消息内容
{
uid: string, --对应用户 ID
appId: String, --用户所属的应用 ID
accessToken: string, --访问 Token
refreshToken:string, --刷新 Token
expireDate:datetime --过期时间，使用 unix timestamp 格式
}

注：
1. 观察者应用在进行授权码判断的时候，需要判断过期时间
2. 如果观察者应用未找到对应的授权码，可以调用 3.1.4 接口进行授权码验证查询，将获取到的结果放到本地应用的缓存中

## 4.4. 短信模板设置接口

### 4.4.1. 功能说明

不同的 APP 可针对用户注册、找回密码、等场景来设置发送短信的模板及短信模板的状态，如启用或禁用，来通过调用短信网关发送短信。

### 4.4.2. 约定

**Resource** msg/template
**METHOD** POST

### 4.4.3. 接口定义

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| appId | String | Yes | 所属应用 |
| msgInfo | String | Yes | 基础信息，使用 JSON 二次包装 |

**注：**
msgInfo 数据结构如下表：

| 参数名 | 类型 | 必须提供 | 说明 |
|---|---|---|---|
| templateCode | String | Yes | 模板业务 code(01:注册,02:找回密码.....) |
| content | String | Yes | 内容 |
| state | String | No | 状态 0:启用,1:禁用,默认为启用, |

例：

```
{
    templateCode:01,
    content:"【向日葵】您在向日葵网的验证码:{code}",
    state:"0",

}
```

再序列化成字符串

## 4.4.4. 错误代码

| | |
|---|---|
| 10000003 | appID 不存在 |
| 10020500 | 模板信息不合法 |

## 4.4.5. 返回值

```
{
    result: Boolean,    --布尔值, true-成功，false-失败
}
```