统一认证服务设计文档

版本：V0.1

**修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **时间** | **修订人** | **版本** | **修订内容** |
| 2015-3-25 | 顺炽国 | V0.1 | 初稿 |
| 2015-4-2 | 顺炽国 | V0.2 | 1. 增加账户管理服务相关需求 2. 统一管理用户资料信息 3. 增加内部数据流 4. 修改接口请求、响应格式，增加注册、更新、绑定等接口 |
| 2015-4-3 | 顺炽国 | V0.2.1 | 1.增加关联用户接口及数据流说明 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1. 概要 5](#_Toc415846471)

[1.1. 背景 5](#_Toc415846472)

[1.2. 目标 5](#_Toc415846473)

[2. 系统框架 6](#_Toc415846474)

[2.1. 系统关系图 6](#_Toc415846475)

[2.2. 系统结构图 7](#_Toc415846476)

[2.2.1. 统一认证服务 7](#_Toc415846477)

[2.2.2. 账户管理服务 8](#_Toc415846478)

[2.3. 应用数据协议流 8](#_Toc415846479)

[2.3.1. 传输协议及编码格式 8](#_Toc415846480)

[2.3.2. 单服务器情况 9](#_Toc415846481)

[2.3.3. 多服务器情况 10](#_Toc415846482)

[2.4. 内部数据流 12](#_Toc415846483)

[2.4.1. 注册及更新 12](#_Toc415846484)

[2.4.2. 登录授权 13](#_Toc415846485)

[2.4.3. 账号关联、查询 13](#_Toc415846486)

[2.4.4. 监听推送 14](#_Toc415846487)

[2.5. 用户体系 15](#_Toc415846488)

[3. 系统方案 15](#_Toc415846489)

[3.1. 业务接口 17](#_Toc415846490)

[3.1.1. 约定 17](#_Toc415846491)

[3.1.2. 验证用户参数 19](#_Toc415846492)

[3.1.3. 用户注册 21](#_Toc415846493)

[3.1.4. 更新用户信息 22](#_Toc415846494)

[3.1.5. 更新用户密码 24](#_Toc415846495)

[3.1.6. 删除用户 25](#_Toc415846496)

[3.1.7. 绑定应用账号 26](#_Toc415846497)

[3.1.8. 关联已有账号 27](#_Toc415846498)

[3.1.9. 查询关联账号 28](#_Toc415846499)

[3.1.10. 认证接口 29](#_Toc415846500)

[3.1.11. 更新授权码 30](#_Toc415846501)

[3.1.12. 授权查询 31](#_Toc415846502)

[3.1.13. 注销授权 32](#_Toc415846503)

[3.1.14. 用户授权推送注册 33](#_Toc415846504)

[3.1.15. 用户授权信息推送 34](#_Toc415846505)

[3.2. 扩展接口 35](#_Toc415846506)

[3.2.1. 文件访问接口 35](#_Toc415846507)

[3.2.2. 数据访问接口 35](#_Toc415846508)

[3.2.3. 缓存接口 35](#_Toc415846509)

[3.2.4. 集群服务接口 36](#_Toc415846510)

[3.3. 系统初始化 36](#_Toc415846511)

[3.4. 登录认证 36](#_Toc415846512)

[3.5. 授权信息管理 39](#_Toc415846513)

[3.6. 同步用户数据 40](#_Toc415846514)

[3.7. 性能统计 42](#_Toc415846515)

[3.8. 相关开源项目 42](#_Toc415846516)

[4. 实施计划 43](#_Toc415846517)

[5. 风险评估 43](#_Toc415846518)

[6. 性能指标 43](#_Toc415846519)

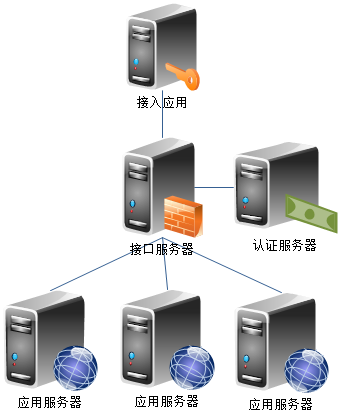
# 概要

## 背景

提供一个统一认证的服务中心，需要进行登录认证的用户统一在认证服务中心获取授权认证，根据获取到的授权Token进行业务接口调用操作。

## 目标

实现用户账号的统一分配、登录信息的统一管理，提供认证服务接口，授权Token的查询、验证功能，同时提供用户授权信息的监听功能，主动推送用户授权信息到监听者，以提升检测效率。提供的所有接口服务都会通过接口服务器进行对外发布，即各后台应用所提供的服务接口，都是接受来自接口服务器的请求，而不直接提供外部接入的途径，同时，接口服务器会负责访问授权的处理，提供服务的应用服务器只需要对接口服务器的身份进行识别和认证即可，简要结构如下：



# 系统框架

## 系统关系图

说明：

1. 后端服务与前端应用系统之间统一走内部协议
2. 前端应用中，涉及到两个接口，对第三方平台的协议建议采用OAuth2.0协议，对于手机端的App接口服务，则由手机端自定
3. 统一认证服务主要有三种消息，认证消息、用户同步消息、授权信息推送相关消息
4. 账户管理服务则主要提供用户注册、登录认证接口

## 系统结构图

### 统一认证服务



说明：

1. 接口分为两类，一类是与业务相关的接口，另一类是属于服务自身的扩展接口
2. 缓存服务提供统一的接口，可以实现本地缓存或外部缓存，为了后续扩展考虑，缓存的内容都必须可序列化
3. 数据访问层也遵循统一的接口，可以实现多数据库存储，如：MySql，PostgreSQL、Oracle等
4. 集群扩展主要是提供给后续认证服务横向扩展时预留的通讯接口，用于服务间的消息通知、数据同步等。

### 账户管理服务



说明：

1. 账户管理服务提供了账户关系的管理，如：账号与应用、第三方系统注册、绑定的帐号等。
2. 注册、修改用户密码成功的，需要将信息同步到统一认证服务中
3. 用户账户只存放基本信息，此信息在所有应用中共享，不同应用的特有信息，由各自维护。
4. 用户基本信息变更可以通过业务接口进行变更，如：修改密码等

## 应用数据协议流

### 传输协议及编码格式

因为统一认证服务运行于内网，对数据包大小的优先级较低，可以考虑选择的方案有两种：

|  |  |  |
| --- | --- | --- |
| **方案** | **优点** | **缺点** |
| Thrift框架 | 1. 集成了编码与传输层 2. 数据已压缩，节省网络带宽 3. 采用TCP长连接，性能较高 4. 支持多语言平台 | 1. 必须预先定义协议文件，再生成对应的代码，涉及到协议变更或扩展时影响较大 2. 采用TCP长链接，穿透内外网时会较麻烦 3. 各个语言平台的功能实现完成度稍有差异 4. 分布式部署时，实现稍麻烦 |
| JSON+HTTP | 1. 可扩展性更强 2. 实现简单 3. 无连接、无状态，分布式部署更容易 | 1. 性能相对较差 2. JSON属于纯文本编码，压缩率不如Thrift |

建议采用JSON+HTTP的方式。

### 单服务器情况



说明：

1. 三类消息中，认证消息是并发量最高的，而注册同步用户、监听、消息推送则是服务器内部通讯消息，其中注册同步用户、监听消息并发量小，消息推送则与用户认证数量维持一致。
2. 授权消息观察者与统一认证服务之间建议采用**非同步**通知方式，以减轻网络压力，使用缓存及超时机制定期同步授权用户信息到观察者中。

### 多服务器情况



在单服务器访问量支撑不住的时候，需要横向扩展到多服务器以缓解服务器压力，考虑到接口尽量少变化的原则，对于并发量最大的认证消息，目前可以采取的方案是增加一个前置*负载均衡服务*，此服务负责分发请求到后端各个独立认证服务上，分发规则可以考虑使用“*一致性hash*”算法。

对于授权信息观察者消息和用户同步消息，因为涉及到在多个认证服务上进行消息通讯的事情，所以会导致前端应用修改较大，所以引入了*统一认证代理服务*，此服务主要目前是接收其它应用的注册请求和用户同步请求，同时将请求分别转发给各个统一认证服务上。此种方案的优点是无须修改接口，即可满足分布式的需求。

#### 风险及规避方法

1. **单点问题***负载均衡服务*和*统一认证代理服务*都属于单点服务，如果其中一台出现问题，会导致各自负责的业务无法使用，对于此类问题，最简单的解决办法是每个服务增加一个热备，在某台服务出现问题时主动切换到备份机器提供服务。
2. **服务的增加或减少导致负载路由异常**

在某台统一认证服务异常拓机或增加新认证服务器时，会导致一致性Hash路由过来的授权检查请求无法获取到对应的授权认证信息。

对于此类问题，有两种方案可选：

1. 在数据量较小的情况，可以考虑在*统一认证代理服务*上增加一个缓存，用于存储所有的认证请求，在出现上述问题时，在此服务器上查找对应的授权信息并保存在本地缓存。
2. 如果数据量较大，可以采用冗余备份，使用如下规则:

S2 = S1 + int(N/2)

其中S2表示备份存储的位置，S1表示当前服务器，N表示总服务器数

例如：

S1 = 3，N=7，那么S2 = 3 + (7 /2) = 3 + 3 = 6

S1 = 3，N=5，那么S2 = 3 + (5 /2) = 3 + 2 = 5

以此类推，当某台服务器出现异常时，可以通过方法在对应服务器找到认证数据。

因为新增服务器的情况比较少，所以一般情况只考虑服务器异常的时候，以及服务器恢复加入的情况，所以N一般保持不变，只是有状态区分。

1. **内部应用获取授权推送消息**

对于内部应用服务需要获取到当前已认证的授权信息来说，如果在*统一认证代理服务*上转发会导致此服务压力过大，因此由各个认证服务直接转发到应用监听者上

## 内部数据流

### 注册及更新



### 登录授权



### 账号关联、查询



### 监听推送



## 用户体系



说明：

1. 统一管理用户的账户与应用的关系，以及用户的基础 信息
2. 第三方系统的账号可以通过绑定或注册的方式添加到账户子账号中
3. 第三方系统的账号注册验证成功后，必须填写用户的手机号码及密码，成功后可以使用第三方账号、设置的密码登录
4. 统一认证中心统一使用UID作为用户的唯一标志，而不考虑是哪个应用的用户
5. 授权信息可以保存需要授权的应用范围（即：此次授权哪些应用可以使用）

## 数据关系



# 系统方案

系统用例图



注：上图中系统初始化等基本功能并未画出

## 业务接口



### 约定

#### 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
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

#### 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

#### 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

1. **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HTTP响应码

|  |  |  |
| --- | --- | --- |
| **响应码** | **方法** | **说明** |
| 200 | OK - [GET] | 服务器成功返回用户请求的数据，该操作是幂等的（Idempotent） |
| 201 | CREATED  [POST/PUT/PATCH] | 用户新建或修改数据成功 |
| 202 | Accepted - [\*] | 表示一个请求已经进入后台排队（异步任务） |
| 204 | NO CONTENT  [DELETE] | 用户删除数据成功 |
| 400 | INVALID REQUEST  [POST/PUT/PATCH] | 用户发出的请求有错误，服务器没有进行新建或修改数据的操作，该操作是幂等的 |
| 401 | Unauthorized-[\*] | 表示用户没有权限（令牌、用户名、密码错误） |
| 403 | Forbidden-[\*] | 表示用户得到授权（与401错误相对），但是访问是被禁止的 |
| 404 | NOT FOUND - [\*] | 用户发出的请求针对的是不存在的记录，服务器没有进行操作，该操作是幂等的 |
| 406 | Not Acceptable - [GET] | 用户请求的格式不可得（比如用户请求JSON格式，但是只有XML格式） |
| 410 | Gone -[GET] | 用户请求的资源被永久删除，且不会再得到的 |
| 422 | Unprocesable entity - [POST/PUT/PATCH] | 当创建一个对象时，发生一个验证错误 |
| 500 | INTERNAL SERVER ERROR - [\*] | 服务器发生错误，用户将无法判断发出的请求是否成功 |

#### 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。

#### 错误码约定

Code一共8位，每两位为一组，其含义如下：

**10 10 21 10**

应用代码

模块代码

功能代码

异常代码

现有约定如下：

**应用：**

统一认证服务平台：10

**模块**

统一认证模块：00

用户管理：01

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

### 验证用户参数

#### 功能说明

验证用户的关键参数是否已被使用。

#### 约定

**Resource** Account/Parameter

**METHOD**  GET

#### 接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **必须提供** | **说明** |
| appId | String | Yes | 请求的应用ID |
| Email | String | No | 邮箱地址 |
| Mobile | String | No | 手机号码 |

#### 错误代码

#### 返回值

{

UserCode: Boolean, --用户代码是否可用，true-可用，false-不可用

Email: string, --邮箱，true-可用，false-不可用

Mobile：String，--手机号，true-可用，false-不可用

}

注：验证参数是可选项，如果有输入则提供验证，并返回验证结果

如：

送入：Mobile: 13311223344

返回:

{

Mobile: true

}

送入：Mobile：13311223344，Email：[test@test.com](mailto:test@test.com)

返回

{

Mobile: true,

Email: true

}

### 用户注册

#### 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。



#### 约定

**Resource** Account

**METHOD**  POST

#### 接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **必须提供** | **说明** |
| appId | String | Yes | 用户所属应用 |
| userCode | 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 |

#### 错误代码

#### 返回值

{

Uid:string, --生成的用户唯一ID

appId:string, --所属应用

userCode:string --用户编码

}

### 更新用户信息

#### 功能说明

更新用户的基本信息。



#### 约定

**Resource** Account

**METHOD**  PUT

#### 接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **必须提供** | **说明** |
| accessToken | String | Yes | 获得的授权访问Token |
| appId | String | Yes | 请求的应用ID |
| Uid | String | Yes | 统一认证中心分配的用户唯一ID |
| UserInfo | String | No | 用户的基础信息，使用JSON二次包装 |

#### 错误代码

#### 返回值

{

true|false --成功返回true，否则返回false

}

### 更新用户密码

#### 功能说明

更新用户密码。

#### 约定

**Resource** Account

**METHOD**  PUT

#### 接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **必须提供** | **说明** |
| accessToken | String | Yes | 获得的授权访问Token |
| appId | String | Yes | 请求的应用ID |
| Uid | String | Yes | 统一认证中心分配的用户唯一ID |
| Oldpwd | String | Yes | 原密码，经过MD5与用户名混淆后的值 |
| Password | String | Yes | 新密码，对原始密码经过MD5加密后的值 |

#### 错误代码

#### 返回值

{

true|false --成功返回true，否则返回false

}

### 删除用户

#### 功能说明

删除指定用户。

#### 约定

**Resource** Account

**METHOD**  DELETE

#### 接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **必须提供** | **说明** |
| accessToken | String | Yes | 获得的授权访问Token |
| appId | String | Yes | 请求的应用ID |
| Uid | String | Yes | 统一认证中心分配的用户唯一ID |

#### 错误代码

#### 返回值

{

true|false --成功返回true，否则返回false

}

***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。***

### 获取验证码

#### 功能说明

应用在需要进行校验邮箱、手机或其它通讯设备时，认证平台提供一个验证码，应用获取到验证码后自行组织验证内容并发送出。

#### 约定

**Resource** Account/ActiveCode

**METHOD**  POST

#### 接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **必须提供** | **说明** |
| accessToken | String | Yes | 获得的授权访问Token |
| appId | String | Yes | 请求的应用ID |
| Uid | String | Yes | 要激活的用户ID |
| type | Int | Yes | 1-验证，2-找回密码 |
| object | Int | Yes | 激活对象：1.手机号2.邮箱地址 |

#### 错误代码

10011001 对象已验证  
10011002 验证对象不存在

#### 返回值

{

ActiveCode：String --用于校验的验证码

Uid:String --激活码对应的用户ID

}

### 验证验证码

### 绑定应用账号

#### 功能说明

将已注册的用户绑定到未登录过的App上，如果应用以前使用临时账号记录用户的信息，则可以将临时账号也绑定上来。也支持绑定第三方应用的账号。



（绑定内部应用）



（绑定第三方应用账号）

#### 约定

**Resource** Account/Relation

**METHOD**  POST

#### 接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **必须提供** | **说明** |
| accessToken | String | Yes | 获得的授权访问Token |
| AppID | String | Yes | 要绑定的应用ID，由管理员预先设置好，如：第三方的微信、内部应用等 |
| Account | String | Yes | 应用使用的账号ID  内部应用为临时ID，没有则设置为登录账号。  外部应用为第三方登录账号 |

#### 错误代码

10011401 AppID不存在

10011402 账号为空

#### 返回值

{

true|false --成功返回true，否则返回false

}

### 查询绑定账号

#### 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。

#### 约定

**Resource** Account/Relation

**METHOD**  GET

#### 接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **必须提供** | **说明** |
| accessToken | String | Yes | 获得的授权访问Token |
| AppID | String | Yes | 请求的应用ID |

#### 错误代码

#### 返回值

[uidObj1, uidObj2, ……, uidObjn]

每个对象结构如下:

{

Account: String, --账号名称

AppID:String, --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

}

### 认证接口

#### 功能说明

提供登录认证的操作，同时取得接口访问授权 Key

#### 约定

**Resource** AccessToken

**METHOD** POST

#### 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 |
| appId | String | Yes | 执行登录的AppID |
| 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 |

**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)+uid)

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)+uid)+timestamp)

**timestamp:**

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

#### 错误代码

10000100 时间戳已过期  
10000101 用户不存在  
10000102 密码错误

#### 返回值

{

uid: string, --对应用户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

}

#### 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

1. 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

1. 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

### 更新授权码

#### 功能说明

在授权码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。

#### 约定

**Resource** AccessToken

**METHOD**  PUT

#### 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 |
| AppID | String | Yes | 当前应用ID |

#### 错误代码

10000110 refreshToken已过期  
10000111 用户不存在

#### 返回值

{

uid: string, --对应用户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

}

### 授权查询

#### 功能说明

验证授权码是否可用。

#### 约定

**Resource** AccessToken

**METHOD**  GET

#### 接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **必须提供** | **说明** |
| accessToken | String | Yes | 获得的授权访问Token |
| appId | String | Yes | 请求的AppID |

#### 错误代码

10000120 accessToken已过期  
10000121 accessToken不存在

#### 返回值

{

true|false --成功返回true，否则返回false

}

### 注销授权

#### 功能说明

注销已经授权的请求。

#### 约定

**Resource** AccessToken

**METHOD**  DELETE

#### 接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **必须提供** | **说明** |
| accessToken | String | Yes | 获得的授权访问Token |
| appId | String | Yes | 请求的AppID |

#### 错误代码

#### 返回值

{

true|false --成功返回true，否则返回false

}

### 用户授权推送注册

#### 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。

#### 约定

**Resource**  Observer

**METHOD**  POST

#### 接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **必须提供** | **说明** |
| AppID | String | Yes | 当前应用ID |
| PushAppUser | Bool | No | 是否只推送符合应用的用户，默认为true，如果指定为false则将推送所有认证通过的用户 |
| Callback | String | Yes | 回调Url |

#### 错误代码

130 AppID不存在  
131 回调地址不合规范

#### 返回值

{

true|false --成功返回true，否则返回false

}

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

### 用户授权信息推送

#### 功能说明

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，而会在缓冲区满或到指定推送时间后主动进行推送。

#### 推送内容

以数组的格式推送过来

[authObj1, authObj2,……]

其中每个对象格式如下：

{

uid: string, --对应用户ID

appId: String, --用户所属的应用ID

accessToken: string, --访问Token

expireTime: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

## 扩展接口

### 文件访问接口

提供本地文件的访问接口，如：日志、配置文件读写。

### 数据访问接口

提供对数据库访问接口的抽象，业务层仅使用接口，交由不同的数据库处理层实现数据库的驱动和存储。以便于后期数据迁移。因此功能较常用，此处不作特殊说明。

可以考虑基于Spring框架进行设计。

### 缓存接口

缓存接口抽象出来主要目的是为了后续缓存迁移工作，在用户量小的情况，可以使用本地缓存来实现，用户量大的时候，则需要使用外部缓存，如Memcached、Redis等。

本地缓存的实现可以考虑基于LRU或相关的扩展算法来实现。

在统一认证服务中，缓存主要是缓存用户信息以及对应的授权认证信息。



### 集群服务接口

保留，暂时不考虑。

## 日志记录

目前日志统一记录在文件中，格式统一。在任何主要操作，如：查询、修改、登录、注销等都需要记录下用户的操作信息，记录的信息包括：调用的方法、操作人、操作参数、返回值等。

## 系统初始化

系统初始化时，需要初始化部份功能，其中对于用户信息，建议延时加载到缓存中。

## 登录认证

登录流程活动图



登录交互序列图



## 授权信息管理

活动图

**注册监听对象**



**同步授权信息数据**



## 同步用户数据

活动图



**新增流程**



说明：

1. 对于内部系统来说，1.3，1.5步骤是新增加的
2. 如果未能获取到UID，内部系统则无法正常登录，所以如果UID未获取到，则有两种处理方式：
3. 注册失败，取消已保存的用户
4. 将用户放到待处理队列中，后续用同步进程继续执行1.3，1.6步骤
5. 上述序列图未考虑异常情况

## 数据缓存

### 用户数据

存储当前登录的用户信息，更新时机为：

1. 程序启动（需要考虑，可以不实现，需要时再加载）
2. 用户新增
3. 用户修改
4. 查找用户（从数据库加载）

### 授权数据

存储授权成功的数据，因此数据需要实例化，可以考虑使用redis之类的第三方应用来存储，更新时机如下：

1. 程序启动（如果是Redis存储则不考虑）
2. 授权成功
3. 授权码超时（删除）
4. 注销
5. 更新授权码

### 观察者数据

存储需要监听授权信息的观察者应用信息，更新时机如下：

1. 程序启动（从数据库中加载）
2. 监听者注册后
3. 监听者取消注册后
4. 监听者变更注册后

## 性能统计

## 相关开源项目

1. 模版引擎grovvy
2. Java动态编译库:javassit、cglib
3. JSON序列化：Gson、fastJSON
4. 日志记录：log4j、slf4j
5. 数据库:mysql、PostgreSQL、Mongodb
6. 缓存：memcache、redis、Ehcache
7. IOC框架：Spring
8. ORM框架：Hibernate

# 实施计划

# 风险评估

# 性能指标

单服务器支持并发量为800~1000/s，单次请求响应时间在2秒内